Machine Learning (2023) 112:3851–3879
https://doi.org/10.1007/s10994-022-06203-x

LADDER: Latent boundary-guided adversarial training

Xiaowei Zhou[1,2] · **Ivor W. Tsang**[1,3] · **Jie Yin**[4]

Received: 21 September 2020 / Revised: 13 May 2022 / Accepted: 26 May 2022 /
Published online: 9 September 2022

Abstract
Deep Neural Networks (DNNs) have recently achieved great success in many classification tasks. Unfortunately, they are vulnerable to adversarial attacks that generate adversarial examples with a small perturbation to fool DNN models, especially in model sharing scenarios. Adversarial training is proved to be the most effective strategy that injects adversarial examples into model training to improve the robustness of DNN models against adversarial attacks. However, adversarial training based on the existing adversarial examples fails to generalize well to standard, unperturbed test data. To achieve a better trade-off between standard accuracy and adversarial robustness, we propose a novel adversarial training framework called *LAtent bounDary-guided aDvErsarial tRaining* (LADDER) that adversarially trains DNN models on latent boundary-guided adversarial examples. As opposed to most of the existing methods that generate adversarial examples in the input space, LADDER generates a myriad of high-quality adversarial examples through adding perturbations to latent features. The perturbations are made along the normal of the decision boundary constructed by an SVM with an attention mechanism. We analyze the merits of our generated boundary-guided adversarial examples from a boundary field perspective and visualization view. Extensive experiments and detailed analysis on MNIST, SVHN, CelebA, and CIFAR-10 validate the effectiveness of LADDER in achieving a better trade-off between standard accuracy and adversarial robustness as compared with vanilla DNNs and competitive baselines.

Editors: Daniel Fremont, Mykel Kochenderfer, Alessio Lomuscio, Dragos Margineantu, Cheng Soon-Ong.

✉ Xiaowei Zhou
Xiaowei.Zhou@student.uts.edu.au

Ivor W. Tsang
ivor.tsang@uts.edu.au

Jie Yin
jie.yin@sydney.edu.au

[1] Australian Artificial Intelligence Institute (AAII), School of Computer Science, FEIT, University of Technology Sydney, Sydney, NSW 2007, Australia

[2] Data61, CSIRO, Sydney, NSW 2015, Australia

[3] Center for Frontier AI Research, Agency for Science, Technology and Research (A*STAR), Singapore, Singapore

[4] Discipline of Business Analytics, The University of Sydney, Sydney, NSW 2006, Australia

Keywords Adversarial training · Adversarial robustness · Boundary-guided generation

1 Introduction

In recent years, deep neural networks (DNNs) have been successfully applied to empower many advanced applications, such as image processing (Huang et al., 2017; Shi et al., 2021), speech generation (Seide et al., 2011; Amodei et al., 2016) and natural language processing (Fedus et al., 2018; Alikaniotis et al., 2016). Nevertheless, training a DNN model often requires large amounts of labeled data and significant efforts of parameter tuning. As such, this catalyzes new ways of developing DNN models as a service that can be shared by a third party. This development leads to many offline and online Machine Learning as a Service (MLaaS) platforms that provide shared services for various tasks based on DNN models, such as image and video analysis from the AWS pre-trained AI Services (Amazon, 2019), and powerful image analysis from Google Cloud Vision (Google, 2019).

In such model sharing scenarios, the increasing use of DNN models, however, has raised serious security and reliability concerns. As illustrated in Fig. 1, the service provider trains a DNN model using the training data that they have collected, expecting it to achieve high classification accuracy on test samples with similar distributions. End-users then provide their own test samples, mostly unknown to the service provider in advance, into the shared model to obtain prediction results. However, very often, test samples are likely to be mixed with unknown adversarial examples (Szegedy et al., 2013; Yuan et al., 2019), which are generated by attackers by adding small and hardly visible perturbations to standard test samples. What makes it even worse is that adversarial examples can be generated by various types of unknown attacks, resulting in a dramatic drop in classification accuracy. This is due to the fact that the shared DNN models are not robustly trained to defend against unknown adversarial attacks before they are released as a service. Thus, research works have been proposed to improve the robustness of DNN models against adversarial attacks.

Adversarial training (Goodfellow et al., 2015; Shaham et al., 2018; Shrivastava et al., 2017) is one of the most successful techniques for improving the robustness of DNN models against adversarial attacks. Its key idea is to augment the training data with adversarial examples to retrain DNN models. However, as studied in (Tsipras et al., 2019), there exists a trade-off between standard accuracy on clean data and adversarial robustness. Better robustness often leads to worse classification accuracy on clean data. To address this,

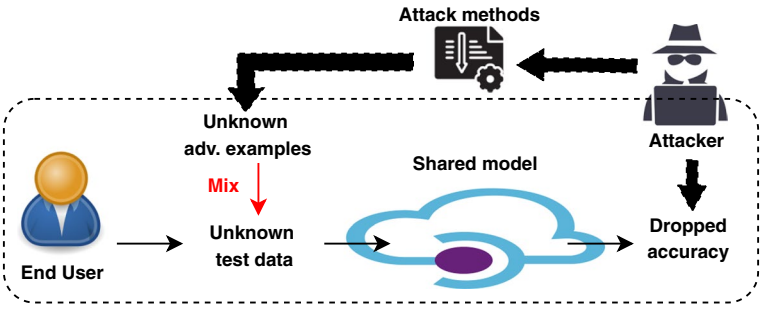

Fig. 1 Attacks in model sharing scenarios. The end user performs classification tasks through the shared model without the knowledge on adversarial examples generated by attackers to attack the shared model

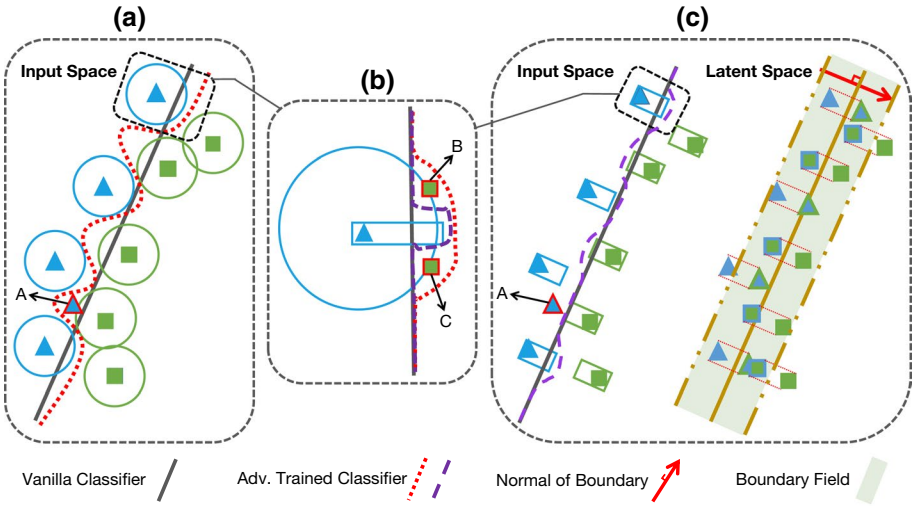

Fig. 2 **a**: Adversarial examples generated by the existing methods are often within a ball in the input space, which leads the decision boundary of the adversarially trained classifier (dotted line) to dramatically change. The legitimate sample, sample A, would be misclassified. **c**: Our LADDER method perturbs latent features of samples within the boundary field in the latent space. The generated adversarial examples would reside in a more restricted area. Sample A would thus be classified correctly. **b**: The adversarially trained classifier by existing adversarial examples (red dotted line) would misclassify legitimate samples (sample B and C), which would hurt the standard accuracy. The adversarially trained classifier using our method (purple dashed line) would change less remarkably than the existing one (red dotted line), which would correctly classify the samples (Color figure online)

several recent studies (Zhang et al., 2019b; Ding et al., 2020; Wang et al., 2020) have been proposed to add additional regularization terms into the loss function of adversarial training. These methods employ the modified adversarial training loss function with regularization to achieve better generalization and adversarial robustness. This has raised a research question: Could adversarial examples generated by the existing methods account for the trade-off between standard accuracy and adversarial robustness? This motivates us to explore the curses and blessings of adversarial examples for achieving a better trade-off between standard accuracy and robustness against adversarial attacks.

The adversarial examples generated by the existing methods suffer from two major curses. First, most of these methods (Goodfellow et al., 2015; Madry et al., 2018; Papernot et al., 2016b; Carlini & Wagner, 2017) generate adversarial examples through adding a small perturbation to legitimate samples in the input space, which often only craft adversarial examples of repeating patterns. The DNN models adversarially trained with these examples would be effective only in defending very specific types of adversarial attacks, but still vulnerable to other adversarial attacks. Thus, this presses a need for increasing the diversity of generated adversarial examples so that DNN models can fully explore the unknown data space to improve the robustness against adversarial attacks.

Second, the generated adversarial samples might hurt the standard accuracy of DNN models on clean (legitimate) samples. As shown in Fig. 2a, the adversarial examples generated by most existing methods are within a ball in the input space. When adversarially trained with these samples, the decision boundary of the DNN classifier would dramatically change as compared with the original one. This leads legitimate samples (sample A)

to be misclassified. Moreover, as perturbations are added in the input space, the generated adversarial examples would contain lots of noise. This can be seen from adversarial examples (in Fig. 6) generated by FGSM (Goodfellow et al., 2015) or JSMA (Papernot et al., 2016b). Consequently, it would hurt the standard accuracy of adversarially trained models. This inspires us to generate adversarial examples of better quality through the latent space.

Furthermore, existing methods treat all examples equally while generating adversarial examples, but neglect the decision boundary information. Our work stems from the observation that incorporating information about the decision boundary in the latent space into the generation of adversarial examples would be a blessing to achieve better generalization on standard clean data. As shown in Fig. 2c, latent features of samples are perturbed along the normal of decision boundary to move within the boundary field, i.e., the nearby area of the decision boundary. The generated adversarial examples would thus reside in a restricted area and prevent the adversarially trained classifier from misclassifying legitimate samples (sample B and C), which, however, could be misclassified by existing methods. Motivated by this observation, we focus on how to leverage the decision boundary to guide the generation of adversarial examples, aiming to achieve a better trade-off between standard accuracy and adversarial robustness.

In this paper, we propose a novel adversarial training framework called Latent Boundary-guided Adversarial Training (LADDER), which adversarially trains DNN models with myriad adversarial examples generated based on the decision boundary in a latent space. Unlike some of the existing methods that operate in the input space and generate noisy adversarial examples of repeated patterns, LADDER generates high-quality and diverse adversarial examples by adding perturbations to latent features. The generation of adversarial examples is guided by the normal of decision boundary in the latent space, which is learned via a linear support vector machine (SVM) (Boser et al., 1992) with an attention mechanism. After adversarial training on generated adversarial examples, the adversarially trained DNN model is effective in achieving a trade-off between standard accuracy and robustness against adversarial attacks. Comprehensive experiments and analyses are conducted on MNIST, SVHN, CelebA, and CIFAR-10 to verify the effectiveness of the proposed method.

The novelty and contribution of this paper are three-fold:

- We analyze the curses and blessings of adversarial examples for adversarial training and explain the advantages of latent boundary-guided solution.
- We propose a new method called LADDER that generates high-quality and diverse adversarial examples by adding boundary-guided perturbations in a latent feature space.
- After adversarial training on the generated adversarial examples, LADDER achieves a better trade-off between standard accuracy and adversarial robustness as compared with vanilla DNNs and competitive baselines.

2 Related work

This section reviews two branches of related literature: adversarial attack methods and adversarial defence methods.

2.1 Adversarial attack

From the methodology point of view, most of the existing adversarial attack methods can be grouped into two categories: *gradient-based attacks* (Goodfellow et al., 2015; Madry et al., 2018; Papernot et al., 2016b) and *decision-based attacks* (Carlini & Wagner, 2017; Song et al., 2018; Moosavi-Dezfooli et al., 2016; Su et al., 2019).

Gradient-based attacks: Gradient-based attacks mainly add perturbations in the direction of the gradient of loss function with respect to the input sample. (Goodfellow et al., 2015) proposed the fast gradient sign method (FGSM) that uses the sign of gradient ($\nabla_{\mathbf{x}} J(\theta, \mathbf{x}, y)$) of loss function with respect to input examples as perturbation. Built upon FGSM, the one-step attack method, (Madry et al., 2018) proposed a multi-step attack method called PGD. PGD iteratively uses the gradient information and generates adversarial examples on the results of the last step. Similarly, DI^2-FGSM (Xie et al., 2019) generates adversarial examples on the results of last step, but it uses the gradients of stochastic transformed inputs rather than the original ones. (Papernot et al., 2016b) introduced saliency map based on Jacobian matrix into the generation of adversarial examples. The saliency values computed by forward derivative of a target model are used as an indicator to determine the locations in the input examples to add perturbations. This method is called Jacobian saliency map attack (JSMA).

Decision-based attacks: Decision-based attacks manipulate the labels of training data to make the learned DNN model beneficial to their specific purposes. This line of methods uses Eq. (1) as a measure, i.e., changing the original label to target label, to generate adversarial examples. For a given classifier f, the predicted label of an input sample $\mathbf{x}$ is defined as $f(\mathbf{x})$, and $f(\mathbf{x}')$ is the label of adversarial example $\mathbf{x}'$:

$$\mathbf{x}' = \mathbf{x} + \boldsymbol{\delta}, \text{ s.t. } f(\mathbf{x}') \neq f(\mathbf{x}), \tag{1}$$

where $\boldsymbol{\delta}$ is a small perturbation added to an input sample $\mathbf{x}$. $\mathbf{x}'$ is the generated adversarial example. Carlini and (Carlini & Wagner, 2017) proposed an approach, CW, to generate adversarial examples by adding small changes to the original images in the input space. CW tries to minimize the distance between benign examples and adversarial ones, while enforcing the labels of adversarial examples as the targeted ones. A deep learning based attack method was developed by Song et al. (2018). This method explored the AC-GAN (Odena et al., 2017) latent space to generate adversarial images, which could most likely mislead the targeted classifier. GeoDA (Rahmati et al., 2020) estimates the decision boundary for each data point in the input space to generate adversarial examples. In contrast, our LADDER trains an SVM to obtain the decision boundary between any two classes in the latent space.

Other adversarial attacks: Recently, distribution-based methods have also been proposed for adversarial attack. DAA (Zheng et al., 2019), HMCAM (Wang et al., 2020) and $\mathcal{N}$attack (Li et al., 2019) generate diverse adversarial examples through modeling their probability distribution in the input space. The goal of DAA (Zheng et al., 2019) is to generate globally optimal adversarial examples and $\mathcal{N}$attack (Li et al., 2019) aims to develop a powerful black-box adversarial attack method. HMCAM (Wang et al., 2020) generates a sequence of adversarial examples to improve adversarial robustness. In contrast, our LADDER, as one adversarial training based defence method, generates adversarial examples individually for each input to achieve a better trade-off between standard accuracy and adversarial robustness.

Apart from distribution-based methods, (Croce & Hein, 2020b) proposed an ensemble attack method called AutoAttack. This method firstly extends the PGD attack method into APGD_{CE} by automatically choosing step sizes and APGD_{DLS} by using a difference of logits ratio (DLR) loss. Then, four attack methods, including APGD_{CE}, APGD_{DLS}, FAB (Croce & Hein, 202a) and square attack (Andriushchenko et al., 2020) are combined as AutoAttack.

From the knowledge accessibility point of view, the adversarial attacks can be divided into *white-box attacks* and *black-box attacks*. Under white-box attack settings, the attackers have access to full knowledge about the target model, i.e., model structure and model parameters. On the contrary, the attackers have no knowledge about the target model under black-box attack settings. In this work, we are mainly concerned with model sharing scenarios, where model structure and model parameters are unknown to attackers. Thus, we focus on defending black-box attacks.

2.2 Adversarial defence

For various types of adversarial attacks, a key research question is, how can we improve the adversarial robustness of a DNN model before it is deployed as a service? In response, adversarial defence strategies have been proposed to mitigate the effect of adversarial attacks (Papernot et al., 2016c; Papernot & McDaniel, 2017; Samangouei et al., 2018; Meng & Chen, 2017; Guo et al., 2018; Kannan et al., 2018; Mustafa et al., 2019; Xiao et al., 2020).

Gradient masking methods: Gradient masking methods (Papernot et al. 2017) construct a model which does not provide useful gradients to be attacked. For example, *Defensive distillation* (Papernot et al., 2016c; Papernot & McDaniel, 2017) learns a smooth targeted defence model with d training times, where the predicted labels from $(d-1)$-th model are used as ground truth to train the d-th model except for the first time.

Clean sample reconstruction: Defense-GAN (Samangouei et al., 2018), DIPDefend (Dai et al., 2020) and MagNet (Meng & Chen, 2017) are three typical methods that remove the perturbation added on adversarial examples to reconstruct a clean sample similar to the legitimate one. The reconstructed examples can be easily recognised by the model, compared with adversarial examples.

Adversarial training: Among others, *adversarial training* (Goodfellow et al., 2015; Shaham et al., 2018; Shrivastava et al., 2017; Kurakin et al., 2016) is proved as one of the most effective defence methods, which augments the training data with adversarial examples when training the targeted model. It can be achieved either by training the targeted model with original samples augmented with adversarial examples (Kurakin et al., 2016) or with a modified loss function (Goodfellow et al., 2015). Our LADDER falls into the realm of adversarial training based defence methods.

Recently, Tsipras et al. (2019) found the adversarial robustness is at odds with the standard accuracy on clean samples. Several recent methods were proposed to trade the adversarial robustness off the standard accuracy. Most of these methods considered adding different regularization terms to the adversarial training loss to achieve a better trade-off (Zhang et al., 2019b; Ding et al., 2020; Wang et al., 2020; Zhang et al., 2021). TRADES (Zhang et al. 2019b) is a regularization-based method that minimizes the loss between the predicted labels and ground truth of legitimate samples as well as the 'difference' between the predictions of legitimate samples and the corresponding adversarial examples. The 'difference' term was regarded as the regularization. GAIRAT (Zhang et al., 2021) uses the distance to the decision boundary to assign a weight for each adversarial example in the

adversarial training loss. The weight is estimated based on the difficulty of attacking the input by PGD rather than the actual distance to the decision boundary. On the contrary, our LADDER derives an explicit decision boundary to generate adversarial examples for adversarial training. It is worth noting that, the existing regularization based methods and our LADDER method provide two different views to achieve a better trade-off between the standard accuracy and adversarial robustness. Our LADDER method can be used in combination with these regularization based methods to further boost their performance (See our empirical results in Sect. 4.5).

Different from the existing methods that use a regularized loss to improve the trade-off, AVmixup (Lee et al., 2020) is a data augmentation method that uses linear interpolation to acquire the augmented examples in the input space based on adversarial examples generated by PGD attack. However, as proved in Manifold Mixup (Verma et al., 2019a), AVmixup may produce less semantically meaningful examples as a result of linear interpolation in the input space. In contrast, our LADDER method alleviates this issue by adding perturbations along the normal of the decision boundary in the latent space.

3 Latent boundary-guided adversarial training

To achieve a better trade-off between standard accuracy and adversarial robustness, LADDER aims to generate better adversarial examples based on the decision boundary constructed in a latent space. Perturbations are added to latent features along the normal of decision boundary and inverted to the input space by the trained generator. Through adversarial training with the generated adversarial examples, LADDER achieves a better trade-off between standard accuracy and adversarial robustness.

For clarity, we first define key notations and symbols. Define $\mathbb{X} = \{\mathbf{x_1}, ..., \mathbf{x}_n\}$ as the set of samples, where n is the number of samples. For each sample $\mathbf{x}_i$, $\mathbf{z}_i$ is the latent feature vector extracted from a trained DNN model. $\mathbf{d}$ is the normal of the decision boundary. ϵ is the perturbation added to the latent feature vector $\mathbf{z}_i$. $\hat{\mathbf{x}}_i$ is the generated sample from the latent feature $\mathbf{z}_i$.

3.1 Latent boundary-guided generation

The adversarial examples are generated through perturbating the latent space, guided by the decision boundary, which is obtained from an attention SVM learned on latent features.

3.1.1 Boundary-guided attention

To approximate local decision boundaries in the latent space, we train a linear SVM with an attention mechanism (Zhang et al., 2019a) with latent features of a DNN model. This idea is grounded on the theoretical proof by Li et al. (2018) that, the last layer of neural networks trained by cross-entropy loss converges to a linear SVM. For any neural network used for binary or multi-class classification, when the cross-entropy loss gradually approaches to 0, the last layer weights of neural networks would converge to the solution of an SVM. Specifically, we use the latent feature $\mathbf{z}$, the input to the last layer of the DNN model to train a linear SVM with an attention mechanism. The trained linear SVM provides an explicit margin as compared to other linear models such as a linear mixture model.

By employing an attention mechanism when training the linear SVM, our aim is to capture a better representation with different weights assigned to different elements of latent features. To this end, an attention layer is added to process latent features, before passing them to the SVM. The attention layer is defined as follows:

$$\boldsymbol{\alpha}_i = \tanh\left(conv(\mathbf{z}_i)\right),\ \boldsymbol{\beta}_i^j = \frac{\exp(\boldsymbol{\alpha}_i^j)}{\sum_{j=1}^{N}\exp(\boldsymbol{\alpha}_i^j)},\ \text{and}\ \ \mathbf{z}_i^{att} = \boldsymbol{\beta}_i \mathbf{z_i}, \tag{2}$$

where $\mathbf{z}_i$ is a latent feature vector, the input to the last layer of the DNN model; *conv* is the convolutional operation; tanh is the activation function; $\boldsymbol{\beta}_i$ is the attention weight vector that can be learned; $\mathbf{z}_i^{att}$ is the output after applying attention weights on latent feature vector $\mathbf{z}_i$.

3.1.2 Latent feature perturbation

After training an SVM with attention, the latent features of each sample are perturbed along the normal of the decision boundary of the SVM. The normal $\mathbf{d}$ provides a direction to guide the generation, where the attention weight $\boldsymbol{\beta}$ captures the importance of different components of latent features to move across the boundary. Different perturbations ϵ can be added to the same latent features $\mathbf{z}_i$ to obtain the perturbed latent features $\mathbf{z}_i^j$ by:

$$\mathbf{z}_i^j = \mathbf{z}_i + \epsilon^j \boldsymbol{\beta}_i \mathbf{d}, \tag{3}$$

where vector $\mathbf{d}$ is the normal of the decision boundary of the linear SVM; $\boldsymbol{\beta}_i$ is the attention weight vector obtained by the SVM for each sample; $\epsilon^j > 0$ represents the perturbation; j is the index of different perturbation.

When the perturbation is big enough, the class label of the perturbed latent features would change from positive to negative, or vice versa. That means it would cross the decision boundary of the DNN model. As shown in Fig. 3, perturbed latent features $\mathbf{z}_i^1$ move from the left side of the decision boundary to the right side. As the perturbation continues to increase, perturbed latent features $\mathbf{z}_i^2$ would move far away from the decision boundary. The effect of perturbation will later be empirically investigated in Sect. 4.4.

3.1.3 Boundary-guided generation

To enable humans to understand what changes happen in the input space, caused by the perturbations to latent features, we train a generator to invert the perturbed latent features to the input space.

For a specific DNN model (i.e., LeNet), we learn a generator $\hat{G}$ on the training set $\mathbb{X}_{train} = \{\mathbf{x}_1, ..., \mathbf{x}_n\}$ to map latent features to the input space. As shown in Fig. 3, each sample in the training set is fed into the DNN model, to extract the corresponding latent features. The output of the last fully connected layer of a DNN model can be used to construct the set of latent features $\mathbb{Z}_{train} = \{\mathbf{z}_1, ..., \mathbf{z}_n\}$. Sample $\mathbf{x}_i$ and its corresponding latent features $\mathbf{z}_i$ are fed to the designed generator. The objective function of G over the neural network class $\mathcal{G}$ is defined as follows:

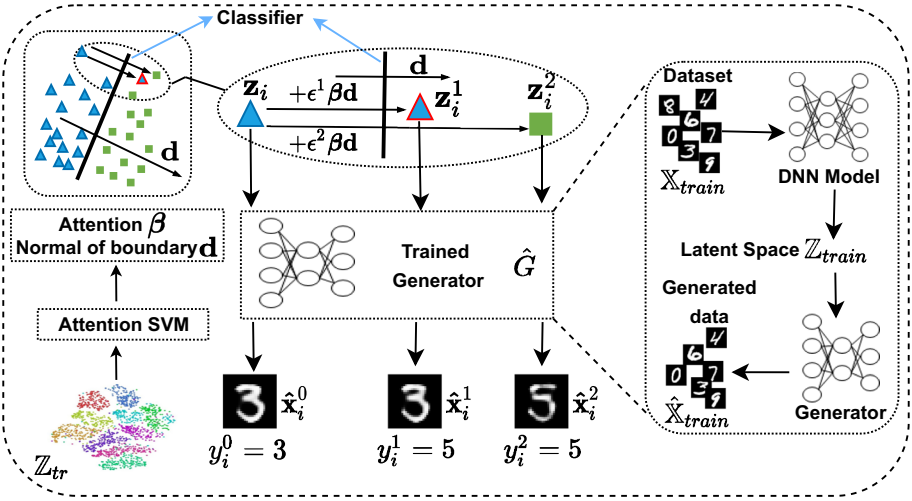

Fig. 3 Overview of *Latent Boundary-guided Adversarial Training*. LADDER generates adversarial examples by perturbing latent features alongside the normal of decision boundary obtained from an SVM with an attention mechanism. These generated adversarial examples are inverted to the input space via a trained generator to adversarially train the DNN model. $\mathbf{z}_i$ is the latent feature of one original sample $\mathbf{x}_i$; $\boldsymbol{\beta}$ is attention weight; $\mathbf{d}$ is the normal of the decision boundary; $\mathbf{z}_i^1$ and $\mathbf{z}_i^2$ are the perturbed latent features for generation; $\hat{\mathbf{x}}_i^j$ $(j = 0, 1, 2)$ are the generated images after perturbing the latent features; y_i^j $(j = 0, 1, 2)$ are the predicted labels of the generated images.

$$\hat{G} = \arg\min_{G \in \mathcal{G}} n^{-1} \sum_{i=1}^{n} \left\| \mathbf{x}_i - G(\mathbf{z}_i) \right\|_p^p, \tag{4}$$

where $\|\cdot\|_p$ denotes L_p norm, $p = 1$ *or* $p = 2$ in this paper; $\mathbf{z}_i = \Phi(\mathbf{x}_i)$, where Φ is a feature extractor in a DNN model. A mapping between the latent space and the input space is learned by optimizing Eq. (4).

The reconstructed sample $\hat{\mathbf{x}}_i$ can be obtained by passing $\mathbf{z}_i$ to the trained generator $\hat{G}$, that is, $\hat{\mathbf{x}}_i = \hat{G}\left(\mathbf{z}_i + \epsilon\boldsymbol{\beta}\mathbf{d}\right)$. For example, in Fig. 3, different latent features ($\mathbf{z}_i$, $\mathbf{z}_i^1$ and $\mathbf{z}_i^2$) are fed into the trained generator $\hat{G}$ to obtain the corresponding samples $\hat{\mathbf{x}}_i^j$ $(j = 0, 1, 2)$ in the input space.

When the generated samples are fed into the targeted DNN model, the predicted labels should be the same as the ground-truth label $\hat{y}$. That is, the following equation should be satisfied:

$$f\left(\hat{G}(\mathbf{z}_i^j)\right) = \begin{cases} \hat{y}_i, \ \textit{if } \mathbf{z}_i^j \textit{ not cross the boundary} \\ \hat{y}_{other}, \ \textit{if } \mathbf{z}_i^j \textit{ cross the boundary} \end{cases} \tag{5}$$

As shown in Fig. 3, samples $\hat{\mathbf{x}}_i^0$ and $\hat{\mathbf{x}}_i^2$ are generated from $\mathbf{z}_i$ and $\mathbf{z}_i^2$, with 0 and ϵ^2 perturbation added, respectively. Among them, $\mathbf{z}_i$ does not cross the boundary and the predicted label of $\hat{\mathbf{x}}_i^0$ is still the ground-truth label, $y_i^0 = 3$. For $\mathbf{z}_i^2$ that has crossed the boundary, the predicted label of its corresponding sample $\hat{\mathbf{x}}_i^2$ has changed to 5, $y_i^2 = 5$. This satisfies the rules specified by Eq. (5).

However, Eq. (5) does not always hold for some perturbed latent features. Figure 3 provides such an illustration. The perturbed $\mathbf{z}_i^1$ has crossed the boundary and the predicted label y_i^1 of its generated sample $\hat{\mathbf{x}}_i^1$ has also changed to 5. However, the ground-truth label

of $\hat{\mathbf{x}}_i^1$ is still 3. Such samples, whose predicted labels of the reconstructed samples and ground-truth labels are inconsistent, are adversarial examples that are effective to attack the targeted DNN model.

3.2 Latent boundary-guided adversarial training

To improve the adversarial robustness of the DNN models, we adopt the adversarial training method (Goodfellow et al., 2015; Shaham et al., 2018) that uses the generated adversarial examples to augment the training data for retraining. Specifically, we get the perturbed latent features $\mathbf{z}_i^j$ through Eq. (3) and pass $\mathbf{z}_i^j$ to the trained generator $\hat{G}$ to generate adversarial examples. Then, we adversarially train the DNN model through the following adversarial loss function:

$$\tilde{J} = \alpha J(\boldsymbol{\theta};\mathbf{x},y) + (1-\alpha)J\big(\boldsymbol{\theta};\hat{G}(\mathbf{z}+\epsilon\boldsymbol{\beta}\mathbf{d}),y\big), \tag{6}$$

where $J(\boldsymbol{\theta})$ is the original loss function of the DNN model and $\boldsymbol{\theta}$ is the parameter of the targeted DNN model. The first and second term is the loss for the original training samples $\mathbf{x}$ and the generated adversarial examples $\hat{G}(\mathbf{z}+\epsilon\boldsymbol{\beta}\mathbf{d})$, respectively. α is the weighting factor that trades off the two terms, which is usually set as 0.5. Through adversarial training on the generated boundary-guided adversarial examples, the adversarially trained DNN model can achieve a better trade-off between standard accuracy and adversarial robustness. Without loss of generality, other consistency losses (Zhang et al., 2018; Liu & Tan, 2021; Verma et al., 2019b) in semi-supervised learning can also be used here, but they require additional modifications to be adapted for our adversarial training purposes.

Complexity analysis: Compared with the adversarial training based defence methods that generate adversarial examples in the input space, the extra overhead of LADDER mainly lies in the construction of a linear SVM and the training of our generator. The complexity of training a linear SVM is $\mathcal{O}(n^2)$, where $n = 400$ is the number of samples used to train the SVM in our method. The complexity of training our generator is related to the number of layers and number of weights in the generator. After our generator is trained, the generation of adversarial examples is just one forward propagation of the trained generator. For the adversarial training part, our method has the same computational complexity as we use the original adversarial training loss function to adversarially train the model.

4 Experimental evaluation

In this section, we present experimental results to show the effectiveness of our method in achieving a better trade-off between standard accuracy and adversarial robustness. We conduct extensive experiments on MNIST (LeCun & Cortes, 1998), SVHN (Netzer et al., 2011), CelebA (Liu et al., 2015), and CIFAR-10 (Krizhevsky & Hinton, 2009) from four perspectives. The source code of our implementations is provided[1].

[1] https://github.com/zhouxiaowei1120/LADDER

P1: Blessings of Adversarial Examples To show the merits of our latent boundary-guided adversarial examples, we visualize and analyse the generated adversarial examples. (Sect. 4.2).
P2: Standard Accuracy and Adversarial Robustness We evaluate the standard accuracy and adversarial robustness of different adversarially trained models and demonstrate the competitiveness of our LADDER method (Sect. 4.3). In model sharing scenarios, we focus on adversarial robustness against black-box attacks. Detailed experiments on adversarial robustness against white-box attacks can be found in Appendix 2).
P3: Effect of Perturbation We investigate how perturbation impacts the performance of our LADDER method. (Sect. 4.4)
P4: Complement to Regularization-based Adversarial Training Methods We verify the complement effect of LADDER to the existing regularization-based adversarial training methods to achieve a better trade-off between standard accuracy and adversarial robustness. (Sect. 4.5)

4.1 Experiments settings

4.1.1 Datasets and shared DNN models

We conduct our experiments on four datasets: MNIST (one grey digits dataset), SVHN (one colorful digits dataset), CelebA (one human face image dataset), and CIFAR-10 (one natual image dataset). On the four datasets, we use DNN models with different architectures and depths, LeNet (LeCun et al., 1995), SVHNNet (shallow VGG model), CelebANet [deep VGG model (Simonyan & Zisserman, 2014)], and CifarNet (ResNet18) as the targeted classifiers for defence in model sharing scenarios, respectively. Note that, on CelebA, because the size of original images is 178×218, we first pre-process the images to 128×128 using DLIB (Dlib, 2019). We detect faces in images and crop them into square sizes. Our task is the classification of smile or non-smile for an input image.

4.1.2 Baseline methods

The competing methods used for comparison are summarized as follows. FGSM (Goodfellow et al., 2015), JSMA (Papernot et al., 2016b), PGD (Madry et al., 2018), CW (Carlini & Wagner, 2017) and AutoAttack (Croce & Hein, 2020b) are five baselines that generate adversarial examples by adding perturbations in the input space. (Song et al., 2018) is one baseline method that generates adversarial examples in the latent space. TRADES (Zhang et al., 2019b) is one representative method that adds regularisation into the adversarial training loss to improve the trade-off between standard accuracy and adversarial robustness. It uses adversarial examples generated by FGSM for adversarial training. We also compare with another baseline called TRADES+LADDER that combines TRADES with LADDER. This baseline is used to assess whether methods that regularize the adversarial training loss can be complemented when using adversarial examples generated by our LADDER method.

For ablation study, we compare with two variants of our LADDER method: LADDER_cavRandom and LADDER_Random, which use different strategies for generating adversarial examples $\hat{\mathbf{x}}_i$. LADDER_cavRandom adds some random noise $\boldsymbol{\delta}$ on the normal of decision boundary obtained from the SVM: $\hat{\mathbf{x}}_i = \hat{G}\big(\mathbf{z}_i + \epsilon\boldsymbol{\beta}(\mathbf{d} + \boldsymbol{\delta})\big)$. LADDER_Random uses a random noise $\boldsymbol{\gamma}$ to replace the normal of decision boundary for generation:

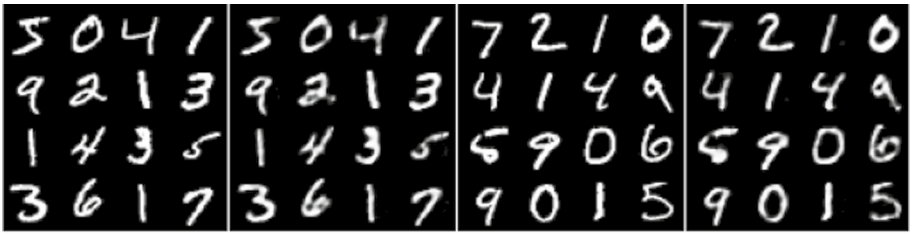

Fig. 4 Reconstructed images of our generator trained on MNIST. **a** and **c** indicate the original training and test images, whereas **b** and **d** show the generated training and test images

$\hat{\mathbf{x}}_i = \hat{G}(\mathbf{z}_i + \epsilon\boldsymbol{\beta}\boldsymbol{\gamma})$. The two baselines are used to show LADDER's effectiveness in using the normal of decision boundary to guide the generation of adversarial examples.

For FGSM, PGD, JSMA and CW, we generate adversarial examples using the open-source attack library *cleverhans* (Papernot et al., 2016a). For the method of (Song et al., 2018), AutoAttack (Croce & Hein, 2020b) and TRADES (Zhang et al., 2019b), we use the source code released by the authors. The number of generated adversarial examples for adversarial training on each dataset is: 4,500 on MNIST; 4,500 on SVHN; 2,000 on CelebA; and 50,000 on CIFAR-10. The hyper-parameters used for all methods in adversarial training are summarized in Table 6 in Appendix 1.

4.2 Blessings of adversarial examples

4.2.1 Fidelity of generator

We first validate the performance of our trained generator in terms of the quality of the generated samples. Here, MNIST is used as a case study to visualize and analyze the results. We train a generator on MNIST using latent features with a dimension of 500, the input to the last layer of LeNet. All components of the generator architecture except for activation functions are provided in Table 10 in Appendix 4. After the last convolutional layer, a sigmoid activation function is added and the loss function used is mean squared error (MSE): $\ell(\mathbf{x}, \hat{\mathbf{x}}) = \frac{1}{n}\sum_{i=1}^{n}\left(\mathbf{x}_i - \hat{\mathbf{x}}_i\right)^2$.

We evaluate our generator through both quantitative and qualitative results. The training loss on the training dataset after 1000 epoches and the test loss over test dataset are 0.00757 and 0.00765, respectively. Figure 4 shows examples of reconstructed images using our generator trained on MNIST, where (a) and (c) are the original training and test images, while (b) and (d) are the generated training and test images. The generated images are very similar to the original ones. This indicates that the trained generator is able to capture the mapping between the latent space and the input space.

To demonstrate the quality of adversarial examples generated by LADDER on natural images, we show adversarial examples of selected classes on SVHN, CIFAR-10 and CelebA in Fig. 5. These examples are generated by perturbing the latent features with different perturbations ϵ. As we can see, these generated images are of high quality without any pepper noise.

(a) SVHN (b) CIFAR-10 (c) CelebA

Fig. 5 Adversarial examples generated by our LADDER method on SVHN, CIFAR-10 and CelebA, where the texts on the left indicate the actual class labels

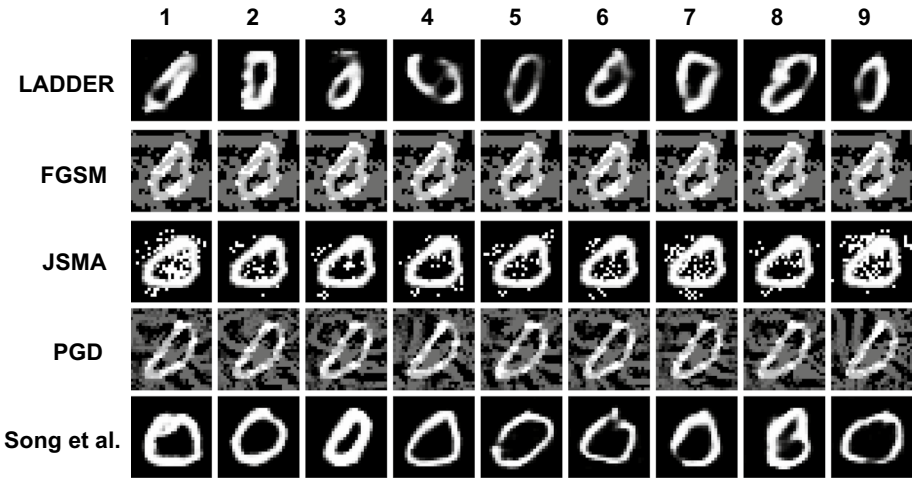

Fig. 6 Adversarial examples generated by FGSM, JSMA, PGD, (Song et al., 2018) and our LADDER method, where the topmost number indicates the predicted class label

4.2.2 Diversity of generated adversarial examples

We compare adversarial examples generated by our LADDER method and other methods (FGSM, JSMA, PGD and (Song et al., 2018)) on MNIST. For LADDER, we used the trained generator to generate adversarial examples against the vanilla LeNet. The latent features that input to the last fully connected layer in LeNet are used to train a linear SVM which yields the normal of boundary for generation. Each extracted latent feature is changed by adding perturbations. Finally, perturbed latent features are fed into the trained generator to generate adversarial examples.

Figure 6 shows example images generated by FGSM, JSMA, PGD, (Song et al., 2018) and our LADDER method. Clearly, LADDER generates a more diverse set of distinct examples, whereas FGSM, JSMA and PGD tend to generate noisy images of repeating patterns. This is because LADDER generates the examples by modifying latent features rather than slightly altering the original images in the input space. As compared with

(Song et al., 2018), adversarial examples generated by LADDER are in general more visually diverse. This diversity property enables LADDER to be more effective for defending against adversarial attacks.

4.2.3 High-quality adversarial examples near boundary

Figure 7 shows adversarial examples generated by our LADDER method in relation to the decision boundary. Compared with adversarial examples generated by FGSM and JSMA (see Fig. 6), LADDER is able to generate non-blurry images of adversarial examples that contain no noise in the background. Such high-quality adversarial examples would not hurt the standard accuracy.

From classification perspectives, samples close to the decision boundary are more likely to be misclassified by a classifier. These samples should be more useful for constructing the classifier to obtain good standard accuracy. LADDER uses the normal of decision boundary as a guide to generate adversarial examples near the boundary. As shown in Fig. 7a, the original sample is apparently a digit 2. When we increase the perturbation (ϵ) added to its corresponding latent features along the normal of the decision boundary, the predicted label of the generated samples changes from 2 to 3. The two examples near the decision boundary, generated with $\epsilon = 7$ and $\epsilon = 9$, are inherently ambiguous, even making humans difficult to make a judgement. If we add these ambiguous adversarial examples with labels to the training set, it would enrich the data space near the decision boundary, thereby improving the generalization of the trained classifier. This is the same case for Fig. 7b. The DNN classifier predicts the two examples, generated with $\epsilon = 9$ and $\epsilon = 10$, as class 1 and class 7, while they look very similar. Such similar adversarial examples would be beneficial to improve the standard accuracy. We provide more illustrative examples in Fig. 10 in Appendix 3 to demonstrate the ability of our generator to generate sensible adversarial examples and the effectiveness of perturbing latent features along the normal of the decision boundary.

4.3 Standard accuracy and adversarial robustness

To validate the efficacy of our LADDER method on standard accuracy, i.e., the accuracy on clean test datasets, as well as adversarial robustness, we conduct experiments on MNIST, SVHN, CelebA and CIFAR-10.

The results are reported in Tables 1, 2, 3 and 4, where row 1 indicates the vanilla model, and other rows indicate adversarially trained models; column 2 represents the clean test dataset, and columns 3–8 represent different attack methods that are used to generate adversarial examples for attacking the targeted model. Under our setting, we focus on defence methods based on adversarial training, and each adversarially trained model is trained on adversarial examples generated by different methods under white-box setting. For FGSM and PGD, we set perturbation as 0.3; for CW, we choose l_2 norm distance. For CW and JSMA, the generation of adversarial examples is under targeted attack condition. For other methods, the generation is under untargeted attack condition. For LADDER, the architectures of the generators on four datasets are provided in Appendix 4. To improve the generation performance on natural images, i.e., CelebA and CIFAR-10, we generalize LADDER by replacing the L_p norm loss with an adversarial

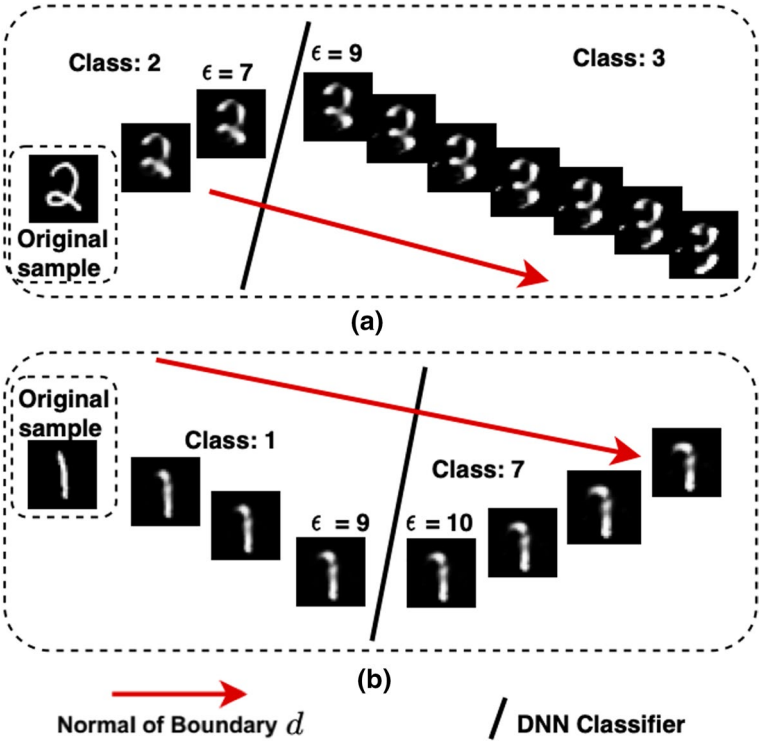

Fig. 7 Generated adversarial examples by our LADDER method are high quality and they are generated near the decision boundary. The number on top of an image is the perturbation (ϵ)

loss used in generative adversarial network (GAN) (Goodfellow et al., 2014) to train a stronger generator. This leads to a variant of our method called LADDER-GAN.

In model sharing scenarios, after one trained model is released, it could be targeted by different attacks, which are unknown to the trained models. Thus, we focus on black-box attacks as indicated in columns 3–8, where adversarial examples are generated with no access to the trained models. In particular, we assess the ability of each adversarially trained model to defend other types of attacks. Thus, the robustness results of each adversarially trained model are not reported against the same attack used to generate adversarial examples.

We focus on assessing the performance of different models in terms of both standard accuracy on clean test dataset and adversarial robustness. We thus calculate the average rank for each adversarially trained model to show its trade-off between standard accuracy and adversarial robustness against several other adversarial attacks. The average rank is calculated over the ranks of each adversarially trained model on clean test dataset and defending all other attacks, which is reported as the last column in the tables.

Table 1 SVHN: Classification accuracy of vanilla and adversarially trained models on clean test dataset and adversarial examples

Defence method	Clean (%)	FGSM (%)	JSMA (%)	PGD (%)	CW (%)	Song et al. (%)	Auto attack (%)	Avg. rank
Vanilla model	93.85	25	34.04	17.16	86.78	99.42	54.62	4.00
FGSM adv.	88.66	–	37.49	20.62	83.07	97.51	66.36	4.83
JSMA adv.	91.04	28.4	–	19.6	86.22	98.69	61.8	*3.83*
PGD adv.	87.75	34.2	42.18	–	85.96	96.69	73.18	4.17
CW adv.	91.11	23.6	37.64	18.11	–	98.16	63.42	4.33
Song et al. adv.	93.53	28	33.91	17.18	87.29	–	56.27	4.00
LADDER_ cavRandom	91.55	24.8	36.96	14.78	84.93	98.72	50.78	5.86
LADDER_Random	90.12	21	35.69	16.42	83.96	98.33	53.87	6.86
LADDER	91.71	26.8	37.29	16.82	86.42	98.96	62	**3.71**

Smaller means better for the average rank (Avg. Rank). The best method is highlighted in bold and the second best is italic

Table 2 MNIST: Classification accuracy of vanilla and adversarially trained LeNet on clean and adversarial examples

Defence method	Clean (%)	FGSM (%)	JSMA (%)	PGD (%)	CW (%)	Song et al. (%)	Auto attack (%)	Avg. rank
Vanilla model	99.13	46.6	93.91	29.93	99.09	99.82	99.56	3.57
FGSM adv.	92.31	–	83.67	80.91	90.58	95.53	92.22	6.50
JSMA adv.	98.56	57.8	–	51.04	98.56	99.87	98.69	4.17
PGD adv.	90.79	76.2	83.31	–	90.67	94.44	90.36	7.00
CW adv.	98.87	59.2	94.67	44.62	–	99.91	99.36	**3.17**
Song et al. adv.	97.23	55.6	89.16	49.91	96.87	–	96.69	6.00
LADDER_ cavRandom	99.01	54.8	92.76	48.2	98.56	99.78	98.89	5.00
LADDER_Random	98.99	64.4	92.93	56.87	98.33	99.89	99.04	*3.29*
LADDER	99.12	55.8	93.13	49.2	98.9	99.82	99.36	*3.29*

Smaller means better for the average rank (Avg. Rank). The best method is highlighted in bold and the second best is italic

4.3.1 Results on SVHN

Table 1 reports the standard accuracy on clean SVHN test dataset and adversarial robustness of different models against other adversarial attacks. We can see that, among all the adversarially trained models, LADDER achieves the second best standard accuracy

Table 3 CelebA: Classification accuracy of vanilla and adversarially trained CelebANet on clean examples and adversarial examples

Defence method	Clean (%)	FGSM (%)	JSMA (%)	PGD (%)	CW (%)	Song et al. (%)	Auto attack (%)	Avg. rank
Vanilla model	91.4	52.65	83.05	13.9	62.1	92.05	49.30	5.43
FGSM adv.	89.4	–	67.15	18.55	62.95	54.95	53.95	7.67
JSMA adv.	90.45	53.1	–	14.95	65.9	93.5	40.35	5.50
PGD adv.	89.55	51.75	63.65	–	67.05	59.55	55.95	6.00
CW adv.	89.5	50.1	77.1	43.25	–	77.8	68.35	5.67
Song et al. adv.	91.15	53.2	83.9	20	66.15	–	49.95	*3.50*
LADDER_ cavRandom	91.1	52.8	81.9	24.9	64.95	86.8	42.55	5.43
LADDER_Random	91.05	55.1	80.15	24.65	64.45	87	55.1	4.71
LADDER-GAN	91.45	52.75	80.9	25.25	64.7	88	46.7	4.71
LADDER	91.95	53.25	82.4	27.6	65.1	87.1	48.95	**3.29**

Smaller means better for average rank (Avg. Rank). The best method is highlighted in bold and the second best is italic

Table 4 CIFAR-10: Classification accuracy of vanilla and adversarially trained models on clean and adversarial examples

Defence method	Clean (%)	FGSM (%)	JSMA (%)	PGD (%)	CW (%)	Song et al. (%)	Auto attack (%)	Avg. rank
Vanilla model	88.99	58.02	80.81	56.87	59.99	28.01	42.31	4.57
FGSM adv.	67.21	–	63.4	63.85	68.56	24.1	63.08	4.83
JSMA adv.	87.87	70.72	–	73.54	78.76	29.61	67.63	**2.17**
PGD adv.	79.12	70.71	75.7	–	72.15	23.92	69.37	4.17
CW adv.	84.53	76.9	81.3	79.46	–	28.15	77.55	**2.17**
Song et al. adv.	48.66	9.99	45.7	11.01	8.87	–	10.75	7.33
LADDER-GAN	85.04	59.71	75.84	58.83	60.39	31.27	45.68	*3.86*
LADDER	85.92	58.26	75.84	60.18	53.92	29.85	47.75	4.00

Smaller means better for the average rank (Avg. Rank). The best method is highlighted in bold and the second best is italic

(91.71%) on clean test dataset, which lags behind the Song et al. Adv. model only. Compared with other models, LADDER achieves an improvement of 3.96% and 3.05% over PGD and FGSM, respectively. As compared with the other two variants, LADDER_ cavRandom and LADDER_Random, LADDER performs better on clean test dataset.

We also find that, LADDER achieves the best performance in terms of defending the (Song et al., 2018) attack, compared with all other adversarially trained models. As for

the overall performance on defending all attacks and on clean test dataset, LADDER achieves an average rank of 3.71, outperforming all other methods. This shows that LADDER achieves a better trade-off between standard accuracy and adversarial robustness. Compared with two variants LADDER_cavRandom and LADDER_Random, LADDER improves the average rank by 2.15 and 3.15, respectively. This validates the necessity of using the normal of decision boundary as guidance to generate adversarial examples.

4.3.2 Results on MNIST

Table 2 reports the classification results of the vanilla and adversarially trained LeNet models on the clean MNIST test dataset and adversarial robustness against other attacks. In terms of standard accuracy on clean test dataset, LADDER performs the best and LADDER_cavRandom achieves the second best among all the adversarially trained models. The performance of LADDER (99.12%) is very close to that of the vanilla model (99.13%), with only 0.01% difference. Moreover, LADDER outperforms the baseline PGD Adv. by a large margin of 8.33%. LADDER is also observed to perform better than its two counterparts, LADDER_cavRandom and LADDER_Random, while LADDER_cavRandom performs better than LADDER_Random.

In terms of adversarial robustness, it is clear to observe that LADDER improves the vanilla model in defending FGSM attack and PGD attack by 9.2% and 19.27%, respectively. When defending the JSMA attack, LADDER performs similarly to the vanilla model. Among all attacks, LADDER achieves the best performance of defending the CW attack and AutoAttack, compared with other adversarially trained models. Overall, LADDER and LADDER_Random achieve an average rank of 3.29, which is the highest among all adversarially trained models except for CW Adv. model. Yet, LADDER achieves better performance than CW Adv. on clean test dataset and against PGD attack, and achieves the same performance against AutoAttack. LADDER_cavRandom also outperforms FGSM, PGD and Song et al. Adv. model. This confirms the usefulness of leveraging the latent features to generate adversarial examples.

4.3.3 Results on CelebA

Table 3 reports standard accuracy and adversarial robustness of the vanilla model and different adversarially trained models on CelebA. In terms of standard accuracy on clean test dataset, LADDER yields the highest accuracy, while LADDER-GAN achieves the second best performance. For the two variants of LADDER, LADDER_cavRandom performs better than LADDER_Random, while both variants outperform the FGSM, PGD, JSMA and CW Adv. models. This shows that performing feature perturbations in the latent space is beneficial to achieve better standard accuracy.

As for the adversarial robustness performance against adversarial attacks, LADDER achieves better performance of defending FGSM, JSMA, PGD and (Song et al., 2018) attacks, compared with most of the baseline models. In particular, for the PGD attack,

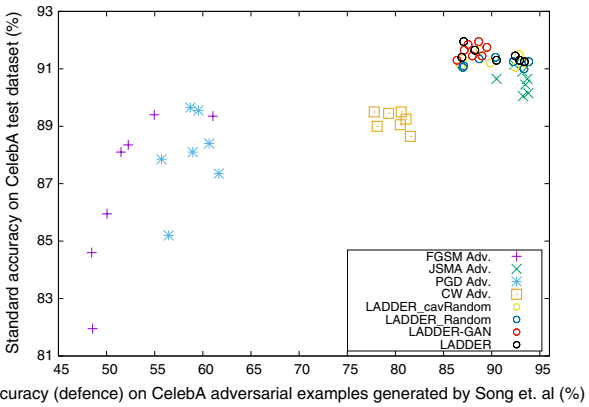

Fig. 8 Classification accuracy of different defence methods on adversarial examples generated by (Song et al., 2018) and on CelebA clean test dataset. Each model is adversarially trained on varying numbers of adversarial examples, with 7 points for each method compared in the figure (Color figure online)

LADDER improves the accuracy from 13.90% to 27.60%. As a whole, LADDER achieves an average rank of 3.29, which is the best among all methods. The smaller the average rank, the better the overall performance of defending adversarial attacks and achieving standard accuracy simultaneously. LADDER-GAN and LADDER_Random both achieve an average rank of 4.71, which stands behind only Song et al. Adv. and LADDER. This proves the overall effectiveness of LADDER and its variants.

4.3.4 Results on CIFAR-10

We also compare standard accuracy and adversarial robustness of LADDER with other baseline methods on CIFAR-10 – a more challenging dataset for the generation task. Table 4 shows the classification results of the vanilla model and different adversarially trained models. We can see that, LADDER is the second best performer among all adversarially trained models, achieving an accuracy of 85.92%. Only the JSMA Adv. model performs slightly better than LADDER with a small gap of 1.95%. Compared to Song et al., FGSM and PGD Adv., LADDER achieves significant improvements by 37.26%, 18.71%, and 6.8%, respectively. The performance of LADDER-GAN slightly lags behind LADDER. This signifies the competitive performance of LADDER in achieving good standard accuracy on CIFAR-10.

For the adversarial robustness, LADDER achieves the best performance when defending the (Song et al., 2018) attack. In general, LADDER achieves a better average rank than FGSM, PGD and Song et al. based adversarially trained models. As the generation task on CIFAR-10 is more challenging, we also compare with LADDER-GAN. As can be seen, LADDER-GAN improves the average rank of LADDER from 4.0 to 3.86. Yet, we find that LADDER and LADDER-GAN perform worse than CW and JSMA adversarially trained models. This indicates that generator-based defence methods have difficulties in achieving the most appealing results on challenging datasets like CIFAR-10. Our findings reaffirm the results of (Song et al., 2018) and those reported in (Jang et al., 2019) where a recursive and stochastic generator is used to generate adversarial examples for adversarial training. We leave further investigation of this problem to future work.

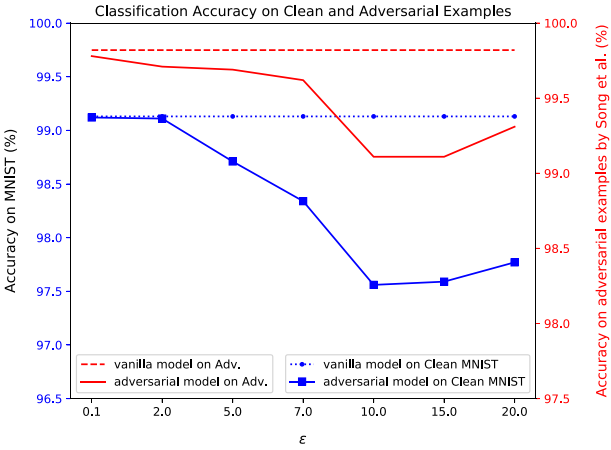

Fig. 9 Classification accuracy of vanilla LeNet and adversarially trained LeNet on MNIST test dataset and adversarial examples with different perturbations ϵ (Color figure online)

4.3.5 Analyse of the trade-off between standard accuracy and robustness

To visually demonstrate the advantages of our LADDER method in achieving a better trade-off between standard accuracy and adversarial robustness, we explicitly compare the trade-off performance of different defence methods with respect to different numbers of adversarial examples on CelebA as a case study. Specifically, we vary the number of examples used to adversarially train the models from 100 to 2,000. The classification results are plotted in Fig. 8, where there are 7 points for each adversarially trained model. In the figure, the x-axis indicates the accuracy of adversarially trained models on adversarial examples generated by (Song et al., 2018), and the y-axis indicates the standard accuracy of adversarially trained models on clean CelebA test dataset. If the trade-off achieved by one method is better, the method is expected to locate in the top right corner. It can be seen clearly that, our LADDER method and its variants (marked in circles) are located in the top right corner. Markedly, our LADDER method outperforms FGSM Adv., PGD Adv., and CW Adv. by a large margin. Again, this confirms that our LADDER method is able to achieve a better trade-off between standard accuracy and adversarial robustness.

4.4 Effect of perturbation

Next, we empirically evaluate the effect of perturbation ϵ on the performance of our LADDER method. First, we study the impact of ϵ on standard accuracy. To adversarially train the LeNet, we randomly select 450 images per class from MNIST dataset to generate 4,500 adversarial examples for each perturbation [0.1, 2.0, 5.0, 7.0, 10.0, 15.0, 20.0]. These adversarial examples with different perturbations are separately used to adversarially train the LeNet. We undertake classification on clean MNIST test dataset using these adversarially trained LeNet models. The results are reported in Fig. 9, colored in blue. It is clear to observe that: (1). As ϵ increases, the classification accuracy of the adversarially trained models decreases firstly and then slightly increases at a later stage. (2). With different ϵ values, the changes in classification accuracy are within an interval of 1.56% only. (3). When ϵ is not too large, i.e. < 7, the performance of the adversarially trained models and the vanilla LeNet is very close.

Table 5 Classification accuracy of vanilla and adversarially trained models on clean test dataset and adversarial examples generated by different attack methods

Dataset	Defence method	Clean (%)	FGSM (%)	JSMA (%)	PGD (%)	CW (%)	Song et al. (%)	Auto attack (%)
MNIST	TRADES	98.44	**60.8**	92.6	**52.31**	98.4	99.8	98.71
	LADDER	**99.12**	55.8	93.13	49.2	**98.9**	**99.82**	99.36
	TRADES+LADDER	98.94	54.8	**93.36**	52.04	98.78	**99.82**	**99.78**
SVHN	TRADES	85.88	13	27.24	10.38	73.93	92.31	44.36
	LADDER	**91.71**	**26.8**	37.29	16.82	**86.42**	**98.96**	62
	TRADES+LADDER	91.01	26.2	**38.02**	**20.11**	84.89	98.36	**64.16**
CelebA	TRADES	**91.95**	52.85	83.1	12.85	63.45	89.2	**53.7**
	LADDER	**91.95**	**53.25**	82.4	**27.6**	**65.1**	87.1	48.95
	TRADES+LADDER	91.7	52.75	**83.3**	18.05	64.55	**93.25**	50
CIFAR-10	TRADES	74.32	73.49	72.39	73.89	73.01	17.2	73.79
	LADDER	**85.92**	58.26	75.84	60.18	53.92	**29.85**	47.75
	TRADES+LADDER	80.7	**75.83**	**76.58**	**77.04**	**76.20**	28.18	**76.78**

Higher means better for classification accuracy. The best results are highlighted in bold

Second, we study the impact of ϵ on adversarial robustness. We still use the adversarially trained LeNet models in the previous step for conducting experiments. These models are used to defend adversarial examples generated using the (Song et al., 2018) attack method. From the red part in Fig. 9, we can see that, as ϵ increases, the performance of the adversarially trained model drops slightly. Overall, with different ϵ values, our LADDER method is able to achieve stable performance within a reasonably small range.

4.5 Complement to regularization-based adversarial training methods

Experiments are further performed to testify whether LADDER can complement the existing regularization-based adversarial training methods that regularize the adversarial loss to achieve a better trade-off between standard accuracy and adversarial robustness. TRADES (Zhang et al., 2019b) is a strong competing method in this category. To achieve the same objective, our LADDER method takes a complementary approach to generate better adversarial examples but to use the original adversarial training loss. We expect that the performance of TRADES could be improved in combination with LADDER.

We perform experiments on MNIST, SVHN, CelebA and CIFAR-10 to compare LADDER, TRADES and the combined TRADES+LADDER. The results are shown in Table 5. As we can see, LADDER achieves better defence performance than TRADES in 18 out of 28 cases on the four datasets. Especially on SVHN, LADDER outperforms TRADES against all attacks and on clean test dataset. As expected, TRADES+LADDER is found to outperform TRADES in most cases (23 out of 28) on the four datasets. This proves that, by generating high-quality and diverse adversarial examples, LADDER can complement regularization-based methods that modify the adversarial training loss function to further improve the performance.

5 Conclusion and future work

We proposed a novel adversarial training framework called *Latent Boundary-guided Adversarial Training* (LADDER), which adversarially trains DNN models through adversarial examples generated based on decision boundary in the latent space. We analyzed that, LADDER can generate high-quality and diverse adversarial examples. After adversarial training on the generated adversarial examples, LADDER achieves a better trade-off between standard accuracy and adversarial robustness. The effectiveness of our LADDER method was validated through extensive experiments on MNIST, SVHN, CelebA, and CIFAR-10. From the new angle of improving the generation of adversarial examples, we showed that our method is also able to complement the existing regularization-based adversarial training methods.

In the future, we will extend our work from the following aspects. Firstly, our method generates adversarial examples by perturbing along the normal of decision boundary to reduce the level of minimal perturbations in the latent space. For inverting to the input space, we will try to derive theoretical bounds about when the perturbations of our generated examples are narrower than the L_p norm perturbations in the input space. Secondly, for complex datasets like CIFAR-10 and ImageNet, where the generation task is more challenging, we have made attempts to use an adversarial loss rather than the L_p norm loss for training a strong generator. We will investigate how to generate better adversarial examples to boost the adversarial robustness on complex datasets. Finally, we would like to reduce the computational complexity of our proposed method by removing the generator and directly using the adversarial features vectors in the latent space for adversarial training.

Appendix

Hyper-parameters in experiments

The hyper-parameters used for adversarial training in our experimental part are summarized in the Table 6.

Table 6 Hyper-parameters of adversarial training for all methods

Dataset	Learning rate	Epochs	Batch Size	Optimizer (momentum; weight decay)
MNIST	0.01	100	64	SGD (0.5; 0)
SVHN	0.001	100	128	SGD (0.9; 5e-4)
CelebA	0.01	100	64	SGD (0.5; 5e-4)
CIFAR-10	0.001	100	128	SGD (0.9; 5e-4)

Robustness under white-box attacks

Robustness on defending white-box attacks

To verify the robustness of our method in defending white-box attacks, we have conducted experiments under white-box settings, where attack methods generate adversarial examples with gradients available from the network. The comparison of our method and other baseline methods on MNIST are shown in Table 7. As can be seen, our methods (LADDER and LADDER_Random) are the best two performers for defending different types of attacks simultaneously. Especially, our LADDER method achieves the best defence performance, when defending CW attacks. Compared with the vanilla model, LADDER improves the performance against all attacks except for CW and (Song et al., 2018). Overall, our proposed method exhibits competitive performance in defending white-box attacks.

LADDER's robustness against LADDER attacks

We have also conducted experiments to compare the defence performance of the vanilla model and our trained model against white-box adversarial examples generated using LADDER. The results are reported in Table 8. As can be clearly seen, our trained model (LADDER) significantly improves the defence performance of the vanilla model on the three datasets by 28.54%, 31.56, and 5.95%, respectively. This proves the efficacy of our

Table 7 Defending white-box attacks targeted on LeNet: classification accuracy of the vanilla LeNet and adversarially trained LeNet models on white-box adversarial examples generated by different attack methods

Defence method	FGSM (%)	JSMA (%)	PGD (%)	CW (%)	Song et al. (%)	Avg rank
Vanilla model	1.80	90.72	0.69	96.88	98.28	–
FGSM adv.	–	82.04	76.28	86.88	90.76	5
JSMA adv.	11.24	–	2.08	95.54	98.65	4.25
PGD adv.	81.00	80.39	–	86.71	92.40	5.25
CW adv.	6.29	92.17	1.41	–	99.38	4
Song et al. adv.	28.74	85.63	17.13	93.83	–	3.25
LADDER_cavRandom	15.31	90.31	4.60	93.47	94.19	4.8
LADDER_Random	25.80	91.54	8.52	95.86	98.51	*3*
LADDER	19.6	92.07	9.02	96.70	98.23	**2.8**

Smaller means better for the average rank (Avg. Rank). The best method is in bold and the second best is italic

Table 8 Classification accuracy on white-box adversarial examples generated by LADDER

Dataset	Vanilla model (%)	LADDER (%)
MNIST	69.78	**98.23**
SVHN	42.48	**74.04**
CelebA	83.85	**89.8**

Higher means better for classification accuracy. The best results are highlighted in bold

trained models against the adversarial examples generated under white-box settings using the same latent based attack method.

Susceptibility of Baseline methods against LADDER attacks

In Table 9, rows 3–7 show the defence performance of five conventional adversarially trained models against the adversarial examples generated by LADDER on three datasets. As compared to the vanilla model, the best performer among the five conventional adversarially trained models improves the defence performance by only 3.55% on MNIST. All five conventional adversarially trained models exhibit worse defence performance than the vanilla model on SVHN and CelebA. This confirms the susceptibility of the conventional adversarially trained methods to the adversarial examples generated using LADDER. In contrast, the adversarially trained LADDER model is very successful in defending against adversarial samples generated using LADDER.

Table 9 Defence performance of the conventional adversarially trained models and our LADDER method against adversarial samples generated using LADDER

Defence method	MNIST (%)	SVHN (%)	CelebA (%)
Vanilla model	69.78	42.48	83.85
FGSM adv.	66.81	40.01	73.00
JSMA adv.	72.16	39.68	83.30
PGD adv.	67.65	39.12	75.35
CW adv.	73.33	42.23	80.95
Song et al. adv.	72.55	42.16	83.15
LADDER	**98.23**	**74.04**	**89.8**

Higher means better for classification accuracy. The best method is highlighted in bold

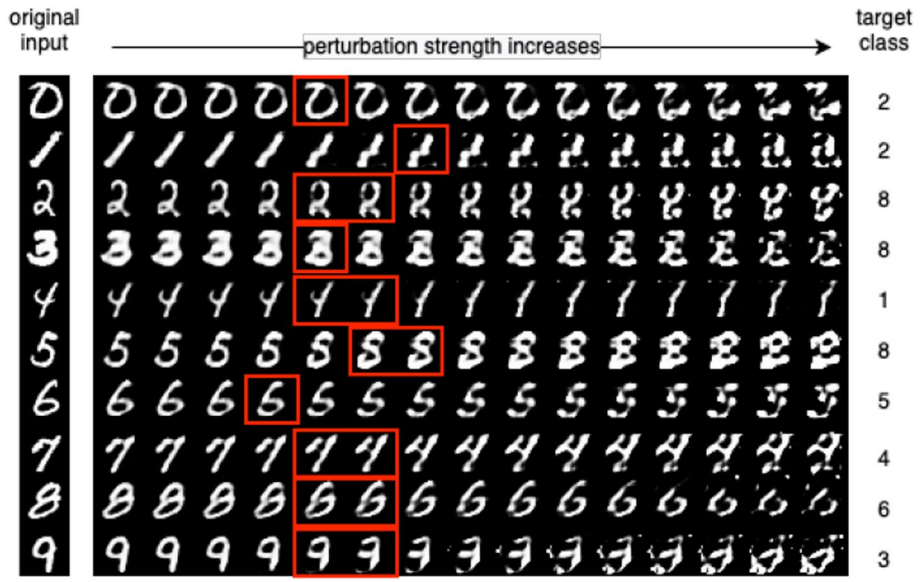

Fig. 10 Illustrative examples generated using our LADDER method with an increasing perturbation (ϵ)

Illustrative examples generated by LADDER

We further provide more illustrative examples to demonstrate the ability of our generator to generate sensible adversarial examples and the effectiveness of perturbing latent features along the normal of the decision boundary.

As shown in Fig. 10, the first column is the original input image; the last column is the randomly sampled target class; columns 2–16 are the generated examples, which are generated by adding different perturbations (ϵ) to latent features of the original inputs. From column 2 to column 16, the perturbation increases gradually from 0.5 to 30.0 along the normal of decision boundary between the class of original input and target class. We can see from the figure, when we increase the perturbation, the generated examples gradually change from the original class to the target class, and when the perturbation is too large (i.e., the last 3 columns), the generated images are distorted. The images marked with red rectangles are inherently ambiguous between the class of the original input and the target class,

Table 10 The architecture of boundary-guided generator for MNIST

Layers	Layer parameters
Linear	Input: 500, output: $50 \times 4 \times 4$
Conv_Transpose	kernel: 2×2, stride: 4×4
Conv	kernel: 3×3, stride: 1×1
Conv_Transpose	kernel: 2×2, stride: 3×3
Conv	kernel: 4×4, stride: 1×1
Conv	kernel: 5×5, stride: 1×1

Table 11 The architecture of boundary-guided generator for SVHN

Layers	Layer parameters
Linear	Input: 4096, output: $512 \times 2 \times 2$
Conv & BN & ReLU	kernels: 512, kernel: 3×3, stride: 1
Conv_Transpose & BN	kernels: 512, kernel: 2×2, stride: 1
Conv & BN & ReLU	kernels: 512, kernel: 3×3, stride: 1
Conv_Transpose & BN	kernels: 512, kernel: 2×2, stride: 1
Conv & BN & ReLU	kernels: 256, kernel: 3×3, stride: 1
Conv_Transpose & BN	kernels: 256, kernel: 2×2, stride: 1
Conv & BN & ReLU	kernels: 128, kernel: 3×3, stride: 1
Conv_Transpose & BN	kernels: 128, kernel: 2×2, stride: 1
Conv & BN & ReLU	kernels: 64, kernel: 3×3, stride: 1
Conv & Tanh	kernels: 3, kernel: 1×1, stride: 1

Table 12 The architecture of boundary-guided generator for CelebA

Layers	Layer parameters	Repeat
Linear	Input: 4096, output: $512 \times 4 \times 4$	1
Conv_Transpose & BN	kernels: 512, kernel: 2, stride: 1	1
Conv & BN & ReLU	kernels: 512, kernel: 3, stride: 1	3
Conv_Transpose & BN	kernels: 512, kernel: 2, stride: 1	1
Conv & BN & ReLU	kernels: 512, kernel: 3, stride: 1	3
Conv_Transpose & BN	kernels: 256, kernel: 2, stride: 1	1
Conv & BN & ReLU	kernels: 256, kernel: 3, stride: 1	3
Conv_Transpose & BN	kernels: 128, kernel: 2, stride: 1	1
Conv & BN & ReLU	kernels: 128, kernel: 3, stride: 1	2
Conv_Transpose & BN	kernels: 64, kernel: 2, stride: 1	1
Conv & BN & ReLU	kernels: 64, kernel: 3, stride: 1	2
Conv & Tanh	kernels: 3, kernel: 1, stride: 1	1

Table 13 The architecture of boundary-guided generator for CIFAR-10

Layers	Layer parameters	Repeat
Linear	input: 512, output: $512 \times 4 \times 4$	1
Conv & BN & ReLU	kernels: 512, kernel: 3, stride: 1	1
Conv_Transpose & BN	kernels: 512, kernel: 2, stride: 2	1
Conv & BN & ReLU	kernels: 512, kernel: 3, stride: 1	4
Conv_Transpose & BN	kernels: 512, kernel: 2, stride: 2	1
Conv & BN & ReLU	kernels: 256, kernel: 3, stride: 1	4
Conv_Transpose & BN	kernels: 256, kernel: 2, stride: 2	1
Conv & BN & ReLU	kernels: 128, kernel: 3, stride: 1	4
Conv & BN & ReLU	kernels: 64, kernel: 3, stride: 1	2
Conv & BN & ReLU	kernels: 32, kernel: 3, stride: 1	2
Conv & BN & ReLU	kernels: 8, kernel: 3, stride: 1	2
Conv & Sigmoid	kernels: 3, kernel: 1, stride: 1	1

even making humans difficult to make a judgement. These images enrich the data space near the decision boundary, thereby improving the generalization of the trained classifier.

Generator architectures

The neural network architectures of boundary-guided generator for MNIST, SVHN, CelebA and CIFAR-10 are detailed in this part. In each table, *Linear* indicates linear transformation; *Conv_Transpose* denotes transposed convolution; *Conv* represents convolution; *BN* represents batch normalization. *kernels* means number of kernels. *kernel* means the dimension of kernel. *stride* means steps of convolutions. *ReLU* means the ReLU activation function (Tables 10, 11, 12 and 13).

Author contributions XZ carried out the experiments and wrote the manuscript. IWTand JY conceived of the presented idea. JY wrote and proofread the manuscript. All authors discussed the experimental results and commented on the manuscript.

Funding Open Access funding enabled and organized by CAUL and its Member Institutions. Xiaowei Zhou is supported by a Data61 PhD Scholarship from CSIRO. Ivor Tsang is supported by the Center for Frontier AI Research, A*STAR and ARC under grants DP200101328. This work is partially supported by the USYD-Data61 Collaborative Research Project grant.

Availability of data and material All datasets used in this work are publicly available.

Code availability The source code of our work is available at: https://github.com/zhouxiaowei1120/LADDER.

Declarations

Conflict of interest The authors de clare that they have no conflict of intrest.

Ethics approval Not applicable

Consent to participate Not applicable

Consent for publication Not applicable

References

Alikaniotis, D., Yannakoudakis, H., & Rei, M. (2016). Automatic text scoring using neural networks. In *ACL* (Vol. 1, pp. 715–725, Long Papers), Association for Computational Linguistics, https://doi.org/10.18653/v1/P16-1068

Amazon. (2019). Machine learning on aws. Retrieved Feb 22, 2019 from https://aws.amazon.com/machine-learning/.

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., & Chen, J. (2016). Deep speech 2: End-to-end speech recognition in english and mandarin. In *ICML* (pp. 173–182).

Andriushchenko, M., Croce, F., Flammarion, N., & Hein, M. (2020). Square attack: A query-efficient black-box adversarial attack via random search. In: *ECCV* (pp. 484–501). Springer.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992) A training algorithm for optimal margin classifiers. In: *Proceedings of the fifth annual workshop on computational learning theory* (pp. 144–152) ACM.

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

Croce, F., & Hein, M. (2020a). Minimally distorted adversarial examples with a fast adaptive boundary attack. In: *ICML* (pp. 2196–2205). PMLR.

Croce, F., & Hein, M. (2020b). Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In: *ICML* (pp. 2206–2216). PMLR.

Dai, T., Feng, Y., Wu, D., Chen, B., Lu, J., Jiang, Y., & Xia, S. T. (2020). Dipdefend: Deep image prior driven defense against adversarial examples. In: *ACM MM* (pp. 1404–1412).

Ding, G., Sharma, Y., Lui K. Y. C., & Huang, R. (2020). Mma training: Direct input space margin maximization through adversarial training. In: *ICLR*.

Dlib. (2019). Dlib python library. Retrieved May 20, 2019 from http://dlib.net/.

Fedus, W., Goodfellow, I., & Dai, A. M. (2018) Maskgan: Better text generation via filling in the_. In: *ICLR*.

Goodfellow, I. J., Shlens, J., & Szegedy, C. (2015). Explaining and harnessing adversarial examples. In: *ICLR*.
Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. *NIPS, 27,* 2672–2680.
Google. (2019). Cloud vision. Retrieved Feb 22, 2019 from https://cloud.google.com/vision/.
Guo, C., Rana, M., Cisse, M., & van der Maaten, L. (2018). Countering adversarial images using input transformations. In: *ICLR*.
Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In: *CVPR* (pp. 4700–4708).
Jang, Y., Zhao, T., Hong, S., & Lee, H. (2019). Adversarial defense via learning to generate diverse attacks. In: *ICCV* (pp. 2740–2749).
Kannan, H., Kurakin, A., & Goodfellow, I. (2018). Adversarial logit pairing. arXiv preprint arXiv:1803.06373
Krizhevsky, A., & Hinton. G, et al. (2009). Learning multiple layers of features from tiny images. Technical Report.
Kurakin, A., Goodfellow, I., & Bengio, S. (2016). Adversarial machine learning at scale. In: *ICLR*.
LeCun, Y., & Cortes, C. (1998). The mnist database of handwritten digits. http://yann.lecun.com/exdb/mnist/.
LeCun, Y., Jackel, L., Bottou, L., Cortes, C., Denker, J. S., Drucker, H., et al. (1995). Learning algorithms for classification: A comparison on handwritten digit recognition. *Neural Networks: The Statistical Mechanics Perspective, 261,* 276.
Lee, S., Lee, H., & Yoon, S. (2020). Adversarial vertex mixup: Toward better adversarially robust generalization. In: *CVPR* (pp. 272–281).
Li, Y., Ding, L., & Gao, X. (2018). On the decision boundary of deep neural networks. arXiv preprint arXiv:1808.05385
Li, Y., Li, L., Wang, L., Zhang, T., & Gong, B. (2019). Nattack: Learning the distributions of adversarial examples for an improved black-box attack on deep neural networks. In: *ICML* (pp. 3866–3876). PMLR.
Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. In: *ICCV*.
Liu, L., & Tan, R. T. (2021). Certainty driven consistency loss on multi-teacher networks for semi-supervised learning. *Pattern Recognition, 120,* 108140.
Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. In: *ICLR*.
Meng, D., Chen, H. (2017). Magnet: a two-pronged defense against adversarial examples. In: *The 2017 ACM SIGSAC CCS* (pp. 135–147). ACM.
Moosavi-Dezfooli, SM., Fawzi, A., & Frossard, P. (2016). Deepfool: A simple and accurate method to fool deep neural networks. In: *CVPR* (pp. 2574–2582).
Mustafa, A., Khan, S., Hayat, M., Goecke, R., Shen, J., & Shao, L. (2019). Adversarial defense by restricting the hidden space of deep neural networks. In: *ICCV* (pp. 3385–3394).
Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., & Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning. In: *NIPS*.
Odena, A., Olah, C., & Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. In: *ICML* (pp. 2642–2651). JMLR. org.
Papernot, N., Faghri, F., Carlini, N., Goodfellow, I., Feinman, R., Kurakin, A., Xie, C,. Sharma, Y., Brown, T., & Roy, A., et al. (2016a). Technical report on the cleverhans v2. 1.0 adversarial examples library. arXiv preprint arXiv:1610.00768
Papernot, N., McDaniel, P., Goodfellow, I., Jha, S., Celik, Z. B., & Swami, A. (2017). Practical black-box attacks against machine learning. In: *ACM ASIACCS* (pp. 506–519).
Papernot, N., McDaniel, P., Jha, S., Fredrikson, M., Celik, ZB., & Swami, A. (2016b). The limitations of deep learning in adversarial settings. In: *2016 IEEE European symposium on security and privacy* (EuroS &P) (pp. 372–387). IEEE.
Papernot, N., McDaniel, P., Wu, X., Jha, S., & Swami A (2016c) Distillation as a defense to adversarial perturbations against deep neural networks. In *2016 IEEE symposium on security and privacy (SP)*, (pp. 582–597). IEEE.
Papernot, N., & McDaniel, P. (2017). Extending defensive distillation. arXiv preprint arXiv:1705.05264
Rahmati, A., Moosavi-Dezfooli, S. M., Frossard, P., & Dai, H. (2020). Geoda: a geometric framework for black-box adversarial attacks. In: *CVPR* (pp. 8446–8455).
Samangouei, P., Kabkab, M., & Chellappa, R. (2018). Defense-gan: Protecting classifiers against adversarial attacks using generative models. In *ICLR*.

Seide, F., Li, G., & Yu, D. (2011). Conversational speech transcription using context-dependent deep neural networks. In: *Twelfth annual conference of the international speech communication association*.
Shaham, U., Yamada, Y., & Negahban, S. (2018). Understanding adversarial training: Increasing local stability of supervised models through robust optimization. *Neurocomputing, 307,* 195–204.
Shi, Y., Zhou, X., Liu, P., & Tsang, I. (2021). Generative transition mechanism to image-to-image translation via encoded transformation. arXiv preprint arXiv:2103.05193
Shrivastava, A., Pfister, T., Tuzel, O., Susskind, J., Wang, W., & Webb, R. (2017) Learning from simulated and unsupervised images through adversarial training. In: *CVPR* (pp. 2107–2116).
Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556
Song, Y., Shu, R., Kushman, N., & Ermon, S. (2018). Constructing unrestricted adversarial examples with generative models. In *NeurIPS* (pp. 8312–8323).
Su, J., Vargas, D. V., & Sakurai, K. (2019). One pixel attack for fooling deep neural networks. *IEEE Transactions on Evolutionary Computation, 23*(5), 828–841.
Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow. I., & Fergus, R. (2013) Intriguing properties of neural networks. arXiv preprint arXiv:1312.6199
Tsipras, D., Santurkar, S., Engstrom, L., Turner, A., & Madry, A. (2019). Robustness may be at odds with accuracy. In: *ICLR*.
Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D., & Bengio, Y. (2019a). Manifold mixup: Better representations by interpolating hidden states. In: *ICML* (pp. 6438–6447). PMLR.
Verma, V., Lamb, A., Kannala, J., Bengio, Y., & Lopez-Paz, D. (2019b). Interpolation consistency training for semi-supervised learning. In: *IJCAI* (pp. 3635–3641).
Wang, H., Li, G., Liu, X., & Lin, L. (2020). A hamiltonian monte carlo method for probabilistic adversarial attack and learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., & Gu, Q. (2020). Improving adversarial robustness requires revisiting misclassified examples. In: *ICLR*.
Xiao, C., Zhong, P., & Zheng, C. (2020). Enhancing adversarial defense by k-winners-take-all. In: *ICLR*.
Xie, C., Zhang, Z., Zhou, Y., Bai, S., Wang, J., Ren, Z., & Yuille, A. L. (2019). Improving transferability of adversarial examples with input diversity. In: *CVPR* (pp. 2730–2739).
Yuan, X., He, P., Zhu, Q., & Li, X. (2019). Adversarial examples: Attacks and defenses for deep learning. IEEE transactions on neural networks and learning systems.
Zhang, H., Cisse, M., Dauphin, Y. N., & Lopez-Paz, D. (2018). Mixup: Beyond empirical risk minimization. In: *ICLR*.
Zhang, H., Goodfellow, I., Metaxas, D., & Odena, A. (2019a). Self-attention generative adversarial networks. In: *ICML, PMLR* (pp. 7354–7363).
Zhang, H., Yu, Y., Jiao, J., Xing, E., El Ghaoui, L., & Jordan, M. (2019b). Theoretically principled trade-off between robustness and accuracy. In: *ICML* (pp. 7472–7482).
Zhang, J., Zhu, J., Niu, G., Han, B., Sugiyama, M., & Kankanhalli M. S. (2021). Geometry-aware instance-reweighted adversarial training. In: *ICLR*.
Zheng, T., Chen, C., & Ren, K. (2019). Distributionally adversarial attack. AAAI*, 33,* 2253–2260.