LEARNING CONTEXTUAL PERTURBATION BUDGETS FOR TRAINING ROBUST NEURAL NETWORKS

Anonymous authors
Paper under double-blind review

ABSTRACT

Existing methods for training robust neural networks generally aim to make models uniformly robust on all input dimensions. However, different input dimensions are not uniformly important to the prediction. In this paper, we propose a novel framework to train certifiably robust models with non-uniform perturbation budgets on different input dimensions, in contrast to the popular ℓ_∞ threat model that has uniform perturbation budgets. We incorporate a perturbation budget generator into the existing certified defense framework, and perform certified training with generated perturbation budgets. We also impose a "robustness volume" constraint on generated perturbation budgets to control the robustness intensity, in comparison to the radius of ℓ_∞ ball in previous works. We evaluate our method on MNIST and CIFAR-10 datasets and show that we can achieve lower clean and certified errors under the same robustness volume, compared to methods using uniform perturbation budgets. Further with two artificial datasets constructed from MNIST and CIFAR-10, we also demonstrate that the perturbation budget generator can produce semantically-meaningful budgets, which implies that the generator can capture contextual information and the sensitivity of different features in input images.

1 INTRODUCTION

It has been demonstrated that deep neural networks, although achieving impressive performance on various tasks, are vulnerable to adverarial perturbations (Szegedy et al., 2013). Models with high accuracy on clean and unperturbed data can be fooled to have extremely poor performance when input data are adversarially perturbed. The existence of adversarial perturbations causes concerns in safety-critical applications such as self-driving cars, face recognition and medical diagnosis.

A number of methods have been proposed for training robust neural networks that can resist to adversarial perturbations to some extent. Among them, adversarial training (Goodfellow et al., 2015; Madry et al., 2018) and certified defenses (Wong et al., 2018; Gowal et al., 2018; Zhang et al., 2020) are of the most reliable ones, and both of them are trying to make the network robust to any perturbation within an ℓ_p norm ball. Taking the commonly used ℓ_∞-ball defense as an example, robust training methods aim to make model robust to ϵ perturbation on any pixel, which means the model is *uniformly robust* on all the input dimensions. But is this a valid assumption we should make?

Intuitively, we expect a good model to be more robust to important features and less robust to unimportant features. Moreover, the "importance" of features should be context-dependent. Taking the MNIST hand-written digit classification problem as an example, the digit 9 can be transformed to 4 simply by modifying just a few pixels on its head, so those pixels should be considered more important, and enforcing them to be robust to ϵ perturbation may not be correct. On the other hand, the pixels on the frame of such an input image, however, can be safely modified without changing the ground-truth label of the image. Therefore, a uniform ϵ budget in robust training can greatly hamper the performance of neural networks on certain tasks, and will force network to ignore some important features that are important for classification.

But how to learn a context-dependent perturbation budget in robust training? This is a nontrivial task and has not been done in the literature of robust training. An intuitive way to assign a learnable ϵ vector, corresponding to the perturbation budget for each individual example, will introduce too

many learnable parameters that are proportional to the data size. Further, a new robust loss has to be carefully designed to promote robustness in the non-uniform sense.

In this paper, we propose the first method that can learn context-dependent perturbation budgets in robust training. Our algorithm is based on certified defense algorithms that conduct robust verification at each training step (Mirman et al., 2018; Gowal et al., 2018; Wong et al., 2018; Zhang et al., 2020; Xu et al., 2020). To learn a context-dependent budget without introducing too many learnable parameters, we introduce a perturbation budget generator with an auxiliary neural network, to generate the context-dependent ϵ based on the input image. We then train the classifier with a linear-relaxation-based certified defense algorithm, auto_LiRPA (Xu et al., 2020) generailized from CROWN-IBP (Zhang et al., 2020), to minimize the verified error under a given robustness volume. The gradients of the loss function can be back-propagated to the perturbation budgets, allowing training the classification network and budget generator jointly in robust training. Empirical evaluations on MNIST and CIFAR-10 datasets confirm that our algorithm is able to learn a meaningful perturbation budget for certified defense.

Our contribution can be summarized below:

- We propose a novel algorithm to train robust networks with contextual perturbation budgets rather than uniform ones. We show that it can be incorporated into certified defense methods with linear relaxation-based robustness verification.
- We demonstrate that our method can effectively train both the classifier and the perturbation generator jointly, and models trained in this way outperform those trained with uniform budgets, in terms of the clean error and verified error under the same certified robustness volume.
- We also show that the learned perturbation budgets are semantically meaningful and align well with the importance of different pixels in the input image. We further confirm this with two artificial tasks and datasets constructed from MNIST and CIFAR-10 respectively.

2 Background and Related Work

Since the discovery of adversarial examples (Szegedy et al. (2013), Biggio et al. (2013)), a great number of works has been devoted to improving the robustness of neural networks from both attack and defense perspectives (Moosavi-Dezfooli et al., 2016; Carlini & Wagner, 2017; Papernot et al., 2016; Moosavi-Dezfooli et al., 2017; Gowal et al., 2019). On a K-way classification task, training an adversarially robust neural network f_w with weight w can generally be formulated as solving the following min-max optimization problem:

$$\min_{w} \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\delta\in\Delta} L(f_w(\mathbf{x}+\delta), y), \tag{1}$$

where $\mathcal{D}$ is the data distribution, and Δ is a threat model defining the space of the perturbations, and L is a loss function.

Adversarial training (Goodfellow et al., 2015; Madry et al., 2018) applies adversarial attacks to solve the inner maximization problem and train the neural network on generated adversarial examples, with efficiency advanced in some recent works (Shafahi et al., 2019; Wong et al., 2020). However, robustness improvements from adversarial training do not have provable guarantees.

Some other recent works seek to train networks that have provable robustness, namely certified defense methods. Such methods solves the inner maximization by computing certified upper bounds that provably hold true for any perturbation within the threat model, including abstract interpretation (Singh et al., 2018), interval bound propagation (Gowal et al., 2018; Mirman et al., 2018), randomized smoothing (Cohen et al., 2019; Salman et al., 2019; Zhai et al., 2020), and linear-relaxation-based methods (Wong & Kolter, 2018; Mirman et al., 2018; Wong et al., 2018; Zhang et al., 2020; Xu et al., 2020). However, nearly all existing certified defense methods treat all input features equally in the threat model, such as an ℓ_p-ball threat models, $\Delta = \{\delta : \| \delta \|_p \leq \epsilon\}$, especially the ℓ_∞ threat model commonly used in many previous works. But different input pixels are not uniformly important to the prediction, and thus we propose to learn contextual perturbation budgets for different pixels.

3 PROPOSED METHOD

3.1 PROBLEM SETTING

Variable threat model Unlike many previous works that use an ℓ_∞ threat model with a uniform ϵ on all the input dimensions, we allow different pixels to have different perturbation budgets, but they need to meet some constraints as we will define below. For an n-dimensional input $\mathbf{x}$, when the perturbation budget of pixel $\mathbf{x}_i$ is ϵ_i, the threat model is

$$\Delta(\epsilon_1, \epsilon_2, \cdots, \epsilon_n) = \{\delta : |\delta_i| \leq \epsilon_i, 1 \leq i \leq n\}.$$

And thereby the ℓ_∞ threat model is a special case with $\epsilon_i = \epsilon_0 (\forall 1 \leq i \leq n)$. We define the *robustness volume* of a threat model Δ as:

$$V(\Delta) = \prod_{i=1}^{n} \epsilon_i, \quad \text{i.e., } \log V(\Delta) = \sum_{i=1}^{n} \log \epsilon_i. \tag{2}$$

In principal, we have a *target robustness volume*, $V_0 = \epsilon_0^n$, and f_w is considered to be provably robust under this target robustness volume on instance $(\mathbf{x}, y)$ if and only if:

$$\exists \Delta \in \mathbb{D}, \forall i \neq y, \min_{\delta \in \Delta} ([f_w(\mathbf{x}+\delta)]_y - [f_w(\mathbf{x}+\delta)]_i) > 0. \tag{3}$$

This means that the predicted score of the ground-truth class y is certifiably larger than any other class $i \neq y$ under some threat model Δ. And instead of using a fixed Δ, in the proposed framework Δ can be taken from a threat model space $\mathbb{D}$, which consists of threat models $\Delta(\epsilon_1, \epsilon_2, \cdots, \epsilon_n)$ satisfying the following two constraints:

$$\text{Volume constraint:} \qquad \sum_i \log \epsilon_i = n \log \epsilon_0, \tag{4}$$

$$\text{Range constraint:} \qquad l \leq \epsilon_i \leq u,\ l = \underline{\alpha}\epsilon_0,\ u = \min(\overline{\alpha}\epsilon_0, 1). \tag{5}$$

The volume constraint states that the robustness volume of Δ is equal to ϵ_0^n (written in $\log$ domain above). For the range constraint, there are relative factors $\underline{\alpha}$ and $\overline{\alpha}$ controlling the perturbation budget range of each pixel, namely $[l, u]$. This constraint is to guarantee a minimum robustness and also prevent the model from over-invariant on each pixel.

Perturbation budget generation A classifier can be accompanied by a perturbation budget generator $\epsilon(\mathbf{x})$ which tries to find perturbation budgets $\epsilon_1, \epsilon_2, \cdots, \epsilon_n$ and the corresponding threat model Δ that can minimize the verified loss of the classifier while satisfying constraints (4) and (5), so that (3) is more likely to hold true with Δ generated by $\epsilon(\mathbf{x})$. We will state the optimization problem to the next paragraph.

Robust classification Accompanied by a perturbation budget generator, we aim to learn a robust classifier f_w with the following min-max optimization problem:

$$\min_w \mathop{\mathbb{E}}_{(\mathbf{x},y)\sim\mathcal{D}} \max_{\delta \in \Delta(\mathbf{x})} L(f_w(\mathbf{x}+\delta), y). \tag{6}$$

Note that a key difference between this problem and the traditional one in (1) is that now the threat model $\Delta(\mathbf{x})$ is variable and dependent on the input $\mathbf{x}$. This $\Delta(\mathbf{x})$ is generated by $\epsilon(\mathbf{x})$. Under the given volume and range constraints, we evaluate the robustness of a classifier f_w by computing an average verified error on all the test instances, where the verified correctness on each instance is evaluated as (3).

3.2 ALGORITHM FRAMEWORK

Figure 1 illustrates our algorithm framework. Given an input $\mathbf{x}$, the perturbation budget generator outputs perturbation budgets $\epsilon(\mathbf{x})$ based on an auxiliary network $g_\theta(\mathbf{x})$ parameterized by θ. Then

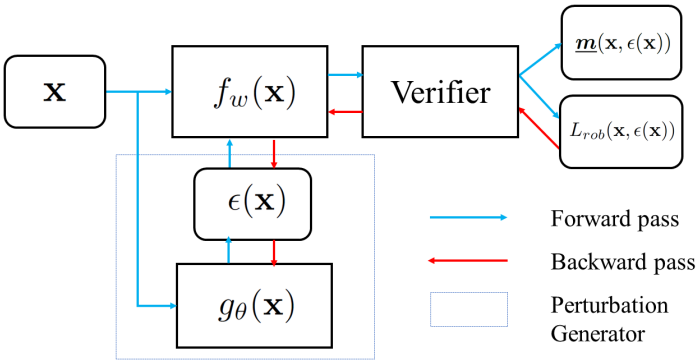

Figure 1: A flowchart of training a certifiably robust network with a perturbation budget generator. The dashed rectangle consists of the perturbation generator, which is based on an auxiliary neural network $g_\theta(\mathbf{x})$. The blue arrows and red arrows represent the forward pass and the back-propagation pass for gradient computation respectively.

a verifier takes the unperturbed input $\mathbf{x}$ and perturbation budgets $\epsilon(\mathbf{x})$ as input and computes a robust loss function $L_{rob}(\mathbf{x}, \epsilon(\mathbf{x}))$. During training, this loss function is back propagated to compute its gradients w.r.t. the parameters of the classifier f_w and also perturbation budgets $\epsilon(\mathbf{x})$, and the gradients w.r.t. $\epsilon(\mathbf{x})$ are further propagated the parameters of $g_\theta(\mathbf{x})$. f_w and g_θ are jointly trained to minimize the robust loss. During test, the verifier also outputs a verified error for evaluation by computing the lower bound of the margin, $\underline{\mathbf{m}}(\mathbf{x}, \epsilon(\mathbf{x}))$ (see Sec. 3.4).

3.3 Perturbation Budget Generator

For the perturbation budget generator $\epsilon(\mathbf{x})$, we generate perturbation budgets in two steps. In the first step, we generate initial perturbation budgets that are only required to meet the range constraint. And in the second step, we refine the perturbation budgets to meet the volume constraint. These two steps are detailed below.

Initial perturbation budgets We first use a neural network $g_\theta(\mathbf{x}) \in [0, 1]^n$ to generate an *initial* budget distribution. Specifically, we assign $[\tilde{\epsilon}(\mathbf{x})]_i = \exp(\log l + [g_\theta(\mathbf{x})]_i(\log u - \log l))$ as the initial budget of the i-th pixel, and thereby $[\tilde{\epsilon}(\mathbf{x})]_i \in [l, u]$. In our work, we employ a two-layer convolutional model with a sigmoid activation as $g_\theta(\mathbf{x})$.

Refining the perturbation budgets We then refine the perturbation budgets to make them meet the volume constraint. We compute the log volume of these budgets, $V = n \log l + (\log u - \log l) \sum_i [\tilde{\epsilon}(\mathbf{x})]_i$, and compare it with the goal log volume $V_0 = n \log \epsilon_0$. If $V < V_0$, we need to allocate $(V_0 - V)$ to the log perturbation budgets of different pixels, while still ensuring that each budget is no larger than u. This allocation is done according to the distance between the current log perturbation budget $\log[\tilde{\epsilon}(\mathbf{x})]_i$ and the upper limit $\log u$ of each pixel, for each pixel i, and then budget $\log[\epsilon(\mathbf{x})]_i$ is obtained with:

$$\log[\epsilon(\mathbf{x})]_i = \log[\tilde{\epsilon}(\mathbf{x})]_i + (n \log \epsilon_0 - V) \frac{\log u - \log[\tilde{\epsilon}(\mathbf{x})]_i}{n \log u - V} \qquad \text{for } V < V_0 = n \log \epsilon_0.$$

If $V > n \log \epsilon_0$, we remove redundant perturbation budgets. Similar to previous case, this refinement is done according to the distance between $\log[\epsilon(\mathbf{x})]_i$ and the lower limit $\log l$:

$$\log[\epsilon(\mathbf{x})]_i = \log[\tilde{\epsilon}(\mathbf{x})]_i - (V - n \log \epsilon_0) \frac{\log[\tilde{\epsilon}(\mathbf{x})]_i - \log l}{V - n \log l} \qquad \text{for } V > V_0 = n \log \epsilon_0.$$

3.4 VERIFIER AND CERTIFIED TRAINING

We perform certified training to minimize (6). For the verifier, we use auto_LiRPA (Xu et al., 2020), which is generalized from CROWN-IBP (Zhang et al., 2020) and also further added with a *loss fusion* technique for more efficient robust training. It is capable of computing a lower or upper bound of any node in a computational graph for a given input and its perturbation space defined by the threat model, when the perturbation can be arbitrarily taken within the threat model. This is achieved by computing and propagating linear bounds and performing linear relaxations on the nonlinear activation functions. The lower or the upper bound of the bounded node can be represented in the form of a linear function w.r.t. the model input, and this linear function can be concretized to real values according to the threat model. We refer readers to Xu et al. (2020) for their detailed algorithm.

In our study, during training, we compute and optimize the robust loss $L_{rob}(\mathbf{x}, \epsilon(\mathbf{x}))$:

$$L_{rob}(\mathbf{x}, \epsilon(\boldsymbol{x})) \geq \max_{\delta \in \Delta(\mathbf{x})} L(f_w(\mathbf{x}+\delta), y), \tag{7}$$

where the robust loss is an upper bound of the right-hand-side, obtained from the verifier. The auto_LiRPA verifier is differentiable and thus we are able to back-propagate the gradients of the loss function w.r.t. the parameters of the both classifier and perturbation budget generator for training. In contrast, this joint training appears to be more difficult for adversarial training methods such as Projected Gradient Descent (PGD) attack (Madry et al., 2018) as it is not fully differentiable w.r.t. the perturbation budgets.

During inference time, in order to compute the verified error besides the loss function, we compute the lower bound of the minimum margin between the ground truth label y and any other label $i \neq y$:

$$\underline{\mathbf{m}}(\mathbf{x}, \epsilon(\mathbf{x})) = \min_{\delta \in \Delta(\mathbf{x})} \boldsymbol{m}(\mathbf{x}+\delta), \text{where } \boldsymbol{m}(\mathbf{x}+\delta) = \min_{i \neq y}([f_w(\mathbf{x}+\delta)]_y - [f_w(\mathbf{x}+\delta)]_i), \tag{8}$$

where $\underline{\mathbf{m}}(\mathbf{x}, \epsilon(\mathbf{x}))$ is the lower bound of the margin for all possible perturbation $\delta \in \Delta$. The classifier is verifiably robust on this instance and the threat model $\Delta(\mathbf{x})$ from the generated perturbation budgets, if $\underline{\mathbf{m}}(\mathbf{x}, \epsilon(\mathbf{x})) > 0$, as defined in (3).

4 EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

We conduct experiments on MNIST and CIFAR-10, and our algorithm is implemented based on auto_LiRPA[1] (Xu et al., 2020), which is generalized from CROWN-IBP (Zhang et al., 2020). The perturbation budget generator is a two-layer convolutional model with ReLU activations after each convolutional layer, and the two convolutional layers have 4 and 8 channels respectively and a kernel size of 3. After that, there is a linear layer with a Sigmoid activation for computing initial perturbation budgets afterwards. For certified defense, we mainly follow settings in CROWN-IBP. We use the DM-large model mentioned in their work for experiments on CIFAR-10, and we use the DM-small model for MNIST since we find the DM-large model has a serious overfitting problem on MNIST. , but we still use DM-large to report the results of models with uniform budgets following the best setting in CROWN-IBP. We use similar training hyperparameters provided in CROWN-IBP, but we use the loss fusion technique proposed in auto_LiRPA for faster training. On MNIST, we set $\underline{\alpha}$ to 0.5 and $\overline{\alpha} = 2.0$. On CIFAR-10, we set $\underline{\alpha}$ to 0.125 and $\overline{\alpha}$ to 2.0. The values of different robustness volumes are reported in later sections.

4.2 EXPERIMENTS ON MNIST AND CIFAR-10

For MNIST, we consider robustness volumes with $\epsilon_0 = 0.4, 0.6, 0.8$ respectively. For models using uniform budgets, we follow the best setting in CROWN-IBP (Zhang et al., 2020). We find that the models totally fail at $\epsilon_0 = 0.6$ and $\epsilon_0 = 0.8$ with clean and verified errors larger than 85%. This appears to be reasonable as we are able to perturb every pixel to 0.5 under uniform perturbation

[1] https://github.com/KaidiXu/auto_LiRPA

Table 1: Results of using uniform perturbation budgets and our learned ones respectively on MNIST with various robustness volume goals. "-" denotes that models with uniform perturbation budgets totally fail on $\epsilon_0 = 0.6$ and $\epsilon_0 = 0.8$. Results of uniform budgets on $\epsilon_0 = 0.4$ are from Zhang et al. (2020).

Volume	Uniform budgets (Zhang et al., 2020)		Learned budgets (Ours)	
	Clean error (%)	Verified error (%)	Clean error (%)	Verified error (%)
$\epsilon_0 = 0.4$	2.17	12.06	**1.41**	**7.81**
$\epsilon_0 = 0.6$	-	-	**2.95**	**13.93**
$\epsilon_0 = 0.8$	-	-	**12.15**	**26.37**

Table 2: Results on CIFAR-10, in a similar format with Table 1. Results of uniform budgets with $\epsilon_0 = 8/255$ and $\epsilon_0 = 16/255$ are from Zhang et al. (2020), and others are obtained by ourselves.

Volume	Uniform budgets (Zhang et al., 2020)		Learned budgets (Ours)	
	Clean error (%)	Verified error (%)	Clean error (%)	Verified error (%)
$\epsilon_0 = 8/255$	54.02	66.94	**49.34**	**60.72**
$\epsilon_0 = 16/255$	66.06	76.80	**55.64**	**63.85**
$\epsilon_0 = 32/255$	80.63	85.05	**63.34**	**69.31**

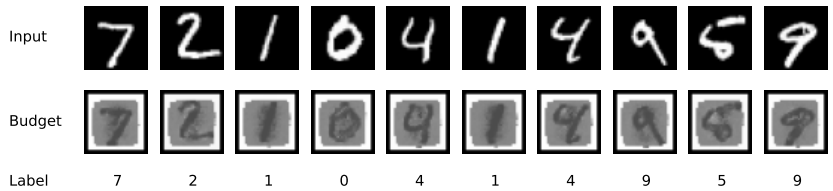

Figure 2: Visualization of learned perturbation budget distributions on MNIST with $\epsilon_0 = 0.6$. The *budget* images have pixels ranging from 0 to 1 representing a per-pixel, per-example ϵ. Lighter colors indicate larger perturbations and darker colors indicate smaller perturbations.

budgets $\epsilon = 0.6$ or $\epsilon = 0.8$. In contrast to using uniform budgets, with perturbation budgets learned with our method, we are able to achieve lower clean and verified error under the same robustness volume $\epsilon_0 = 0.4$. Moreover, we are able to train models with larger robustness volume $\epsilon_0 = 0.6$ and $\epsilon_0 = 0.8$ and achieve acceptable errors, while models using uniform budgets totally fail. On CIFAR-10, we consider robust volumes with $\epsilon_0 = 8/255, 16/255, 32/255$, while CROWN-IBP only considered uniform budgets with ϵ no larger than $16/255$ in test. As shown in Table 2, under the same robustness volume, models with learned budgets have smaller errors compared to those with uniform budgets.

We also visualize our learned perturbation budget distributions in Figure 2 and Figure 3 for MNIST and CIFAR-10 respectively. Our perturbation budget generator is able to identify important and sensitive features in the input image and assign relatively smaller budgets to the corresponding pixels, while insensitive pixels are assigned with relatively larger perturbation budgets. Note that the perturbation budgets also highlight sensitive pixels that are missing from input images. Therefore, learned perturbations contain important features from both the ground truth class and the classes that are close to it. For example, as visualized in Figure 2, perturbation budgets of digit 4 combine features from both 4 and 9, and digit 5 looks like 5 overlapped with 8.

4.3 Experiments on Watermarked MNIST

To demonstrate another benefit of using different perturbation budgets for different pixels, we artificially constructed a dataset from MNIST, namely the Watermarked MNIST. For each image in the original MNIST dataset, as shown in the first row of Figure 4, we add a small watermark to the left or right margin of the image, which decides the gold label jointly with the main digit in the image. The watermark is darker than the main digit, and thus we expect to see the perturbation generator assign smaller perturbation budgets to the watermark, to retain the information carried by the watermark. Specifically, the watermark is a digit of zero or one, chosen randomly, from an image

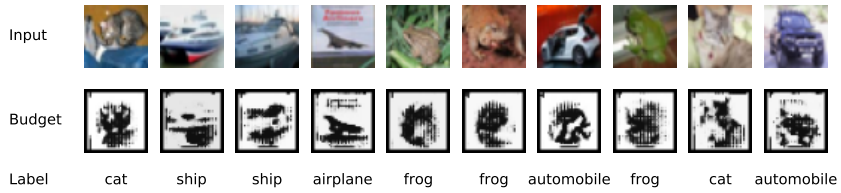

Figure 3: Visualization of learned perturbation budget distributions on CIFAR-10 with $\epsilon_0 = 16/255$. For better visualization, the visualized budgets are normalized by the upper limit of the perturbation budgets, 32/255, since the budgets on CIFAR-10 are much smaller than 1.

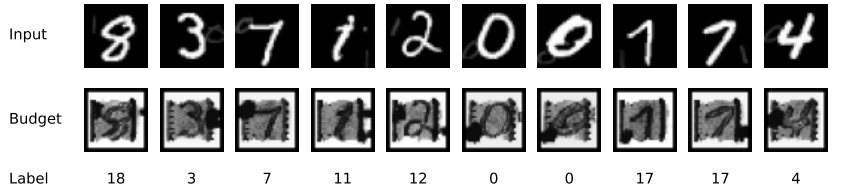

Figure 4: Images in the artificial Watermarked MNIST dataset and the visualization of our learned perturbation budget distributions. The watermarked pixels must use a smaller perturbation to be correctly recognized, and our learned perturbation correctly chooses a smaller perturbation budget for these pixels.

in the original dataset. We scale down the image to 12×12, and we crop out 2 pixels on each border and retain the inner 8×8 part. The opacity of the watermark is reduced to 30%. We randomly put the watermark either on the left or right 28×8 border, and the vertical position span is randomly chosen from $[0, 7], [1, 8], \cdots, [20, 27]$ (vertical position numbered starting from 0). The gold label of the new image is the main digit itself, if the watermark is 0, otherwise the gold label is the main digit *plus* 10. In this way, we construct a 20-class digit classification problem.

We then perform certified defense on this artificial dataset, with $\epsilon_0 = 0.4$. For our learned perturbation budget generator, we set $\underline{\alpha} = 0.125, \overline{\alpha} = 2.5$. We compare the results with the method using uniform perturbation budgets, as shown in Table 3. When uniform perturbation budgets are used, the clean error and the verified error are very large. This is because the values of the pixels with the watermark is around 30% of those in an original digit, and thus when a perturbation budget of $\epsilon \geq 0.3$ is uniformly added to all the pixels, it is enough to perturb the image such that the watermark disappears, and thus the model cannot achieve low errors. In contrast, as shown in Figure 4, by learning contextual perturbation budgets, the budget generator tends to assign relatively small budgets to pixels with the watermark. Pixels with the main digit tend to have budgets that are somewhat larger, and there are empty pixels which receive even larger budgets. With such perturbation budgets, we achieve much lower verified error (19.04 v.s. 59.29 on $\epsilon_0 = 0.4$) and clean error (3.48 v.s. 51.75 on $\epsilon_0 = 0.4$) under the same robustness volume, as shown in Table 3. This experiment demonstrates the importance of learning different perturbation budgets for different pixels and the effectiveness of our proposed method.

Table 3: Results on the artificial Watermarked MNIST dataset. Using an uniform budget resulting in a $> 50\%$ error, as the classifier are not sensitive enough to these watermarked pixels.

Volume	Uniform budgets (Zhang et al., 2020)		Learned budgets (Ours)	
	Clean error (%)	Verified error (%)	Clean error (%)	Verified error (%)
$\epsilon_0 = 0.3$	50.90	55.54	**1.99**	**11.00**
$\epsilon_0 = 0.4$	51.75	59.29	**3.48**	**19.04**

4.4 EXPERIMENTS ON DOUBLED CIFAR-10

To further demonstrate the ability of the perturbation generator in identifying important input features, we create a harder artificial dataset, namely Doubled CIFAR-10, based on CIFAR-10. An original input image in CIFAR-10 has a size of 32×32. In this setting, each new input image consists of two original images on the left and the right respectively. Two images combined in this way yields a new one with a size of 32×64. We also add a 4×64 border to the top of the new image. We make the border either red or green. If the border if red, the gold label is taken as the gold label of the left original image, otherwise it is the gold label of the right original image. The border serves as a hint for the model to tell which original 32×32 image it should look at to make the prediction. The size of each new image is 36×64. And with such a dataset, we expect to further study whether the perturbation budget generator can identify important features in the input image, since now one of the two component image in each new image does not affect the gold label, and thus the model is expected to assign large perturbation budgets on this component image. This is a harder task than the Watermarked MNIST since many peripheral pixels in an MNIST image are black and empty, and the model may rely on this simple feature for generating budgets, while images in CIFAR-10 are much more complex and the unused component image is not empty.

Table 4: Results on the artificial Doubled CIFAR-10 dataset.

Uniform budgets (Zhang et al., 2020)		Learned budgets (Ours)	
Clean error (%)	Verified error (%)	Clean error (%)	Verified error (%)
73.16	81.29	**56.11**	**63.24**

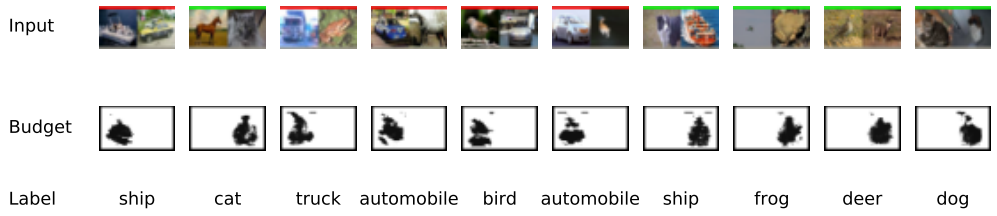

Figure 5: Images in the artificial Doubled CIFAR-10 dataset and the visualization of our learned perturbation budget distributions. The visualized budgets are normalized by $1.5 \times 16/255$ similarly as Figure 3. The generated perturbations correctly correlate with border colors.

We use $\epsilon_0 = 16/255$, $\underline{\alpha} = 0.0625$, $\overline{\alpha} = 1.5$. Results are shown in Table 4. Again, the errors of the model with learned perturbation budgets are much lower than those of the model with uniform budgets. We also visualize the learned perturbation budgets in Figure 5. We observe that when the border is in red, large perturbation budgets are assigned to the right side, which is consistent with that the gold label is determined only by the left side in this case. This observation also holds true for the case when the border is in green. Moreover, we also observe that the perturbation budget distributions of the other side are semantically meaningful. This experiment further demonstrates that our proposed method is able to identify important features in the input and learn semantically meaningful distributions of perturbation budgets.

5 CONCLUSIONS

In this paper, we propose a novel approach to generate contextual perturbation budgets and successfully incorporate it into linear-relaxation-based certified training methods. We show its superiority over methods with uniform perturbation budgets via experiments on MNIST and CIFAR-10. We also demonstrate that the perturbation budget generator can capture semantic information and align well with features in the input image with artificially constructed Watermarked MNIST and Doubled CIFAR10 datasets. Our proposed method can be used in settings where the perturbation budget of a dataset cannot be easily determined as in toy datasets, and can handle context dependent perturbations.

REFERENCES

Battista Biggio, Igino Corona, Davide Maiorca, Blaine Nelson, Nedim Šrndić, Pavel Laskov, Giorgio Giacinto, and Fabio Roli. Evasion attacks against machine learning at test time. In *Joint European conference on machine learning and knowledge discovery in databases*, pp. 387–402. Springer, 2013.

Nicholas Carlini and David Wagner. Adversarial examples are not easily detected: Bypassing ten detection methods. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pp. 3–14. ACM, 2017.

Jeremy M Cohen, Elan Rosenfeld, and J Zico Kolter. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Sven Gowal, Krishnamurthy Dvijotham, Robert Stanforth, Rudy Bunel, Chongli Qin, Jonathan Uesato, Timothy Mann, and Pushmeet Kohli. On the effectiveness of interval bound propagation for training verifiably robust models. *arXiv preprint arXiv:1810.12715*, 2018.

Sven Gowal, Jonathan Uesato, Chongli Qin, Po-Sen Huang, Timothy Mann, and Pushmeet Kohli. An alternative surrogate loss for pgd-based adversarial testing. *arXiv preprint arXiv:1910.09338*, 2019.

Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Matthew Mirman, Timon Gehr, and Martin Vechev. Differentiable abstract interpretation for provably robust neural networks. In *International Conference on Machine Learning*, pp. 3575–3583, 2018.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, and Pascal Frossard. Deepfool: a simple and accurate method to fool deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2574–2582, 2016.

Seyed-Mohsen Moosavi-Dezfooli, Alhussein Fawzi, Omar Fawzi, and Pascal Frossard. Universal adversarial perturbations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1765–1773, 2017.

Nicolas Papernot, Patrick McDaniel, and Ian Goodfellow. Transferability in machine learning: from phenomena to black-box attacks using adversarial samples. *arXiv preprint arXiv:1605.07277*, 2016.

Hadi Salman, Jerry Li, Ilya Razenshteyn, Pengchuan Zhang, Huan Zhang, Sebastien Bubeck, and Greg Yang. Provably robust deep learning via adversarially trained smoothed classifiers. In *Advances in Neural Information Processing Systems*, pp. 11289–11300, 2019.

Ali Shafahi, Mahyar Najibi, Mohammad Amin Ghiasi, Zheng Xu, John Dickerson, Christoph Studer, Larry S Davis, Gavin Taylor, and Tom Goldstein. Adversarial training for free! In *Advances in Neural Information Processing Systems*, pp. 3358–3369, 2019.

Gagandeep Singh, Timon Gehr, Matthew Mirman, Markus Püschel, and Martin Vechev. Fast and effective robustness certification. In *Advances in Neural Information Processing Systems*, pp. 10825–10836, 2018.

Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2013.

Eric Wong and Zico Kolter. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *International Conference on Machine Learning*, pp. 5283–5292, 2018.

Eric Wong, Frank Schmidt, Jan Hendrik Metzen, and J Zico Kolter. Scaling provable adversarial defenses. In *NIPS*, 2018.

Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020.

Kaidi Xu, Zhouxing Shi, Huan Zhang, Yihan Wang, Kai-Wei Chang, Minlie Huang, Bhavya Kailkhura, Xue Lin, and Cho-Jui Hsieh. Provable, scalable and automatic perturbation analysis on general computational graphs, 2020.

Runtian Zhai, Chen Dan, Di He, Huan Zhang, Boqing Gong, Pradeep Ravikumar, Cho-Jui Hsieh, and Liwei Wang. Macer: Attack-free and scalable robust training via maximizing certified radius, 2020.

Huan Zhang, Hongge Chen, Chaowei Xiao, Bo Li, Duane Boning, and Cho-Jui Hsieh. Towards stable and efficient training of verifiably robust neural networks. In *International Conference on Learning Representations*, 2020.