Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Towards robust neural networks via a global and monotonically decreasing robustness training strategy*

Zhen LIANG†1, Taoran WU[2,3], Wanwei LIU†‡[4,5], Bai XUE[2], Wenjing YANG[1], Ji WANG[1], Zhengbin PANG[4]

[1]*Institute for Quantum Information & State Key Laboratory of High Performance Computing, National University of Defense Technology, Changsha 410073, China*

[2]*State Key Laboratory of Computer Science Institute of Software, Chinese Academy of Sciences, Beijing 100190, China*

[3]*School of Computer Science and Technology, University of Chinese Academy of Sciences, Beijing 100190, China*

[4]*College of Computer Science and Technology, National University of Defense Technology, Changsha 410073, China*

[5]*Key Laboratory of Software Engineering for Complex Systems, National University of Defense Technology, Changsha 410073, China*

†E-mail: liangzhen@nudt.edu.cn; wwliu@nudt.edu.cn

Received Feb. 1, 2023; Revision accepted Apr. 26, 2023; Crosschecked Sept. 19, 2023

Abstract: Robustness of deep neural networks (DNNs) has caused great concerns in the academic and industrial communities, especially in safety-critical domains. Instead of verifying whether the robustness property holds or not in certain neural networks, this paper focuses on training robust neural networks with respect to given perturbations. State-of-the-art training methods, interval bound propagation (IBP) and CROWN-IBP, perform well with respect to small perturbations, but their performance declines significantly in large perturbation cases, which is termed "drawdown risk" in this paper. Specifically, drawdown risk refers to the phenomenon that IBP-family training methods cannot provide expected robust neural networks in larger perturbation cases, as in smaller perturbation cases. To alleviate the unexpected drawdown risk, we propose a global and monotonically decreasing robustness training strategy that takes multiple perturbations into account during each training epoch (global robustness training), and the corresponding robustness losses are combined with monotonically decreasing weights (monotonically decreasing robustness training). With experimental demonstrations, our presented strategy maintains performance on small perturbations and the drawdown risk on large perturbations is alleviated to a great extent. It is also noteworthy that our training method achieves higher model accuracy than the original training methods, which means that our presented training strategy gives more balanced consideration to robustness and accuracy.

Key words: Robust neural networks; Training method; Drawdown risk; Global robustness training; Monotonically decreasing robustness
https://doi.org/10.1631/FITEE.2300059 **CLC number:** TP311; TP183

1 Introduction

In recent years, artificial intelligence (AI) and deep learning (DL) have witnessed enormous success in various domains, such as image recognition (Chen and He, 2021; Tian et al., 2021), natural language processing (Devlin et al., 2018), and automatic driving (Bojarski et al., 2016). Behind these great achievements, deep neural networks (DNNs) have

‡ Corresponding author

* Project supported by the National Key R&D Program of China (No. 2022YFA1005101), the National Natural Science Foundation of China (Nos. 61872371, 62032024, and U19A2062), and the CAS Pioneer Hundred Talents Program, China

ORCID: Zhen LIANG, https://orcid.org/0000-0002-1171-7061; Wanwei LIU, https://orcid.org/0000-0002-2315-1704

become dominant computing models and have made significant contributions to the prosperity of AI and DL. Nevertheless, DNN behaviors are far from infallible; e.g., imposing human-imperceptible perturbations on the input samples causes drastic changes in the outputs (Goodfellow et al., 2015). The unexpected outputs would result in loss of life and property (Ma et al., 2018), especially in safety-critical applications.

Consequently, there is a pressing need to guarantee that the behaviors of DNNs obey some given properties before their deployments in practice, like safety (Wang et al., 2018a), robustness (Katz et al., 2017), reachability (Tran et al., 2019), and fairness (Sun et al., 2021). Among them, robustness, which refers to DNNs working stably under minor input disturbances, is one of the most important properties in academic and industrial communities. On one hand, researchers focus on proposing dozens of robustness verification approaches on different kinds of DNNs (Weng et al., 2018b; Zhang H et al., 2018; Balunović et al., 2019; Ko et al., 2019; Liu WW et al., 2020; Du et al., 2021; Guo et al., 2021; Liu JX et al., 2022; Zhang YD et al., 2022; Zhao et al., 2022), with the techniques used ranging from abstract interpretation (Ryou et al., 2021) and mixed integer linear programming (Tjeng et al., 2019) to symbolic execution (Li et al., 2019). On the other hand, besides verifying the robustness of certain trained DNNs, researchers pay attention to training robust neural networks (NNs) against certain perturbations. Compared with providing post facto conclusions indicating whether the robustness holds or not, providing guaranteed robust NNs with proper training methods seems to be more important, in the sense of NN applications and deployments.

There have been extensive studies concentrating on training robust NNs so far. One natural robust training method is data augmentation (Goodfellow et al., 2015), which generates training data by sampling randomly in the perturbation-specified input region. The original augmentation idea is inefficient and is developed by attack algorithms, like the fast gradient sign method (FGSM) (Goodfellow et al., 2015) and projected gradient descent (PGD) method (Madry et al., 2018), to improve the effectiveness and efficiency of sampling. Furthermore, in some recent literature, researchers attempted to train DNNs with verifiable guarantees on robustness performance, such as linear relaxation (Dvijotham et al., 2018; Mirman et al., 2018; Wang et al., 2018b; Zhang H et al., 2020), interval bound propagation (IBP) (Gowal et al., 2018; Mirman et al., 2018), ReLU stability regularization based method (Xiao et al., 2019), and global robustness based method (Leino et al., 2021). Among these methods, IBP (Gowal et al., 2018) is a simple and efficient method for training verifiable robust DNNs. To refine the loose bound estimation of IBP, CROWN-IBP (Zhang H et al., 2020) combines IBP with tighter linear relaxation work (Zhang H et al., 2018) and reaches state-of-the-art performance, easing the instability of the training procedures and sensitivity to hyperparameters simultaneously. Additionally, we refer interested readers to Liang et al. (2023) for a comprehensive review.

In this work, instead of seeking a completely novel robust NN training method, we turn our attention to alleviating the great challenge that existing IBP-family training methods encounter; i.e., IBP and CROWN-IBP methods are likely to fail under large input perturbations. This phenomenon is called "drawdown risk" in this paper. Therefore, IBP-family training methods cannot provide the expected robust NNs in large perturbation cases. After analyzing the training paradigm of IBP and CROWN-IBP, we give a possible explanation for the drawdown risk. It is likely to stem from training robust DNNs toward a single perturbation value each time (training epoch), and thus locality leads to failure, with large perturbations in particular. Therefore, our global robustness strategy takes multiple certain perturbations (sub-perturbations) into account in the training phases, in place of a single perturbation. In addition to training DNNs with those sub-perturbations globally, our remedy for existing IBP-based training methods investigates the monotonically decreasing combination weights of the robustness optimization corresponding to different perturbations, termed the monotonically decreasing robustness training strategy. The so-called monotonically decreasing robustness training strategy requires that the smaller a sub-perturbation is, the more weight it should be assigned during the DNN training phases. Moreover, our proposed global and monotonically decreasing robustness training strategy works for any existing training method and is simple to implement.

The contributions of this paper are as follows:

1. We observe and formalize the drawdown risk problem existing in the IBP and CROWN-IBP training methods, and explore a possible reason that IBP-family methods fail under large perturbation cases, that is, the drawdown risk resulting from local robustness training (LRT). To overcome this hazard, we propose the idea of global robustness training (GRT), considering some certain perturbations together, instead of only a single perturbation value, during the training processes.

2. Taking multiple perturbation values into account, we seek a monotonically decreasing weight strategy for combining them during the training phases and form the new loss function. Under the guidance of the monotonically decreasing weight strategy, the remedy training method is compatible with existing IBP-based training methods and can be treated as a generalization of them.

3. We implement our training strategy on the state-of-the-art work, CROWN-IBP. The experimental results illustrate that our strategy keeps (or improves a little) the performance in existing small perturbation cases and enhances the performance significantly in large perturbation cases, alleviating the drawdown risk during training.

2 Background and related works

An L-layer (classification) DNN $\mathcal{N}$ can be defined recursively as follows:

$$\begin{cases} \mathcal{N}(x) = z^{(L)},\ z^{(l)} = W^l h^{(l-1)} + b^{(l)}, \\ W^l \in \mathbb{R}^{n_l \times n_{l-1}},\ b^{(l)} \in \mathbb{R}^{n_l}, \\ h^{(l)} = \sigma^{(l)}(z^{(l)}),\ l \in \{1, 2, \cdots, L-1\}, \end{cases} \tag{1}$$

where $h^{(0)}(x) = x$ (i.e., the input), and n^l is the number of neurons located in the l^{th} layer, also called the layer dimension. n_0 refers to the input dimension of neural network $\mathcal{N}$ and n^L represents the number of prediction classes, i.e., the output dimension. W^l and b^l are the weight matrix and the bias in the adjacent $(l-1)^{\text{th}}$ and l^{th} layers, respectively. σ is the nonlinear element-wise activation function, and we use z to represent pre-activation neuron values and h for post-activation neuron values.

2.1 Robustness of DNNs

Robustness is one of the most important properties of DNNs, and there are some definitions of NN robustness with minor differences (Casadio et al., 2022), like classification robustness, standard robustness, and Lipchitz robustness. In this work, we focus on the widely used classification based definition, which implies that all samples within a given range have the same prediction class:

Definition 1 (Classification robustness) A DNN $\mathcal{N}$ is ϵ-robust with respect to input example x with ground truth label y, if

$$\arg\max(\mathcal{N}(\tilde{x})) = y,\ \forall \tilde{x} \in \{||\tilde{x} - x||_\infty \leq \epsilon\}, \tag{2}$$

where the $\arg\max(\cdot)$ function obtains the classification label corresponding to input $\tilde{x}$ from the network output $y = \mathcal{N}(\cdot)$ and ϵ is called the perturbation radius (or perturbation for short).

In contrast, for a given range of inputs, standard robustness implies that the resulting outputs should vary within a tolerable range, and Lipchitz robustness requires that the range of output variations be proportional to the range of input variations. To formally reason about classification robustness, an important quantity, the robustness margin, is derived through robustness specification in the sense of Definition 1.

Definition 2 (Robustness specification matrix and robustness margin) An ϵ-robustness with respect to example x with ground truth label y is essentially associated with a specification matrix $C \in \mathbb{R}^{n_L \times n_L}$, which gives a linear combination for NN output:

$$C\mathcal{N}(x) \in \mathbb{R}^{n_L}, \tag{3}$$

where

$$C_{i,j} = \begin{cases} 1, & j = y, i \neq y, \\ -1, & i = j, i \neq y, \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

More importantly, each element in the vector $m := C\mathcal{N}(x)$ actually computes the robustness margin between the ground truth label y and other labels, i.e., the difference value between the prediction probability of the truth label and the prediction probability of other labels, denoted by $m(x, \epsilon)$. Moreover, the robustness margin plays an important role in NN verification related work.

Combining Definitions 1 and 2, we propose the following theorem to indicate the satisfiability of the robustness of NNs. The proof is obvious according to Definitions 1 and 2.

Theorem 1 The robustness with respect to a neural network $\mathcal{N}$, an input example x, and a perturbation radius ϵ holds if and only if $m(x,\epsilon)=C\mathcal{N}(x)>0$ (here $>$ is an element-wise operator).

To be more specific, let us assume that an example neural network $\mathcal{N}$ is designed to predict a label among five classes (i.e., five output neurons), and that the ground truth label of the input x is the third class. Then, according to Eq. (4), the specification matrix C is constructed as follows:

$$C=\begin{bmatrix}-1 & 0 & 1 & 0 & 0\\ 0 & -1 & 1 & 0 & 0\\ 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1 & -1 & 0\\ 0 & 0 & 1 & 0 & -1\end{bmatrix}. \tag{5}$$

Further, suppose that the outputs of two different examples $\tilde{x}_1$ and $\tilde{x}_2$ are

$$\mathcal{N}(\tilde{x}_1)=[0.10,0.15,0.60,0.10,0.05]^{\mathrm{T}} \tag{6}$$

and

$$\mathcal{N}(\tilde{x}_2)=[0.03,0.05,0.40,0.50,0.02]^{\mathrm{T}} \tag{7}$$

respectively (the outputs are processed by the softmax activation function, causing the sum to be 1), where $\tilde{x}_1,\tilde{x}_2\in\{\tilde{x}\mid ||\tilde{x}-x||_\infty\leq\epsilon\}$. Next, the computation results with respect to the specification matrix and outputs indicate whether the robustness holds or not. That is to say, because

$$C\mathcal{N}(\tilde{x}_1)=[0.50,0.45,0,0.50,0.55]\geq 0, \tag{8}$$

$\mathcal{N}$ is robust in example $\tilde{x}_1$ and it is not robust with respect to example $\tilde{x}_2$ due to

$$C\mathcal{N}(\tilde{x}_2)=[0.37,0.35,0,-0.1,0.48]\leq 0. \tag{9}$$

Consequently, the robustness of $\mathcal{N}$ with respect to (w.r.t.) the sample x and ϵ is falsified according to Theorem 1, because we find an input sample that does not satisfy $m(x,\epsilon)>0$. Such examples are called "adversarial examples" in the literature.

2.2 Robustness verification of DNNs

Resorting to the nice feature of robustness margin, various methods have been developed estimating $m(x,\epsilon)$ to verify the ϵ-robustness with respect to example x. If the estimated lower bound $\underline{m}(x,\epsilon)>0$, then network $\mathcal{N}$ is verifiably robust for any ℓ-norm perturbation less than ϵ on example x, as $\underline{m}(x,\epsilon)\leq m(x,\epsilon)$. Moreover, it has been found that computing the exact output ranges or margins of DNNs is non-convex and NP-complete (Katz et al., 2017; Weng et al., 2018b). In what follows, we use $\overline{*}$ and $\underline{*}$ to represent the estimated upper bound and lower bound of $*$, respectively. In this paper, our main focus is on the training methods of the IBP family, and we primarily introduce the estimation techniques for $m(x,\epsilon)$ using IBP and CROWN-IBP.

IBP: IBP adopts a simple and efficient bound propagation rule for margin estimation, which can be formulated in Eq. (10), where $|W^{(l)}|$ takes the element-wise absolute value and $\overline{h}^0=x+\epsilon$, $\underline{h}^0=x-\epsilon$ according to the interval arithmetic and perturbation definition. Thus, the robustness margin can be computed with the specification matrix C defined in Eq. (4). Specifically, the upper and lower bounds of the output neurons are obtained from Eq. (10) via layer-by-layer propagation, and then the bounds of $m(x,\epsilon)$ are computed with the specification matrix C, particularly the lower bound $\underline{m}(x,\epsilon)$. Note that the activation functions are required to be monotonically increasing to ensure the computation correctness of Eq. (10), while it is not a strict condition and common activation functions meet this requirement, such as Tanh and Sigmoid.

$$\begin{cases}\overline{z}^{(l)}=W^{(l)}\dfrac{\overline{h}^{(l-1)}+\underline{h}^{(l-1)}}{2}\\ \qquad\quad +|W^{(l)}|\dfrac{\overline{h}^{(l-1)}-\underline{h}^{(l-1)}}{2}+b^{(l)},\\ \underline{z}^{(l)}=W^{(l)}\dfrac{\overline{h}^{(l-1)}+\underline{h}^{(l-1)}}{2}\\ \qquad\quad -|W^{(l)}|\dfrac{\overline{h}^{(l-1)}-\underline{h}^{(l-1)}}{2}+b^{(l)},\\ \overline{h}^{(l)}=\sigma(\overline{z}^{(l)}),\ \ \underline{h}^{(l)}=\sigma(\underline{z}^{(l)}).\end{cases} \tag{10}$$

To sum up, IBP is computationally efficient, which is the dominant reason why it is popular in estimating the robustness margin. However, IBP provides very loose estimation bounds, and thus in most cases DNN robustness is unverifiable.

CROWN: To obtain tighter estimation bounds, CROWN uses linear relaxations on the non-linear

activation function ReLU to estimate the range bounds of DNNs. On the top of $\overline{z}^{(l)}$ and $\underline{z}^{(l)}$ computed by Eq. (10), CROWN first selects the stable neurons, including always active ($\underline{z}^{(l)} > 0$) and always inactive ($\overline{z}^{(l)} < 0$) neurons. Then for the other unstable neurons, CROWN shows a linear relaxation for ReLU activation:

$$\begin{cases} \alpha_k z_k^{(l)} \le \sigma(z_k^{(l)}) \le \frac{\overline{z}_k^{(l)}}{\overline{z}_k^{(l)} - \underline{z}_k^{(l)}} z_k^{(l)} - \frac{\overline{z}_k^{(l)} \underline{z}_k^{(l)}}{\overline{z}_k^{(l)} - \underline{z}_k^{(l)}}, \\ \underline{z}_k^{(l)} < 0 < \overline{z}_k^{(l)}, \quad k \in \{1, 2, \cdots, n_l\}, \end{cases} \tag{11}$$

where $0 \le \alpha_k \le 1$. To minimize the relaxation error further, CROWN proposes to select $\alpha_k = 1$ when $\overline{z}^{(l)} > |\underline{z}^{(l)}|$ and 0 otherwise. Combining the stable and unstable neuron cases, the ReLU function can be effectively replaced with a linear layer, estimating the upper and lower bounds of the output with

$$\underline{D}^{(l)} z^{(l)} \le \sigma(z^{(l)}) \le \overline{D}^{(l)} z^{(l)} + d^{(l)}, \tag{12}$$

where $\underline{D}^{(l)}$ and $\overline{D}^{(l)}$ are the diagonal weight matrices corresponding to the lower and upper cases of the relaxed ReLU respectively. Furthermore, considering the affine part (i.e., $Wx + b$) and ReLU function, the output of the i^{th} neuron is bounded by linear hyperplanes:

$$A_{i,:} \triangle x + b_{\text{L}} \le \mathcal{N}_i(x + \triangle x) \le A_{i,:} \triangle x + b_{\text{U}}, \tag{13}$$

where $A_{i,:} = W_{i,:}^{(l)} D^{(L-1)} W^{(L-1)} \cdots D^{(1)} W^{(1)}$ and b_{L} and b_{U} are two additional bias items.

The estimation is also carried out layer by layer, as in IBP. Similar work based on linear relaxations was conducted in DeepZ (Singh et al., 2018), DeepPoly (Singh et al., 2019), and Fast-Lin (Weng et al., 2018b), and we refer readers to Salman et al. (2019) for a comprehensive review.

CROWN-IBP: Just as its name implies, CROWN-IBP combines IBP and CROWN; i.e., it takes a linear combination of the estimated lower bounds from IBP and CROWN as the final estimation results.

These notations are used and retain the same meanings throughout this paper.

3 Methodology

During DNN training procedures, loss functions are used to quantify the difference between DNN predictions and the ground truth label and optimize the network parameters further (Duda et al., 2001; Murphy, 2012). In nominal training processes, without considering robustness, cross-entropy losses are generally adopted, evaluating the difference between output probability distribution $\mathcal{N}(x)$ and real distribution y (such as a one-hot vector):

$$\begin{aligned} \text{Loss}_{\text{acc}} :&= \mathcal{L}(\mathcal{N}(x); y; \theta) \\ &= -\sum_{(x,y)} \sum_{i=1}^{P} y_i \log(\mathcal{N}(x)_i), \end{aligned} \tag{14}$$

where P denotes the number of prediction classes and θ represents the network parameter set. The smaller the cross-entropy, the more accurate the output, and we label the loss function with acc.

3.1 Issue: locality leads to drawdown risk

In the literature on training robust DNNs, generally the evaluation of robustness has been considered together with the performance of NNs. According to Theorem 1, $\underline{m}(x, \epsilon)$ can indicate the satisfiability of NN robustness ($m(x, \epsilon) \ge \underline{m}(x, \epsilon) > 0$), and thus the new loss function is formed consequently:

$$\begin{aligned} \text{Loss} :&= \kappa \text{Loss}_{\text{acc}} + (1 - \kappa) \text{Loss}_{\text{rob}} \\ &= \kappa \mathcal{L}(\mathcal{N}(x); y; \theta) + (1 - \kappa) \mathcal{L}(-\underline{m}(x, \epsilon); y; \theta), \end{aligned} \tag{15}$$

where Loss_{rob} is the loss term evaluating DNN robustness, and κ ($0 < \kappa < 1$) is the hyperparameter that balances DNN accuracy and robustness.

Gowal et al. (2018) used IBP to estimate $\underline{m}(x, \epsilon)$, forming the following loss function:

$$\kappa \mathcal{L}(\mathcal{N}(x); y; \theta) + (1 - \kappa) \mathcal{L}(-\underline{m}_{\text{IBP}}(x, \epsilon); y; \theta). \tag{16}$$

With the estimation bound, IBP achieves outstanding performance, even though the bounds are relatively loose.

CROWN-IBP (Zhang H et al., 2020) combines IBP and CROWN to bound the robustness margins with forward (IBP-style) and backward (CROWN-style) estimations. With the lower bounds of robustness margins, we have

$$\underline{m}(x, \epsilon) = (1 - \beta) \underline{m}_{\text{IBP}}(x, \epsilon) + \beta \underline{m}_{\text{CROWN}}(x, \epsilon), \tag{17}$$

where β ($0 < \beta < 1$) is a hyperparameter that

balances the IBP estimation bound and CROWN estimation bound. CROWN-IBP obtains state-of-the-art performance by tightening the estimation bounds.

However, applying these methods to large perturbation cases tends to fail and cannot provide faithful and robust DNNs. In Fig. 1, we show the evaluated verified errors trained with representative robust DNN training methods IBP and CROWN-IBP against different perturbations. The verified error is the percentage of test samples where at least one element in $\underline{m}(x,\epsilon)$ is less than 0. It is a guaranteed upper bound of the test error under any ℓ_∞ perturbation (Zhang H et al., 2020). The verified errors are the indications of network robustness, and the larger the errors, the worse the robustness. The verified error varies significantly depending on the verification techniques used. To ensure fair comparisons, we calculate the verified error using a complete verification algorithm (Tjeng et al., 2019) and an exact solver Gurobi, as in Gowal et al. (2018). In Tjeng et al. (2019), the verification problem was formulated as a mixed integer programming (MIP) problem to be solved by Gurobi within a given time limit. If the solver times out, the verification problem is reformulated as a linear programming (LP) problem (Ehlers, 2017) and solved by Gurobi again. If neither approach can provide a solution within the given time limit, we consider the example to be attachable. The calculation method is also used in Section 4.

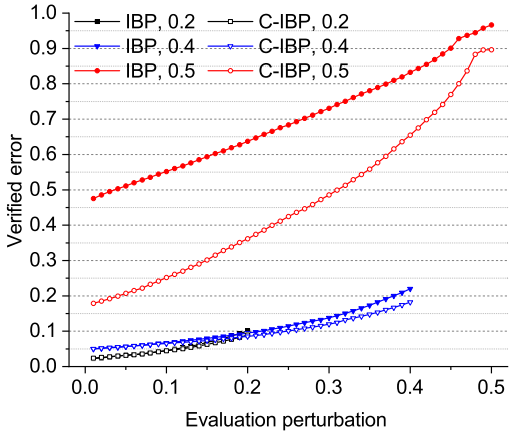

Fig. 1 Illustrations of the drawdown risk. IBP and CROWN-IBP (C-IBP for short) are used to train neural networks (DM-small in Section 4) against perturbations 0.2, 0.4, and 0.5, and the verified errors are evaluated in size 0.01

It can be observed that the verified errors w.r.t. 0.01 evaluated on the models trained by IBP and CROWN-IBP are both below 5% in the perturbation 0.2 and 0.4 cases. The verified errors w.r.t. the final perturbations (i.e., 0.2 and 0.4) are still acceptable, whereas for perturbation 0.5, the verified errors on 0.01 are much larger and they increase obviously, for both IBP and CROWN-IBP training methods. In summary, the verified error remains relatively low in the small perturbation cases, while it drastically increases in the large perturbation ones, which is termed the drawdown risk of performance. This means that the existing IBP-family training methods perform poorly on large perturbations, which is the key issue concerned and addressed in this paper.

Focusing on Eqs. (15)–(17), in each training epoch, the optimization renders the parameter updating robust with respect to a single perturbation value (i.e., the given ϵ). We name this training style "local robustness training" (LRT for short) afterward. Even though "warm-up" (training without Loss_{rob}) and "ramp-up" (increasing the perturbation value to the specified one gradually) processes are introduced in the practical implementation, LRT still exists in each training epoch. It continues to train NNs on the basis of the previous training, while the previous training effect becomes weaker and weaker as the training progresses, because only one perturbation value is considered each time. Consequently, locality can be regarded as a reasonable explanation for the drawdown risk.

3.2 Global robustness training strategy

To alleviate the drawdown risk of LRT, as shown in Fig. 2a, in this paper, we propose a novel training strategy, GRT. Ideally, the GRT strategy refers to training with the sub-perturbation range $0<\epsilon_{\text{train}}\leq\epsilon$ simultaneously, instead of a single perturbation ϵ. Then the robustness loss is formulated as

$$\text{Loss}_{\text{rob}}=\frac{1}{\epsilon}\int_0^{\epsilon}\mathcal{L}(-\underline{m}(x,r);y;\theta)\,\text{dr}. \tag{18}$$

Essentially, Eq. (18) is an ideal presentation, and in practice it can be relaxed and approximated as follows:

$$\text{Loss}_{\text{rob}}=\frac{1}{N}\sum_{i=1}^{N}\mathcal{L}(-\underline{m}(x,r_i);y;\theta),\ r_i=\frac{i}{N}\epsilon, \tag{19}$$

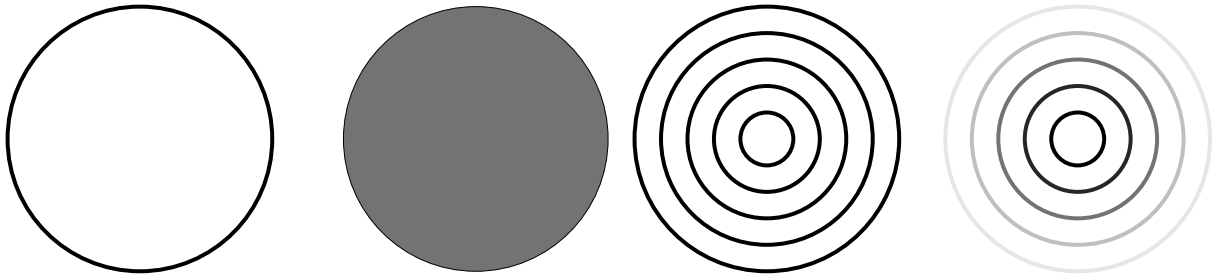

Fig. 2 Demonstration of the global and monotonically decreasing robustness training strategy versus local robust training (LRT): (a) LRT; (b) global robustness training (GRT) strategy (left is the ideal case and right is the practical case); (c) monotonically decreasing robustness training strategy

where N is the number of different sub-perturbations and is termed the robustness size. It can be seen that the GRT strategy degenerates into LRT when N takes 1.

With the GRT strategy, we would like to consider multiple perturbation values (sub-perturbations) during the training phase to avoid the aforementioned locality training. Moreover, when the robustness size is determined, we adopt a uniform sampling strategy to select sub-perturbations during the training phase, as shown in Eq. (19). The comparison of LRT and GRT styles is shown in Fig. 2, along with the ideal and practical cases in Fig. 2b, where circles in different radii represent different sub-perturbations.

3.3 Monotonically decreasing robustness training strategy

In addition to the GRT strategy, which takes multiple sub-perturbations into account in each training epoch, we propose a monotonically decreasing robustness training strategy that combines different robustness loss values corresponding to each sub-perturbation. In Eqs. (18) and (19), the GRT strategy takes the average value of all the losses by default and this undifferentiated combination of the losses is not sufficient, as illustrated in Section 4.

To organize the different loss values, a monotonically decreasing robustness training strategy makes sense. It means that the smaller a sub-perturbation is, the more weight its loss value deserves. Therefore, the robustness loss function becomes

$$\begin{cases} \mathrm{Loss}_{\mathrm{rob}} = \sum_{i=1}^{N} c_i \mathcal{L}(-\underline{m}(x, r_i); y; \theta), \\ r_i = \frac{i}{N}\epsilon, \quad c_i \propto \frac{1}{r_i}, \end{cases} \tag{20}$$

where c_i is the robustness weight. The reasons for adopting the monotonically decreasing robustness training strategy are mainly threefold:

1. From the view of robustness itself, the robustness with respect to smaller perturbations is more important than that of larger ones. That is to say, the smallest perturbation that a DNN can defend is a significant metric for evaluating its robustness. Additionally, in the domain of robustness verification, methods attempt to seek out the minimum perturbation as the robustness radius for the guidance of related applications.

2. From the view of training feasibility, it is more difficult for methods to train robust DNNs against larger perturbations compared to smaller ones. Therefore, during the robust training, more attention should be paid to the relatively feasible objectives and less to the hard ones. However, even with smaller weights, we do not give up optimizing the robustness against larger perturbations.

3. From the view of the relationship among the robustness strategies on different sub-perturbations, a DNN is more likely to be robust on larger perturbations if it is more robust with respect to smaller ones. This means that the efforts to improve the robustness against smaller perturbations during the training procedure also contribute to robustness in larger perturbation cases.

The monotonically decreasing robustness training strategy on the GRT style is demonstrated in Fig. 2c, where the darker the color, the more importance is assigned to the sub-perturbation. Moreover, considering the limit case of the perturbation approaching zero, our proposed monotonically decreasing robustness training strategy causes the robust training to degenerate into nominal training, and it

is absolutely correct to improve the accuracy metric in the case without perturbation.

4 Experiments

In this section, we describe how we implement our proposed training strategy on top of the representative and state-of-the-art work, CROWN-IBP, which we term GM-CROWN-IBP (global and monotonically decreasing CROWN-IBP). We use the PyTorch implementation version of CROWN-IBP (https://github.com/huanzhang12/CROWN-IBP). We evaluate GM-CROWN-IBP on the MNIST and CIFAR datasets with three network models used in CROWN-IBP (Zhang H et al., 2020), whose architectures are displayed in Table 1, and we refer readers to Gowal et al. (2018) for details. Herein, we follow their previously used notations, i.e., DM-small, DM-medium, and DM-large. All the experiments are carried out in the same platform, which includes 32 Intel Xeon Gold 6254 cores with 3.10 GHz frequency. The operating system is Ubuntu 18.04.3, and we list the detailed hyperparameters in the Appendix for reproducing the experimental results.

Table 1 Neural network architectures

DM-small	DM-medium	DM-large
Conv 8 4×4+2	Conv 32 3×3+1	Conv 64 3×3+1
Conv 16 4×4+1	Conv 32 4×4+2	Conv 64 3×3+1
FC 256	Conv 64 3×3+1	Conv 128 3×3+2
	Conv 64 4×4+2	Conv 128 3×3+1
	FC 512	Conv 128 3×3+1
	FC 512	FC 512

FC: fully connected

We compare GM-CROWN-IBP with IBP and CROWN-IBP in terms of the standard error and verified error, together with the best errors reported in the literature. The standard (clean) error refers to the proportion of classification errors, and recall that the verified error represents the percentage of test examples that violate robustness; they are used to evaluate the accuracy and robustness of models separately. Moreover, the verified error is a guaranteed upper bound of the test error under perturbations with any attack algorithms (Zhang H et al., 2020), such as FGSM (Goodfellow et al., 2015) and PGD (Madry et al., 2018). Similarly, the method of computing verified errors is the same as that used in Gowal et al. (2018), as declared in Section 3.

In this section, we focus on answering the following questions, corresponding to the next three subsections:

RQ1: Can GM-CROWN-IBP maintain state-of-the-art performance in existing small perturbation tests?

RQ2: How does GM-CROWN-IBP perform in large perturbation test cases?

RQ3: How does GM-CROWN-IBP perform with different robustness weights and robustness sizes?

4.1 Performance on small perturbations

In this subsection, we compare the standard errors and verified errors of the GM-CROWN-IBP training method against state-of-the-art methods in the existing test cases, which were first proposed in IBP and then used in CROWN-IBP.

In the test cases, the training perturbations (i.e., ϵ_{train}) are relatively small. For MNIST $\epsilon_{\text{train}} \in \{0.2, 0.4\}$ and for CIFAR $\epsilon_{\text{train}} \in \{2.2/255, 8.8/255\}$. We evaluate the models trained with Nominal (without perturbations), IBP, CROWN-IBP, and GM-CROWN-IBP against different evaluation perturbations (i.e., ϵ_{eval}). In addition, we list the best error results reported in the literature for detailed comparison. The reasons for having different ϵ's during the training and evaluation phases of the same network are twofold. On one hand, during the evaluation phase, we are concerned about the robustness within the perturbation radius ($\epsilon_{\text{eval}} \leq \epsilon_{\text{train}}$), instead of only the boundary ($\epsilon_{\text{eval}} = \epsilon_{\text{train}}$). On the other hand, the previous methods in Gowal et al. (2018) and Zhang H et al. (2020) use the same setting, and we thus follow their settings here for fair comparisons.

Table 2 shows the evaluation results and the reported best errors in different combinations of datasets, ϵ_{train} and ϵ_{eval} on models DM-large (with the MNIST dataset) and DM-small (with the CIFAR dataset).

It can be observed that the performance of our proposed global and monotonically decreasing robustness training strategy is close to that of CROWN-IBP and much better than that of IBP and other reported results in terms of robustness (the verified error column). It is also noteworthy that the accuracy metric (the standard error column) of the NN models trained by GM-CROWN-IBP surpasses

that by CROWN-IBP obviously in most test cases, and that GM-CROWN-IBP slightly underperforms CROWN-IBP in only one case. This indicates that using the global and monotonically decreasing robustness training strategy is more likely to give balanced consideration to accuracy and robustness.

Table 2 Comparison with state-of-the-art methods in the existing small perturbation cases

Dataset	Method	Standard error (%)	Verified error (%)
MNIST $\epsilon_{\text{eval}}=0.1$ $\epsilon_{\text{train}}=0.2$	IBP	1.14	2.81
	CROWN-IBP	**0.95**	**2.38**
	GM-CROWN-IBP	1.00	2.53
	Gowal et al. (2018)'s	1.06	2.92
	Dvijotham et al. (2018)'s	1.20	4.44
	Xiao et al. (2019)'s	1.05	4.40
	Wong et al. (2018)'s	1.08	3.01
	Mirman et al. (2018)'s	1.00	3.40
MNIST $\epsilon_{\text{eval}}=0.2$ $\epsilon_{\text{train}}=0.4$	IBP	2.74	5.46
	CROWN-IBP	2.17	4.31
	GM-CROWN-IBP	**1.60**	**3.90**
	Gowal et al. (2018)'s	1.66	4.53
	Xiao et al. (2019)'s	1.90	10.21
MNIST $\epsilon_{\text{eval}}=0.3$ $\epsilon_{\text{train}}=0.4$	IBP	2.74	8.73
	CROWN-IBP	2.17	7.03
	GM-CROWN-IBP	**1.60**	**6.90**
	Gowal et al. (2018)'s	1.66	8.21
	Wong et al. (2018)'s	14.87	43.10
	Xiao et al. (2019)'s	2.67	19.32
MNIST $\epsilon_{\text{eval}}=0.4$ $\epsilon_{\text{train}}=0.4$	IBP	2.74	14.80
	CROWN-IBP	2.17	**12.06**
	GM-CROWN-IBP	**1.60**	13.00
	Gowal et al. (2018)'s	1.66	15.01
CIFAR $\epsilon_{\text{eval}}=1/255$ $\epsilon_{\text{train}}=2.2/255$	IBP	48.21	53.12
	CROWN-IBP	45.72	**51.19**
	GM-CROWN-IBP	**41.05**	51.50
CIFAR $\epsilon_{\text{eval}}=3/255$ $\epsilon_{\text{train}}=8.8/255$	IBP	**40.54**	74.99
	CROWN-IBP	59.25	63.94
	GM-CROWN-IBP	53.56	**62.39**
CIFAR $\epsilon_{\text{eval}}=5/255$ $\epsilon_{\text{train}}=8.8/255$	IBP	**40.54**	91.45
	CROWN-IBP	59.25	**67.35**
	GM-CROWN-IBP	53.56	67.93

The standard errors of the Nominal method are 0.68% and 25.43% on the MNIST and CIFAR datasets, respectively. The baseline values of IBP and CROWN-IBP methods are evaluated with the pre-trained NN models provided in https://download.huan-zhang.com/models/crown-ibp/models_crown-ibp.tar.gz, and we adopt the best results reported in Zhang H et al. (2020) in some hard cases. The results reported in the literature may be obtained by using different architectures. The best results are in bold

Answer to RQ1: The model robustness performance of GM-CROWN-IBP is close to that of CROWN-IBP while the accuracy is improved significantly.

4.2 Performance on large perturbations

In this subsection, we compare the standard errors and verified errors of GM-CROWN-IBP training method on large training perturbations ϵ_{train}, most of which have not been tested before. For MNIST $\epsilon_{\text{train}} \in \{0.5, 0.6\}$ and for CIFAR $\epsilon_{\text{train}} \in \{17.6/255, 22/255\}$. Similarly, we evaluate the models trained with Nominal (without perturbations), IBP, CROWN-IBP, and GM-CROWN-IBP against different evaluation perturbations ϵ_{eval}. The MNIST-based network model is DM-small and the CIFAR-based one is DM-medium.

Table 3 lists the evaluation errors against different combinations of datasets, ϵ_{train} and ϵ_{eval}, on the models. It can be seen that GM-CROWN-IBP outperforms IBP and CROWN-IBP in terms of both standard error and verified error metrics in almost all test cases.

We also display the evaluated verified errors and total errors (summing standard and verified errors) in Fig. 3. For the MNIST-based $\epsilon_{\text{train}}=0.5$ case, the verified error is shown in Fig. 3a with evaluation step size 0.01 and the total error is displayed with step size 0.1 in Fig. 3b. For the CIFAR-based $\epsilon_{\text{train}}=22/255$ case, the verified error is evaluated with step size $1/255$ in Fig. 3c and the total error is shown with step size $4/255$ in Fig. 3d. The label GM-C-IBP is short for GM-CROWN-IBP due to the space limit.

As shown in Fig. 3, the aforementioned drawdown risk is alleviated to a great extent and the total error remains relatively small, on both MNIST and CIFAR datasets. Strictly speaking, the verified errors of GM-CROWN-IBP are much smaller than those of IBP and CROWN-IBP on most evaluation values, which are more acceptable and tolerable in NN applications and deployment. In addition, as shown in Fig. 3c, in some evaluation cases, the verified errors of GM-CROWN-IBP are a little larger. This may stem from the limitation of the IBP-family methods or the hyperparameter settings, and the verified errors might be further improved by sacrificing some model accuracy.

Answer to RQ2: The model robustness and accuracy of GM-CROWN-IBP generally outperform those of IBP and CROWN-IBP on large perturbations, and the drawdown risk is alleviated to a great extent.

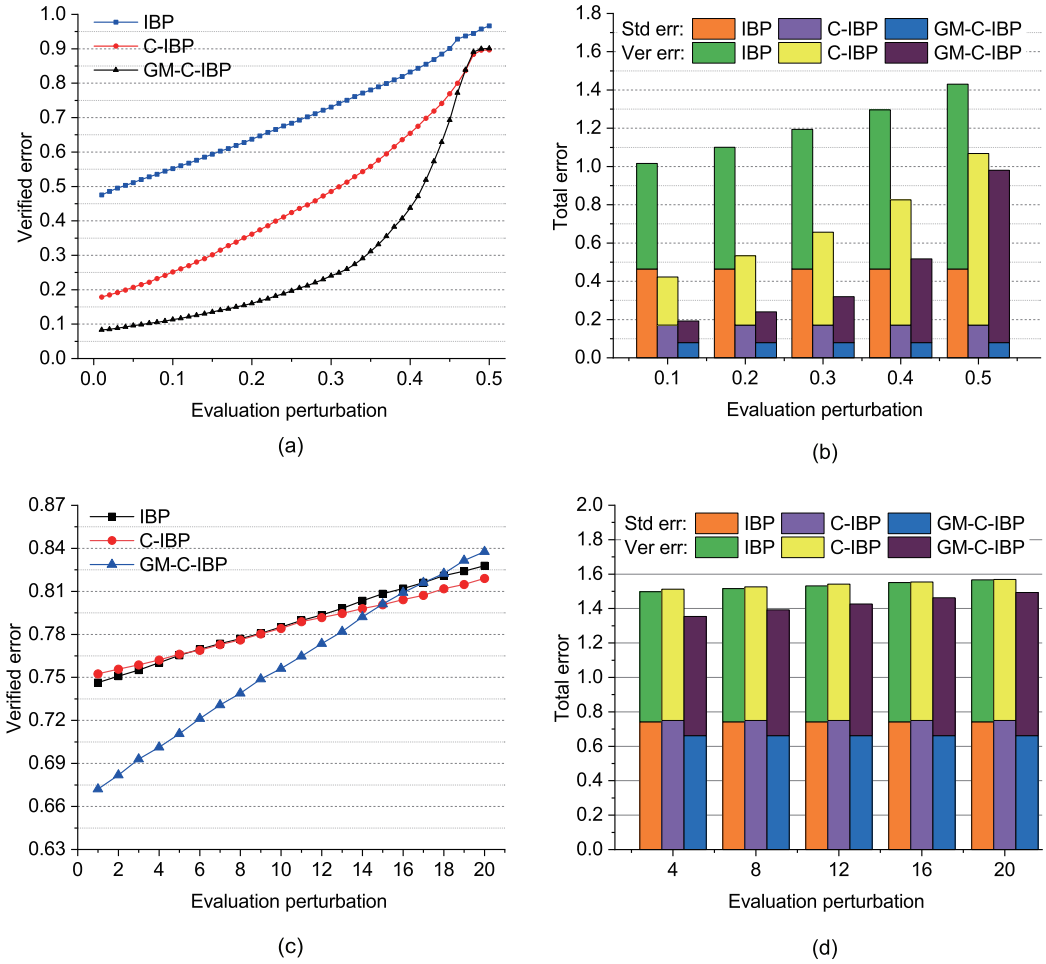

Fig. 3 Alleviating the drawdown risk: (a) verified errors on MNIST with evaluation step size 0.01; (b) total errors on MNIST with step size 0.1; (c) verified errors on CIFAR with step size 1/255; (d) total errors on CIFAR with step size 4/255

4.3 Comparison on robustness weights and sizes

In this subsection, we present our tests on GM-CROWN-IBP with different robustness weights and robustness sizes in the MNIST-based $\epsilon_{\text{train}} = 0.5$ case, to show the compatibility of our proposed training strategy.

First of all, we carry out some experiments to illustrate the necessity of monotonic decrease in the robustness weight. Fig. 4a shows the tested verified errors in four cases, i.e., a monotonically decreasing weight (GM-C-IBP-Dec in Fig. 4a), a random weight (GM-C-IBP-Ran), a monotonically increasing weight (GM-C-IBP-Inc), and the CROWN-IBP baseline (C-IBP) based on the global robustness training strategy. Among these, the weights for GM-C-IBP-Dec and GM-C-IBP-Inc are $1/i$ and $1/(N-i)$ respectively, where $i \in \{1, 2, \cdots, N\}$, and N is the robustness size, set as 10. It can be seen that the monotonically increasing weight performs in a manner similar to the random robustness weight, and that the monotonically decreasing weight outperforms the two other weight types in terms of reducing the drawdown risk (especially the evaluated verified errors in large perturbation cases).

Furthermore, we take five types of monotonically decreasing robustness weights into account. These weights c_i are (1) $1/\sqrt{i}$, (2) $1/i$, (3) $1/i^2$, (4) $1/\ln(i+1)$ (to avoid the division-by-zero error, 1 is added to index i in the ln function of the denominator), and (5) $1/N$ (constant). The first four weights are strictly monotonically decreasing, whereas the last one is not.

The verified errors of different robustness

weights are displayed in Fig. 4b. Compared with the CROWN-IBP baseline, all the robustness weights relax the drawdown risk, and the $1/\sqrt{i}$-style and $1/\ln(i+1)$-style robustness weights work best in terms of reducing the verified error. Moreover, it is actually an ablation experiment with the robustness weight type (5) (trained only with the GRT strategy), and compared with the CROWN-IBP baseline, it can be seen that the GRT strategy has made great contributions to the alleviation of the drawdown risk, while its performance can be further improved by our monotonically decreasing robustness weight.

Table 3 Comparison with state-of-the-art methods against large perturbations

Dataset	Method	Standard error (%)	Verified error (%)
MNIST	IBP	46.38	62.79
$\epsilon_{\text{eval}}=0.2$	CROWN-IBP	17.13	36.19
$\epsilon_{\text{train}}=0.5$	GM-CROWN-IBP	**7.98**	**16.00**
MNIST	IBP	46.38	72.14
$\epsilon_{\text{eval}}=0.3$	CROWN-IBP	17.13	48.55
$\epsilon_{\text{train}}=0.5$	GM-CROWN-IBP	**7.98**	**24.03**
MNIST	IBP	46.38	81.92
$\epsilon_{\text{eval}}=0.4$	CROWN-IBP	17.13	65.48
$\epsilon_{\text{train}}=0.5$	GM-CROWN-IBP	**7.98**	**43.73**
MNIST	IBP	30.34	61.73
$\epsilon_{\text{eval}}=0.25$	CROWN-IBP	**16.47**	48.85
$\epsilon_{\text{train}}=0.6$	GM-CROWN-IBP	17.10	**31.25**
MNIST	IBP	30.34	88.70
$\epsilon_{\text{eval}}=0.35$	CROWN-IBP	**16.47**	80.72
$\epsilon_{\text{train}}=0.6$	GM-CROWN-IBP	17.10	**50.14**
CIFAR	IBP	71.77	72.39
$\epsilon_{\text{eval}}=1/255$	CROWN-IBP	70.35	70.89
$\epsilon_{\text{train}}=17.6/255$	GM-CROWN-IBP	**64.27**	**65.60**
CIFAR	IBP	71.77	75.93
$\epsilon_{\text{eval}}=7/255$	CROWN-IBP	70.35	74.38
$\epsilon_{\text{train}}=17.6/255$	GM-CROWN-IBP	**64.27**	**73.81**
CIFAR	IBP	74.23	79.82
$\epsilon_{\text{eval}}=14/255$	CROWN-IBP	74.82	79.46
$\epsilon_{\text{train}}=22/255$	GM-CROWN-IBP	**66.15**	**79.22**

The standard errors of the Nominal method are 1.12% and 13.80% on the MNIST and CIFAR datasets, respectively. The best results are in bold

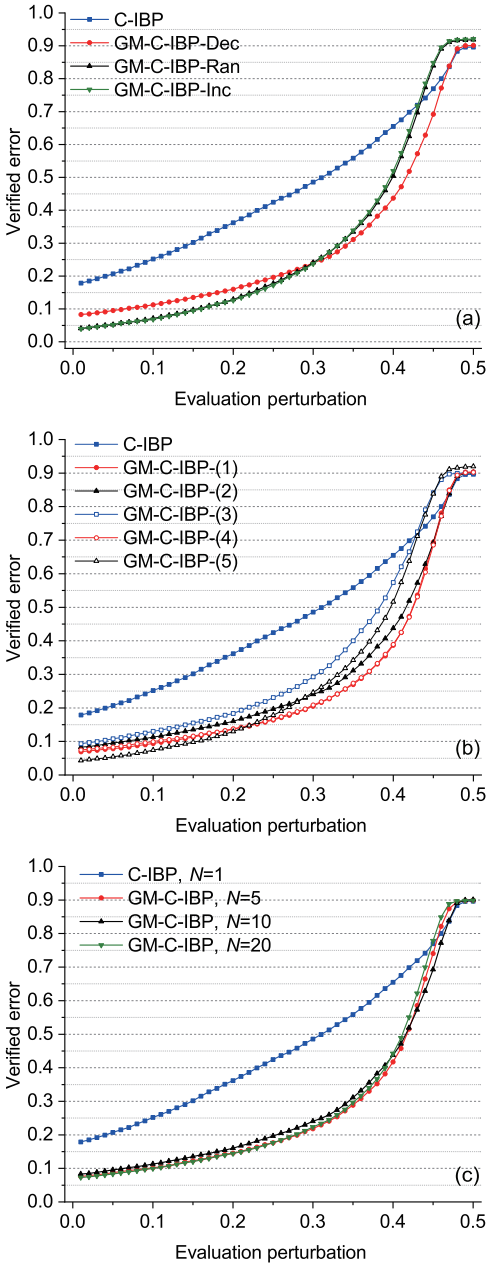

Fig. 4 Performance comparisons of GM-CROWN-IBP on different robustness weight types (a), monotonically decreasing robustness weights (b), and robustness sizes (c)

We also show the verified errors of GM-CROWN-IBP with respect to different robustness sizes $N \in \{5, 10, 20\}$ in Fig. 4c. It can be observed that a large performance discrepancy does not exist among different robustness sizes. It means that a large robustness size is not necessary during the global and monotonically decreasing robustness training; thus, the concern on computation memory and duration can be eased greatly.

Answer to RQ3: The performance of GM-CROWN-IBP can be further improved by the selected monotonically decreasing robustness weight, and it does not vary greatly on different robustness sizes.

Additionally, in all the training phases, GM-CROWN-IBP adopts the same hyperparameter settings as CROWN-IBP to show the strengths of our

training strategy. It also means that our training strategy works more stably in terms of hyperparameter selections.

5 Discussions

In this section, we discuss and explain some well-known and related concepts or techniques that are easily confused with those in our proposed global training strategy.

5.1 Global robustness training strategy vs. global robustness

The global robustness of NNs was first defined in Leino et al. (2021), as an extension of local robustness for NNs. The global robustness captures the operational properties of local robustness for robust training, based on the utilization of global Lipschitz bounds. We refer interested readers to Hein and Andriushchenko (2017), Weng et al. (2018a, 2018b), and Huster et al. (2019) for a more comprehensive view. Different from local robustness, global robustness requires that NN classifiers maintain a minimum separation margin between any pair of regions that are assigned different prediction labels.

Compared with our proposed global training strategy, the differences are mainly twofold. First, the global training strategy is a training method, instead of a newly defined robustness type, whereas global robustness is a robustness description on NNs and there have been some existing training methods on top of it (Leino et al., 2021). Second, in this paper, rather than global robustness, our global training strategy is applied to the more widely used local robustness of NNs.

5.2 Global robustness training strategy vs. randomized smoothing

In recent years, researchers also proposed some robust training methods that provide stochastic guarantees on the robustness property of NNs; that is, NNs tend to be robust with a high probability. Randomized smoothing (Cohen et al., 2019; Lecuyer et al., 2019) is one of the most representative methods. By contrast, our global training method provides deterministic robustness guarantees, like the well-known IBP and CROWN-IBP, which are more acceptable in safety-critical domains. However, randomized smoothing is generally estimated with a false positive rate around 0.1% (Cohen et al., 2019), meaning that there exist thousands of incorrectly certified instances, and this result is less reliable and tolerable. Moreover, randomized smoothing requires a large number of evaluated sample instances (as many as 10 000 samples (Cohen et al., 2019)), which may greatly deteriorate the computation efficiency.

6 Conclusions

In this paper, we focus on alleviating the unexpected drawdown risk encountered in IBP-family robust DNN training methods. First, we introduce multiple sub-perturbations into the training epochs to consider robustness with respect to some certain perturbations globally. Subsequently, we organize different robustness loss values in a monotonically decreasing style to further improve the training performance. Experimental evaluations show that the training strategy significantly reduces the drawdown risk and maintains the original performance on small perturbations. Furthermore, our proposed global and monotonically decreasing robustness training strategy obtains surprising performance on model accuracy.

In future work, we will consider further optimization of the training duration and burdens, such as the utilization of a parallel computation mechanism and a more memory-saved estimation on the robustness margin.

Contributors

Zhen LIANG designed the research. Taoran WU, Wanwei LIU, Bai XUE, Wenjing YANG, Ji WANG, and Zhengbin PANG improved the research design. Zhen LIANG and Taoran WU implemented the experiments. Zhen LIANG drafted the paper. Wanwei LIU, Ji WANG, and Taoran WU helped organize the paper. Zhen LIANG revised and finalized the paper.

Compliance with ethics guidelines

Ji WANG is an editorial board member of *Frontiers of Information Technology & Electronic Engineering*, and he was not involved with the peer review process of this paper. Zhen LIANG, Taoran WU, Wanwei LIU, Bai XUE, Wenjing YANG, Ji WANG, and Zhengbin PANG declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Balunović M, Baader M, Singh G, et al., 2019. Certifying geometric robustness of neural networks. Proc 33rd Int Conf on Neural Information Processing Systems, Article 1372.

Bojarski M, Testa DD, Dworakowski D, et al., 2016. End to end learning for self-driving cars. https://arxiv.org/abs/1604.07316

Casadio M, Komendantskaya E, Daggitt ML, et al., 2022. Neural network robustness as a verification property: a principled case study. Proc 34th Int Conf on Computer Aided Verification, p.219-231. https://doi.org/10.1007/978-3-031-13185-1_11

Chen XL, He KM, 2021. Exploring simple Siamese representation learning. Proc IEEE/CVF Conf on Computer Vision and Pattern Recognition, p.15750-15758. https://doi.org/10.1109/CVPR46437.2021.01549

Cohen JM, Rosenfeld E, Kolter JZ, 2019. Certified adversarial robustness via randomized smoothing. Proc 36th Int Conf on Machine Learning, p.1310-1320.

Devlin J, Chang MW, Lee K, et al., 2018. BERT: pre-training of deep bidirectional transformers for language understanding. Proc Conf of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, p.4171-4186. https://doi.org/10.18653/v1/N19-1423

Du TY, Ji SL, Shen LJ, et al., 2021. Cert-RNN: towards certifying the robustness of recurrent neural networks. Proc ACM SIGSAC Conf on Computer and Communications Security, p.516-534. https://doi.org/10.1145/3460120.3484538

Duda RO, Hart PE, Stork DG, 2001. Pattern Classification (2nd Ed.). Wiley, New York, USA.

Dvijotham K, Gowal S, Stanforth R, et al., 2018. Training verified learners with learned verifiers. https://arxiv.org/abs/1805.10265

Ehlers R, 2017. Formal verification of piece-wise linear feed-forward neural networks. Proc 15th Int Symp on Automated Technology for Verification and Analysis, p.269-286. https://doi.org/10.1007/978-3-319-68167-2_19

Goodfellow IJ, Shlens J, Szegedy C, 2015. Explaining and harnessing adversarial examples. Proc 3rd Int Conf on Learning Representations.

Gowal S, Dvijotham K, Stanforth R, et al., 2018. On the effectiveness of interval bound propagation for training verifiably robust models. https://arxiv.org/abs/1810.12715

Guo XW, Wan WJ, Zhang ZD, et al., 2021. Eager falsification for accelerating robustness verification of deep neural networks. Proc 32nd IEEE Int Symp on Software Reliability Engineering, p.345-356. https://doi.org/10.1109/ISSRE52982.2021.00044

Hein M, Andriushchenko M, 2017. Formal guarantees on the robustness of a classifier against adversarial manipulation. Proc 31st Int Conf on Neural Information Processing Systems, p.2266-2276.

Huster T, Chiang CYJ, Chadha R, 2019. Limitations of the Lipschitz constant as a defense against adversarial examples. Proc Joint European Conf on Machine Learning and Knowledge Discovery in Databases, p.16-29. https://doi.org/10.1007/978-3-030-13453-2_2

Katz G, Barrett C, Dill DL, et al., 2017. Reluplex: an efficient SMT solver for verifying deep neural networks. Proc 29th Int Conf on Computer Aided Verification, p.97-117. https://doi.org/10.1007/978-3-319-63387-9_5

Ko CY, Lyu ZY, Weng L, et al., 2019. POPQORN: quantifying robustness of recurrent neural networks. Proc 36th Int Conf on Machine Learning, p.3468-3477.

Lecuyer M, Atlidakis V, Geambasu R, et al., 2019. Certified robustness to adversarial examples with differential privacy. Proc IEEE Symp on Security and Privacy, p.656-672. https://doi.org/10.1109/SP.2019.00044

Leino K, Wang ZF, Fredrikson M, 2021. Globally-robust neural networks. Proc 38th Int Conf on Machine Learning, p.6212-6222.

Li JL, Liu JC, Yang PF, et al., 2019. Analyzing deep neural networks with symbolic propagation: towards higher precision and faster verification. Proc 26th Int Static Analysis Symp, p.296-319. https://doi.org/10.1007/978-3-030-32304-2_15

Liang Z, Liu WW, Wu TR, et al., 2023. Advances and prospects of training methods for robust neural networks. *Sci Technol Fores*, 2(1):78-89 (in Chinese). https://doi.org/10.3981/j.issn.2097-0781.2023.01.006

Liu JX, Xing YH, Shi XM, et al., 2022. Abstraction and refinement: towards scalable and exact verification of neural networks. https://arxiv.org/abs/2207.00759

Liu WW, Song F, Zhang THR, et al., 2020. Verifying ReLU neural networks from a model checking perspective. *J Comput Sci Technol*, 35(6):1365-1381. https://doi.org/10.1007/s11390-020-0546-7

Ma L, Juefei-Xu F, Zhang FY, et al., 2018. DeepGauge: multi-granularity testing criteria for deep learning systems. Proc 33rd ACM/IEEE Int Conf on Automated Software Engineering, p.120-131. https://doi.org/10.1145/3238147.3238202

Madry A, Makelov A, Schmidt L, et al., 2018. Towards deep learning models resistant to adversarial attacks. Proc 6th Int Conf on Learning Representations.

Mirman M, Gehr T, Vechev MT, 2018. Differentiable abstract interpretation for provably robust neural networks. Proc 35th Int Conf on Machine Learning, p.3575-3583.

Murphy KP, 2012. Machine Learning: a Probabilistic Perspective. MIT Press, Cambridge, USA.

Ryou W, Chen JY, Balunovic M, et al., 2021. Scalable polyhedral verification of recurrent neural networks. Proc 33rd Int Conf on Computer Aided Verification, p.225-248. https://doi.org/10.1007/978-3-030-81685-8_10

Salman H, Yang G, Zhang H, et al., 2019. A convex relaxation barrier to tight robust verification of neural networks. Proc 33rd Int Conf on Neural Information Processing Systems, Article 882.

Singh G, Gehr T, Mirman M, et al., 2018. Fast and effective robustness certification. Proc 32nd Int Conf on Neural Information Processing Systems, p.10825-10836.

Singh G, Gehr T, Püschel M, et al., 2019. An abstract domain for certifying neural networks. Proc ACM on Programming Languages, p.1-30. https://doi.org/10.1145/3290354

Sun B, Sun J, Dai T, et al., 2021. Probabilistic verification of neural networks against group fairness. Proc 24th Int Symp on Formal Methods, p.83-102. https://doi.org/10.1007/978-3-030-90870-6_5

Tian Y, Yang WJ, Wang J, 2021. Image fusion using a multi-level image decomposition and fusion method. *Appl Opt*, 60(24):7466-7479. https://doi.org/10.1364/AO.432397

Tjeng V, Xiao KY, Tedrake R, 2019. Evaluating robustness of neural networks with mixed integer programming. Proc 7th Int Conf on Learning Representations.

Tran HD, Manzanas Lopez D, Musau P, et al., 2019. Star-based reachability analysis of deep neural networks. Proc 3rd Int Symp on Formal Methods, p.670-686. https://doi.org/10.1007/978-3-030-30942-8_39

Wang SQ, Pei KX, Whitehouse J, et al., 2018a. Efficient formal safety analysis of neural networks. Proc 32nd Int Conf on Neural Information Processing Systems, p.6369-6379.

Wang SQ, Chen YZ, Abdou A, et al., 2018b. MixTrain: scalable training of formally robust neural networks. https://arxiv.org/abs/1811.02625

Weng TW, Zhang H, Chen PY, et al., 2018a. Evaluating the robustness of neural networks: an extreme value theory approach. Proc 6th Int Conf on Learning Representations.

Weng TW, Zhang H, Chen HG, et al., 2018b. Towards fast computation of certified robustness for ReLU networks. Proc 35th Int Conf on Machine Learning, p.5273-5282.

Wong E, Schmidt FR, Metzen JH, et al., 2018. Scaling provable adversarial defenses. Proc 32nd Int Conf on Neural Information Processing Systems, p.8410-8419.

Xiao KY, Tjeng V, Shafiullah NM, et al., 2019. Training for faster adversarial robustness verification via inducing ReLU stability. Proc 7th Int Conf on Learning Representations.

Zhang H, Weng TW, Chen PY, et al., 2018. Efficient neural network robustness certification with general activation functions. Proc 32nd Int Conf on Neural Information Processing Systems, p.4944-4953.

Zhang H, Chen HG, Xiao CW, et al., 2020. Towards stable and efficient training of verifiably robust neural networks. Proc 8th Int Conf on Learning Representations.

Zhang YD, Zhao Z, Chen GK, et al., 2022. QVIP: an ILP-based formal verification approach for quantized neural networks. Proc 37th IEEE/ACM Int Conf on Automated Software Engineering, p.82:1-82:13. https://doi.org/10.1145/3551349.3556916

Zhao Z, Zhang YD, Chen GK, et al., 2022. CLEVEREST: accelerating CEGAR-based neural network verification via adversarial attacks. Proc 29th Int Static Analysis Symp, p.449-473. https://doi.org/10.1007/978-3-031-22308-2_20

Appendix: Hyperparameter settings with respect to IBP and CROWN-IBP on the MNIST and CIFAR datasets

Here we supplement the detailed hypermeter values for reproducing the experimental results reported in the paper. The hyperparameters are associated with the codes provided in https://github.com/huanzhang12/CROWN-IBP. Tables A1 and A2 list the hyperparameter settings with respect to training methods IBP and CROWN-IBP on the MNIST and CIFAR datasets, respectively. The models used in the Methodology and Experiments sections are the same NN models trained with these hyperparameters.

It is worth noting that the GM-CROWN-IBP hyperparameter values are omitted here, because GM-CROWN-IBP adopts the same hyperparameter values as CROWN-IBP.

Table A1 Hyperparameter settings with respect to IBP and CROWN-IBP on the MNIST dataset

Parameter	Value							
	$\epsilon = 0.2$		$\epsilon = 0.4$		$\epsilon = 0.5$		$\epsilon = 0.6$	
	IBP	C-IBP	IBP	C-IBP	IBP	C-IBP	IBP	C-IBP
Lr	1e-3	1e-3	1e-3	5e-4	1e-3	5e-4	1e-3	5e-4
Lr_decay_factor	0.1	0.1	0.1	–	0.1	–	0.1	–
Lr_decay_milestones	[25, 42]	[25, 42]	[25, 42]	–	[25, 42]	–	[25, 42]	–
Schedule_start	3	3	3	1	10	1	10	1
Epsilon	0.2	0.2	0.4	0.4	0.5	0.5	0.6	0.6
Schedule_length	18	18	18	61	31	61	31	61
Batch_size	100	100	100	256	100	256	256	256

For all cases, the method used is robust_natural, the number of epochs is 100, Lr_decay_step is null, weight_decay is 0, the optimizer is Adam, starting_epsilon is 0, final-beta is 1, and final-kappa is 0.5

Table A2 Hyperparameter settings with respect to IBP and CROWN-IBP on the CIFAR dataset

Parameter	Value							
	$\epsilon = 2.2/255$		$\epsilon = 8.8/255$		$\epsilon = 17.6/255$		$\epsilon = 22/255$	
	IBP	C-IBP	IBP	C-IBP	IBP	C-IBP	IBP	C-IBP
Epsilon	0.008 63	0.008 63	0.034 51	0.034 51	0.069 02	0.069 02	0.086 27	0.086 27

For all cases, the method used is robust, the number of epochs is 200, Lr is 1e-3, Lr_decay_factor and Lr_decay_milestones are not introduced herein, Lr_decay_step is null, weight_decay is 0, the optimizer is Adam, schedule_start is 10, starting_epsilon is 0, schedule_length is 121, batch_size is 128, final-beta is 1, and final-kappa is 0.5