Learning Diverse-Structured Networks for Adversarial Robustness

Xuefeng Du [*1 2] **Jingfeng Zhang** [*3] **Bo Han** [1] **Tongliang Liu** [4] **Yu Rong** [5]
Gang Niu [3] **Junzhou Huang** [5] **Masashi Sugiyama** [3 6]

Abstract

In *adversarial training* (AT), the main focus has been the objective and optimizer while the model has been less studied, so that the models being used are still those classic ones in *standard training* (ST). Classic *network architectures* (NAs) are generally worse than searched NAs in ST, which should be the same in AT. In this paper, we argue that NA and AT cannot be handled independently, since given a dataset, the optimal NA in ST would be *no longer optimal* in AT. That being said, AT is time-consuming itself; if we directly search NAs in AT over large *search spaces*, the computation will be practically infeasible. Thus, we propose a *diverse-structured network* (DS-Net), to significantly reduce the size of the search space: instead of low-level operations, we only consider predefined *atomic blocks*, where an atomic block is a time-tested building block like the residual block. There are only a few atomic blocks and thus we can weight all atomic blocks rather than find the best one in a searched block of DS-Net, which is an essential trade-off between *exploring* diverse structures and *exploiting* the best structures. Empirical results demonstrate the advantages of DS-Net, i.e., weighting the atomic blocks.

1. Introduction

Safety-critical areas, such as autonomous driving, healthcare and finance, necessitate deep models to be adversarially robust and generalize well (Goodfellow et al., 2015). Recently, *adversarial training (AT)* has been shown effective for improving the robustness of different models (Madry et al., 2018). Compared with *standard training (ST)* on natural data, AT is a new training scheme, which generates adversarial examples on the fly and employs them to update model parameters (Madry et al., 2018; Zhang et al., 2019b; 2020b; Wong et al., 2020; Pang et al., 2021).

Table 1. Performance misalignment for different NA in terms of robustness (PGD-20) and standard accuracy. Models which perform better after ST are not necessarily more robust after AT. AdaRKNet denotes dynamic system-inspired network (Kim et al., 2020). AT is performed by PGD-10 with a perturbation bound of 0.031. Robustness is evaluated with the same perturbation bound of 0.031.

Model	Standard Acc.	Ranking	Robustness	Ranking
WRN-28-10	0.9646	1	0.4872	3
ResNet-62	0.9596	2	0.4855	4
DenseNet-121	0.9504	3	0.4993	2
MobileNetV2	0.9443	4	0.4732	6
AdaRKNet-62	0.9403	5	0.5016	1
ResNet-50	0.9362	6	0.4807	5

The research focus of AT has mainly been the *objective* and *optimizer* while the *model* has been less studied. Therefore, it is urgent to explore the influence of network architectures (NAs) for adversarial robustness. Some emerging studies imply that the classic human-designed NAs (e.g., ResNet (He et al., 2016)), specified for ST, may not be suitable for AT. For example, Li et al. (2020a) and Kim et al. (2020) argued that the forward propagation of ResNet can be explained as an explicit Euler discretization of an ordinary differential equation (ODE), which leads to unstable predictions given perturbed inputs. They proposed to change NAs according to more stable numerical schemes and thus obtained higher model robustness. Meanwhile, Xie et al. (2020a) discovered that the ReLU activation function weakens AT due to its non-smooth nature. They replaced ReLU with its smooth approximations to improve robustness.

In addition, we show in Tab. 1 a *misalignment phenomenon* for different NA in terms of their robustness after AT and standard accuracy after ST. This phenomenon further demonstrates a fact that *manually-crafted NAs for ST may not be suitable for AT*. Specifically, among various architectures, a clear trend can be observed that models which perform better in terms of standard accuracy may not be more robust. Moreover, a newly-designed AdaRKNet-62 (Kim et al., 2020) has the *biggest misalignment*, which inspires us to rethink NAs for AT. Namely, all intriguing results

[*]Equal contribution [1]Hong Kong Baptist University [2]University of Wisconsin-Madison [3]RIKEN [4]University of Sydney [5]Tencent AI Lab [6]University of Tokyo. Correspondence to: Bo Han <bhanml@comp.hkbu.edu.hk>, Tongliang Liu <tongliang.liu@sydney.edu.au>, Gang Niu <gang.niu@riken.jp>.

Proceedings of the 38th International Conference on Machine Learning, PMLR 139, 2021.

suggest improving adversarial robustness requires carefully modified structures. Nevertheless, designing an optimal architecture for AT is still a challenging problem.

One straightforward remedy is to search robust NAs (Guo et al., 2020; Dong et al., 2020a), where the key to success is *exploring* diverse structures. However, it suffers from computational inefficiency and is sometimes not effective. Specifically, AT is inherently time-consuming, and searching NAs over large spaces with AT drastically scales up the computation overhead (Guo et al., 2020). Besides, searching over a large space (denoted as the search phase) requires pruning less useful operations and retraining the model from scratch (denoted as the evaluation phase), which naturally leads to an optimization gap between these two phases. As claimed in Xie et al. (2020b), the search phase seeks to optimize a large network, but a well-optimized large network does not necessarily produce high-quality sub-architectures. Thus, the searched architecture may not always be more robust or generalizing better. This motivates us to find an effective architecture for AT which is easy to build and efficient to train while encouraging flexible NA exploration.

In this paper, we introduce a novel network design strategy which trades off *exploring* diverse structures and *exploiting* the best structures. Concretely, we propose a Diverse-Structured Network (DS-Net) as a novel solution to the trade-off (see Fig. 1). Specifically, DS-Net consists of a sequence of modules. To significantly reduce the fine-grained search space, each module contains a few off-the-shelf time-tested building blocks (i.e., predefined atomic blocks in Section 3.3), which are either human-designed or search-based blocks that can be flexibly chosen. To encourage structure exploration, besides block parameters, we introduce a set of learnable attention weights to weight the outputs of these atomic blocks rather than finding the best one. The weights are concurrently optimized with the block parameters by the robust training objective and are fixed for evaluation.

Our end-to-end design strategy is analogous to manual design, which operates on a fixed set of atomic blocks but leverages attention weights to flexibly explore their relationship. It is different from searching robust NAs that determines the local structures inside each block by two-stage training. Additionally, the structure of DS-Net is consistent during AT, which does not require the operation of pruning that causes the optimization gap during searching NAs. Thus, our main contributions are summarized as follows:

- We propose a novel DS-Net that trades off *exploring* diverse structures and *exploiting* the best structures. DS-Net remains the computational efficiency and effectiveness, which are limited in existing methods.
- DS-Net allows for a flexible choice among powerful off-the-shelf atomic blocks including human-designed and search-based blocks, which are easy to understand, build, and robustify/generalize well.
- DS-Net learns attention weights of predefined atomic blocks based on the objective of AT. It empirically performs better than powerful defense architectures with less parameters on CIFAR-10 and SVHN.

2. Related Work

Adversarial defense. Existing literature on the adversarial defense of neural networks can be roughly divided into two categories, namely certified robustness (Tsuzuku et al., 2018; Zhang & Liang, 2019) and empirical robustness (Cai et al., 2018; Madry et al., 2018; Zhang et al., 2020a). The former one focuses on either training provably robust models (Wong & Kolter, 2018) or obtaining certified models via random smoothing (Cohen et al., 2019), but often with a limited robustness compared to the latter approach. Empirical approaches usually rely on different techniques, such as input transformation (Dziugaite et al., 2016), randomization (Xie et al., 2018) and model ensemble (Liu et al., 2018).

However, most of them are evaded by adaptive attacks (Athalye et al., 2018), while the most effective approaches till now are AT (Madry et al., 2018) and its variants (Zhang et al., 2019b; Wang et al., 2020b). Based on it, many improvements were proposed, e.g., by metric learning (Li et al., 2019), self-supervised learning (Naseer et al., 2020), model-conditional training (Wang et al., 2020a), weight perturbation (Wu et al., 2020), generative models (Wang & Yu, 2019) and semi-supervised learning (Zhai et al., 2019). Besides, several works attempted to speed up AT, such as computation reuse (Zhang et al., 2019a), adaptive inner maximization steps (Wang et al., 2019b; Zhang et al., 2020b) and one-step approximation (Wong et al., 2020; S. & Babu, 2020). Note DS-Net improves AT from the viewpoint of network structure.

Several works attempted to improve adversarial robustness by diversity (Pang et al., 2019; Dong et al., 2020b; Abbasi et al., 2020; Kariyappa & Qureshi, 2019), but neither of them focused on learning diverse-structured networks.

Robust network architecture. To obtain robust NAs, researchers developed smooth activation functions (Xie et al., 2020a), channel activation suppressing (Bai et al., 2021), dynamical system-inspired networks (Li et al., 2020a; Kim et al., 2020), model ensemble (Wang et al., 2019a), sparse coding (Cazenavette et al., 2020) and regularization (Bui et al., 2020; Rahnama et al., 2020) to enhance robustness. Besides, several works explored searching robust architectures (Cubuk et al., 2018; Li et al., 2020b; Hosseini et al., 2020; Vargas & Kotyan, 2019; Yue et al., 2020; Chen et al., 2020; Ning et al., 2020). Note DS-Net attempts to trade off *exploring* diverse structures and *exploiting* the best structures, which is orthogonal to these approaches.

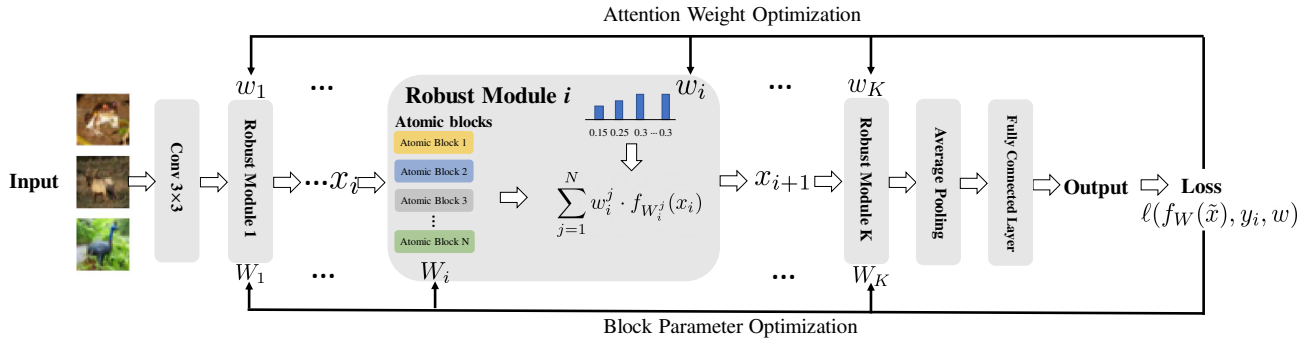

Figure 1. The network structure of DS-Net, where $\tilde{x}$ means the perturbed data and $f_W(\cdot)$ denotes the DS-Net with parameter W. The output from the i-th module x_i goes through every atomic block and outputs x_i^j. The output x_{i+1} is calculated by multiplying attention weights w_i^j $(j = 1, ..., N)$ and the corresponding outputs in a block-wise fashion. Here, N is the number of atomic blocks, K is the number of robust modules in sequence, and W_i denotes the block parameter for the i-th module.

3. Proposed Approach

3.1. Preliminaries

In this section, we briefly introduce the background for AT.

Standard AT. For each input x, let the input feature space $\mathcal{X}$ with the infinity distance metric $d_{\inf}(x,x') = \|x-x'\|_\infty$ be $(\mathcal{X}, d_\infty)$, the closed ball of radius $\varepsilon > 0$ centered at x in $\mathcal{X}$ be $\mathcal{B}_\varepsilon[x] = \{x' \in \mathcal{X} \mid d_{\inf}(x,x') \le \varepsilon\}$, and the function space be $\mathcal{F}$. Given a dataset $S = \{(x_i, y_i)\}_{i=1}^n$ where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y} = \{0, 1, \ldots, C-1\}$, the objective function of the standard adversarial training (Madry et al., 2018) is

$$\min_{f_W \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left\{ \max_{\tilde{x} \in \mathcal{B}_\varepsilon[x_i]} \ell\left(f_W(\tilde{x}), y_i\right) \right\}, \tag{1}$$

where $\tilde{x}$ is the adversarial data centered at x within the ε-ball, $f_W(\cdot) : \mathcal{X} \to \mathbb{R}^C$ is a score function with parameters W and $l(\cdot) : \mathbb{R}^C \times \mathcal{Y} \to \mathbb{R}$ is the loss function that is composed of a base loss $\ell_{\mathrm{B}} : \Delta^{C-1} \times \mathcal{Y} \to \mathbb{R}$ (e.g., the cross-entropy loss) and an inverse link function $\ell_{\mathrm{L}} : \mathbb{R}^C \to \Delta^{C-1}$ (e.g., the soft-max activation). Here Δ^{C-1} is the corresponding probability simplex. In other words, $\ell(f(\cdot), y) = \ell_{\mathrm{B}}(\ell_{\mathrm{L}}(f(\cdot)), y)$. Denote $x^{(0)}$ as the starting point and $\alpha > 0$ as the step size, standard AT generates the most adversarial data by Projected Gradient Descent (PGD) as follows:

$$x^{(t+1)} = \Pi_{\mathcal{B}[x^{(0)}]}\left(x^{(t)} + \alpha \operatorname{sign}\left(\nabla_{x^{(t)}} \ell\left(f_W\left(x^{(t)}\right), y\right)\right)\right), \forall t \ge 0, \tag{2}$$

until a certain stopping criterion is satisfied to get the adversarial data $\tilde{x}$. Π is the projection operator. It then minimizes the classification loss on $\tilde{x}$, which is agnostic to NAs.

TRADES. To trade off natural and robust errors, Zhang et al. (2019b) trained a model on both natural and adversarial data and changed the min-max formulation as follows:

$$\min_{f_W \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left\{ \ell\left(f_W(x_i), y_i\right) + \beta \ell_{\mathrm{KL}}\left(f_W(\tilde{x}_i), f_W(x_i)\right) \right\}, \tag{3}$$

where ℓ_{KL} is the Kullback-Leibler loss. β is a regularization parameter that controls the trade-off between standard accuracy and robustness. When β increases, standard accuracy will decease while robustness will increase, and vice visa. Meanwhile, the adversarial examples are generated by

$$\tilde{x}_i = \arg \max_{\tilde{x} \in \mathcal{B}_\varepsilon[x_i]} \ell_{KL}(f_W(\tilde{x}), f_W(x)). \tag{4}$$

Friendly adversarial training. Friendly AT (Zhang et al., 2020b) is a novel formulation of adversarial training that searches for least adversarial data (i.e., friendly adversarial data) minimizing the inner loss, among the adversarial data that are confidently misclassified. It is easy to implement by just stopping the most adversarial data searching algorithms such as PGD (projected gradient descent) early. The outer minimization still follows Eq. (1). However, instead of generating adversarial data via inner maximization, friendly AT generates $\tilde{x}_i$ as follows:

$$\begin{aligned} \tilde{x}_i = & \arg\min_{\tilde{x} \in \mathcal{B}_\varepsilon[x_i]} \ell\left(f_W(\tilde{x}), y_i\right) \\ & \text{s.t. } \ell\left(f_W(\tilde{x}), y_i\right) - \min_{y \in \mathcal{Y}} \ell(f(\tilde{x}), y) \ge \rho, \end{aligned} \tag{5}$$

where there is a constraint on the margin of loss values ρ (i.e., the misclassification confidence). This constraint firstly ensures $\tilde{x}$ is misclassified and secondly ensures for $\tilde{x}$ the wrong prediction is better than the desired prediction y_i by at least ρ in terms of the loss value.

There are other AT styles, such as misclassification-aware AT (Wang et al., 2020b) and Fast AT (Wong et al., 2020).

3.2. Diverse-Structured Network

The overview of our DS-Net is demonstrated in Fig. 1, which starts with a stem layer (e.g., a convolutional layer for images) for feature transformation. It then stacks K sequential robust modules and ends with an average pooling layer

and a fully connected layer. Each module has N atomic blocks and two sets of variables for optimization, namely the attention weights w and the block parameters W. We denote the attention weight for the j-th atomic block at the i-th module as w_i^j, which is randomly initialized before training. The feature transformation function of the j-th block at the i-th module is denoted as $f_{W_i^j}(\cdot)$ with the parameter W_i^j. During AT, DS-Net alternates between adversarial data generation (with the attention weights fixed) and classification loss minimization. For convenience, the following contents are described in the context of standard AT.

Forward propagation. Denote x_i as the output feature of the $(i-1)^{\text{th}}$ module and $g(\cdot)$ as the first stem layer, given an input x or its adversarial counterpart $\tilde{x}$, the forward propagation of DS-Net is formulated as follows:

$$x_{i+1} = \sum_{j=1}^{N} w_i^j \cdot f_{W_i^j}(x_i), x_1 = g(\tilde{x}), \tag{6}$$

where the output of each module is calculated as the weighted sum of outputs of different atomic blocks. Each atomic block in a robust module has the same number of input and output channels.

Backward propagation. During the backward propagation to generate adversarial examples, DS-Net fixes attention weights w and uses PGD to generate adversarial data as

$$x^{(t+1)} = \Pi_{\mathcal{B}[x^{(0)}]}\left(x^{(t)} + \alpha \operatorname{sign}\left(\nabla_{x^{(t)}} \ell\left(f_W\left(x^{(t)}\right), y, w\right)\right)\right), \forall t \geq 0, \tag{7}$$

which is similar to Eq. (2) but with a set of attention weights.

During classification loss minimization, the attention weights w and the atomic block parameters W are optimized by Stochastic Gradient Descent (SGD) as:

$$\begin{aligned} w' &= w - \alpha_w \nabla_w \ell\left(f_W(\tilde{x}), y_i, w\right), \\ W' &= W - \alpha_W \nabla_W \ell\left(f_W(\tilde{x}), y_i, w\right), \end{aligned} \tag{8}$$

where $\alpha_w, \alpha_W > 0$ are the learning rate for the two set of variables.

Under standard AT, the minimax formulation is changed as

$$\min_{w, f_W \in \mathcal{F}} \frac{1}{n} \sum_{i=1}^{n} \left\{ \max_{\tilde{x} \in \mathcal{B}_\varepsilon[x_i]} \ell\left(f_W(\tilde{x}), y_i, w\right) \right\}, \tag{9}$$

where f_W means DS-Net with the full set of atomic block parameters W. W is simultaneously optimized with the block weights w by the classification loss on the generated adversarial data. Therefore, DS-Net is able to automatically learn to weight different atomic blocks so as to improve architecture exploration and diversity. The training and evaluation outline of DS-Net is presented in Algorithm 1.

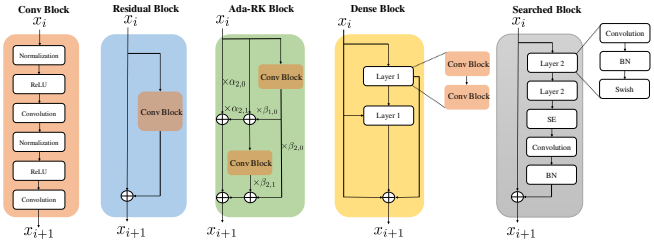

Figure 2. The atomic blocks (except for the Conv block) in DS-Net. SE means the Squeeze-and-Excitation Layer (Hu et al., 2018). Swish is an activation function (Ramachandran et al., 2018).

Algorithm 1 Diverse-Structured Network.

Input: input data $x \in \mathcal{X}$ with label $y \in \mathcal{Y}$, model f_W with block parameters W, loss function ℓ, maximum PGD steps K, perturbation bound ε, step size α, and randomly initialized attention weights w.
Output: learned model f_W and attention weights w.
while *not eval* **do**
  Step 1: Fix w and W, generate $\tilde{x}$ by Eq. (7).
  Step 2: Update w and W by Eq. (8).
end
while *eval* **do**
  Step 3: Fix w and W, generate $\tilde{x}$ by Eq. (7).
  Step 4: Calculate output by Eq. (6) and report accuracy.
end

3.3. Predefined Atomic Blocks

DS-Net allows for a flexible choice of the atomic blocks. We implement DS-Net by using four powerful atomic blocks in Fig. 2, which are either human-designed, such as the residual block (He et al., 2016), dense block (Huang et al., 2017) and Adaptive Runge Kutta (Ada-RK) block (Kim et al., 2020), or searched-based block (Tan et al., 2019). Most blocks have been theoretically or empirically validated to improve ST instead of AT. Although Ada-RK aims at AT, its generalization ability is not satisfactory (Kim et al., 2020). To trade off robustness and generalization, we simultaneously leverage four predefined atomic blocks via the learnable attention weights. Note that there are other potential candidates except the above four predefined blocks, which is a promising future work beyond the scope of our study.

Importantly, using atomic blocks avoids costly structure search for AT while exploiting powerful architectures in literature. Besides, building a robust model by these blocks is easier while retaining sufficient diversity, which is important for AT.

3.4. Robustness Analysis

We provide a robustness analysis of our DS-Net. Previous works (Weng et al., 2018; Hein & Andriushchenko, 2017) usually connected Lipschitz smoothness w.r.t. the input with

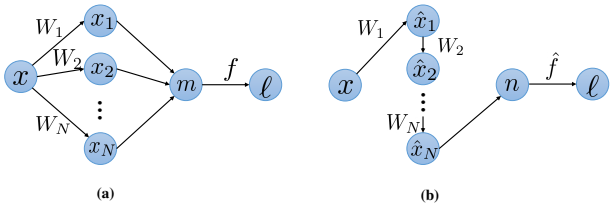

Figure 3. Two different architectures. (a) The robust module for DS-Net. (b) The common network architecture, which covers many human-designed models and searched models. $x_i, \hat{x}_i$ denote the intermediate outputs (atomic block outputs for DS-Net). $f, \hat{f}$ denote the training objective functions. m, n denote the outputs.

network robustness, which suggests that a small input perturbation will not lead to large change of the output. Formally, we give its definition.

Definition 3.1 (Paulavičius & Žilinskas, 2006) Given a model f and a perturbation δ within the ε-ball of the input x. The Lipschitz smoothness of f is represented as

$$|f(x) - f(x+\delta)| \leq L_f \|\delta\|_p \leq L\varepsilon, \tag{10}$$

where $p \geq 0$ is the norm of interest and L_f is the Lipschitz constant. Thus, a robust model holds a small value of L_f.

We fist provide Lemma 3.2 that decomposes the global Lipschitz constant into the value for each atomic block and then propose our main proposition (Proposition 3.3).

Lemma 3.2 *Denote the attention weight and the Lipschitz constant of the j-th block at the i-th module for DS-Net f as w_i^j and L_i^j, and the Lipschitz constant of the j-th block in the common network architecture $\hat{f}$ as L_j. If the number of layers and atomic blocks in DS-Net are K and N, the Lipschitz constants can be decomposed as*

$$L_f = \prod_{i=1}^{K} \sum_{j=1}^{N} w_i^j \cdot L_i^j, \quad L_{\hat{f}} = \prod_{j=1}^{NK} L_j. \tag{11}$$

Proposition 3.3 *Denote the Lipschitz constants for DS-Net with learnable or fixed attention weights as L_f and L'_f, and those of the common network architectures as $L_{\hat{f}}$, we get $L_f \leq L_{\hat{f}}$ and $L_f \leq L'_f$.*

Remark: From Proposition 3.3, we conclude two results: 1) A common network architecture $\hat{f}(\cdot)$ (Fig. 3(b)) with the same number of parameters as DS-Net (Fig. 3(a)) always holds a larger Lipschitz constant $L_{\hat{f}}$. 2) The Lipschitz constant L_f of DS-Net with learnable attention weights is smaller than that of DS-Net with an arbitrary fixed set of attention weights. The proof can be found in Appendix A.

3.5. Convergence Analysis

We provide a convergence analysis of DS-Net for solving the min-max optimization problem in Eq. (9). Due to the nonlinearities in DNNs such as ReLU (Nair & Hinton, 2010) and pooling operations, the exact assumptions of Danskin's theorem (Danskin, 2012) do not hold. Nonetheless, since adversarial training only computes approximate maximizers of the inner problem, we can still provide a theoretical guarantee of convergence. The several necessary assumptions for the analysis are given as follows.

Assumption 3.4 *Let L_{WW}, L_{Wx}, L_{xW} be positive constants, the function $f_W(x,y)$ satisfies the gradient Lipschitz conditions as follows*

$$\begin{aligned}
&\sup_x \|\nabla_W f_W(x,y) - \nabla_W f_{W'}(x,y)\|_2 \leq L_{WW} \|W - W'\|_2; \\
&\sup_W \|\nabla_W f_W(x,y) - \nabla_W f_W(x',y)\|_2 \leq L_{Wx} \|x - x'\|_2; \\
&\sup_x \|\nabla_x f_W(x,y) - \nabla_x f_{W'}(x,y)\|_2 \leq L_{xW} \|W - W'\|_2.
\end{aligned} \tag{12}$$

Assumption 3.4 requires that the loss function satisfies the Lipschitzian smoothness conditions. Despite the non-smoothness of the ReLU activation function, recent studies (Allen-Zhu et al., 2019; Du et al., 2019; Cao & Gu, 2019) justify the loss function of overparamterized networks are semi-smooth. Thus this assumption is satisfied.

Assumption 3.5 *The variance of the stochastic gradient of $f_W(x)$ is bounded by a constant $\sigma^2 > 0$ as*

$$\mathbb{E}\left[\|g(x_k, \xi_k) - \nabla f_W(x_k)\|^2\right] \leq \sigma^2, \tag{13}$$

where $g(x_k, \xi_k)$ is an unbiased estimator of $\nabla f_W(x_k)$ as $\mathbb{E}[g(x_k, \xi_k)] = \nabla f_W(x_k)$. $\xi_k, k \geq 1$ are random variables.

Assumption 3.5 is commonly used in stochastic gradient based optimization algorithms (Wang et al., 2019b).

Then we introduce the convergence analysis of non-convex optimization with the randomized stochastic gradient method (Ghadimi & Lan, 2013) as follows:

Theorem 3.6 *(Ghadimi & Lan, 2013) Suppose the technical assumptions 3.4 and 3.5 hold. Let f be L-smooth and non-convex function and f^* be the optimal value of the optimization problem (9). Given repeated, independent accesses to stochastic gradients with variance bound σ^2, let SGD start with initial network parameters W_0, iterations $N > 0$ and step size $\gamma_k < \frac{1}{L}$, then it converges to the following point by randomly choosing W_k as the final output W_R with probability $\frac{\gamma_k}{H}$. For $H = \sum_{k=1}^{N} \gamma_k$:*

$$\mathbb{E}\left[\|\nabla f_{W_R}(x)\|^2\right] \leq \frac{2\left(f_{W_0}(x) - f^*\right)}{H} + \frac{L\sigma^2}{H} \sum_{k=1}^{N} \gamma_k^2, \tag{14}$$

where f is the robust network in our case.

From this theorem, we can see that the convergence speed and stability of the optimization problem heavily depend on the Lipschitz smoothness L and the gradient variance σ^2

given the fixed number of iterations N and SGD step size (i.e., learning rate) γ_k.

In the following theorem, we explain that DS-Net has a smaller Lipschitz smoothness constant and gradient variance than common NAs. We also conduct an empirical analysis on these two factors to support our claim (Section 4.7).

Due to the complexity of global Lipschitz smoothness, we instead use the block-wise Lipschitz smoothness (Beck & Tetruashvili, 2013) in the following. Consider two architectures in Fig. 3 with the same number of parameters, then we have the following theorem.

Theorem 3.7 *Following Fig. 3, let λ_i be the largest eigenvalue of the network parameters $W_i, i = 1,\dots,N$. $f, \hat{f}$ are the objective functions for DS-Net and common network architectures, respectively. For any two possible assignments W_i^1, W_i^2 of W_i, the block-wise Lipschitz smoothness w.r.t. the network parameters and the gradient variance of the common network architectures are represented as*

$$\begin{aligned} \frac{\|\nabla_{W_i^1}\hat{f} - \nabla_{W_i^2}\hat{f}\|}{\|W_i^1 - W_i^2\|} &\leq L_i \cdot \prod_{k=1}^{i-1}\lambda_k, \\ \mathbb{E}\|\nabla_{W_i}\hat{f} - \mathbb{E}\nabla_{W_i}\hat{f}\|^2 &\leq N\sum_{k=i}^{N}\left(\frac{\sigma_k}{\lambda_i}\prod_{j=1}^{k}\lambda_j\right)^2, \end{aligned} \tag{15}$$

by assuming the two properties of DS-Net satisfy: $\|\nabla_{W_i^1} f - \nabla_{W_i^2} f\| \leq L_i \left\|W_i^1 - W_i^2\right\|$, $\mathbb{E}\left\|\nabla_{W_i} f - \mathbb{E}\nabla_{W_i} f\right\|^2 \leq \sigma_i^2$. N is the number of intermediate layers or atomic blocks.

Remark: From Theorem 3.7, we conclude two results: 1) The common network architectures hold a Lipschitz smoothness constant, namely, $\prod_{k=1}^{i-1}\lambda_k$ times that of DS-Net. Note most of the largest eigenvalues of neural networks are bigger than 1 to prevent vanishing gradients (Pascanu et al., 2013). Therefore, the convergence of Problem (9) is slower than that of DS-Net. 2) The bound of the gradient variance in common network architectures is scaled up by the largest eigenvalue of network parameters and the network depth, which hurts the convergence speed and stability of AT. The proof can be found in Appendix B, which is adapted from Shu et al. (2020). Overall, Theorem 3.7 tells us DS-Net (Fig. 3 (a)) converges faster and more stably than common network architectures (Fig. 3 (b)).

4. Experiments and Results

In this section, we present empirical evidence to validate DS-Net on benchmarks with three AT styles. The code is available at https://github.com/d12306/dsnet.

4.1. Experimental Setting

We evaluated DS-Net on CIFAR-10 and SVHN using: Projected Gradient Descent (PGD) (Madry et al., 2018), Fast Gradient Sign Method (FGSM) (Goodfellow et al., 2015), Carlini & Wagner (C&W) (Carlini & Wagner, 2017) and AutoAttack (AA) (Croce & Hein, 2020). We compared with human-designed models, such as ResNet (He et al., 2016), WideResNet (Zagoruyko & Komodakis, 2016), IE-skips (Li et al., 2020a), AdaRK-Net (Kim et al., 2020) and SAT (Xie et al., 2020a). We also compared with searched NAs for AT, i.e., RobNet (Guo et al., 2020). We used three training styles, i.e., AT (Madry et al., 2018), TRADES (Zhang et al., 2019b) and MART (Wang et al., 2020b).

For CIFAR-10, during training, we set the perturbation bound ε to 0.031 and step size α to 0.007 with 10 steps. We used SGD optimizer with a momentum of 0.9 and weight decay of 5e-4. The initial learning rate is 0.1. We trained for 120 epochs for standard AT and the learning rate is multiplied by 0.1 and 0.01 at epoch 60 and 90. For TRADES, we trained for 85 epochs and the learning rate is multiplied by 0.1 at epoch 75. We tested the performance when the model is trained with regularization factor $\beta = 1$ and $\beta = 6$. For MART, we trained for 90 epochs and the learning rate is multiplied by 0.1 at epoch 60. We set $\beta = 6$. The batch size is set to 128. For SVHN, the step size is set to 0.003 with $\varepsilon = 0.031$. The training epochs including the epoch for learning rate decay is reduced by 20 for AT, TRADES and MART. We trained on one Tesla V100 and used mixed-precision acceleration by apex at O1 optimization level[1]. We select all models 1 epoch after the 1st learning rate decay point following Rice et al. (2020) because robust overfitting also happens for DS-Net. We have tried to use 1,000 images from the training set as validation set to determine the stopping point, which aligns with our selection point.

We used Adam optimizer (Kingma & Ba, 2014) with a learning rate of 1e-3 and a weight decay of 1e-3 to optimize the attention weights, which is then normalized by softmax function. The comparison with using other optimizers is shown in Appendix E. We set the number of layers to 15 and the initial channel number to 20. We used two residual layers at the $\frac{1}{3}$ and $\frac{2}{3}$ of the total depth of the DS-Net to increase the channels by a factor of k and 2, respectively. Meanwhile, the spatial size of the feature map is reduced by a half. We set $k = 4/6$ and obtain a small and large DS-Net in our experiments, denoted as DS-Net-4/6-softmax.

The evaluation ε is set to 0.031. We used PGD attack with 20 steps and CW attack with 30 steps. The step size is set to 0.031 for FGSM attack. For non-FGSM attack, we set the step size to 0.003 on TRADES while the evaluation step size on AT and MART is 0.008. We reported the best accuracy for comparison. Due to the complexity, we reported the accuracy of AA by randomly sampling 12.5% of the test set. Each experiment is repeated by 3 times with three random seeds. The results are averaged for comparisons.

[1] https://github.com/NVIDIA/apex

Table 2. Test accuracy on CIFAR-10 and SVHN using AT (Madry et al., 2018). † means the results by our implementation under the same setting. AA denotes the results of AutoAttack (Croce & Hein, 2020). The perturbation bound ϵ is 0.031. DS-Net-4-softmax means the scale factor is 4 and block weights are normalized by the softmax activation. Improv.(%) is calculated by comparing with the best baseline.

Defense Architecture	Param (M)	Natural	FGSM	PGD-20	C&W∞	AA
CIFAR-10						
ResNet-50 (He et al., 2016)†	23.52	83.83±0.190	54.76±0.229	48.07±0.222	47.77±0.365	44.98±0.237
WRN-34-10 (Zagoruyko & Komodakis, 2016)†	46.16	86.32±0.317	64.84±0.118	51.95±0.291	50.65±0.339	50.01±0.284
SAT-ResNet-50 (Xie et al., 2020a)†	23.52	73.59±0.164	57.50±0.287	48.44±0.105	46.56±0.291	44.11±0.370
SAT-WRN-34-10 (Xie et al., 2020a)†	46.16	78.03±0.298	60.73±0.305	49.54±0.311	49.43±0.042	46.27±0.163
IE-ResNet-50 (Li et al., 2020a)†	22.41	84.49±0.111	55.00±0.229	48.31±0.321	48.04±0.392	43.27±0.138
IE-WRN-34-10 (Li et al., 2020a)†	48.24	84.23±0.200	63.28±0.222	52.61±0.316	49.36±0.501	51.24±0.251
AdaRK-Net (Kim et al., 2020)†	23.61	80.42±0.124	57.23±0.218	51.37±0.411	49.27±0.228	45.11±0.260
RobNet-large-v2 (Guo et al., 2020)†	33.42	84.39±0.129	59.21±0.311	52.54±0.371	51.28±0.212	49.22±0.138
DS-Net-4-softmax (ours)	20.78	85.39±0.216	66.71±0.186	**54.14**±0.100	52.18±0.137	49.98±0.199
DS-Net-6-softmax (ours)	46.35	**86.76**±0.125	**67.03**±0.372	53.59±0.211	**53.28**±0.174	**51.48**±0.191
Improv.(%)	-	0.51%	3.38%	2.91%	3.90%	0.47%
SVHN						
ResNet-50 (He et al., 2016)†	23.52	90.02±0.213	69.03±0.233	47.23±0.177	49.69±0.186	44.11±0.029
WRN-34-10 (Zagoruyko & Komodakis, 2016)†	46.16	94.26±0.175	75.15±0.310	48.57±0.163	50.08±0.271	45.38±0.124
DS-Net-4-softmax (ours)	20.78	95.53±0.172	**78.50**±0.278	49.53±0.301	48.73±0.101	46.21±0.222
DS-Net-6-softmax (ours)	46.35	**95.96**±0.211	75.80±0.170	**50.89**±0.235	**50.12**±0.304	**48.09**±0.258
Improv.(%)	-	1.80%	4.46%	4.78%	0.08%	5.97%

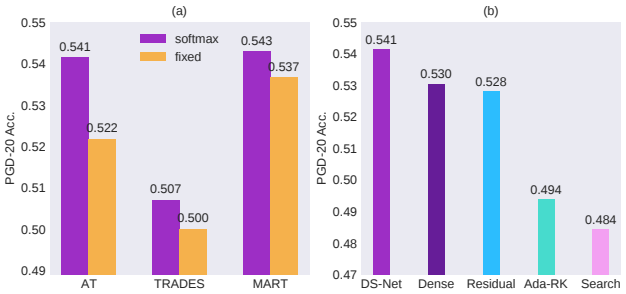

Figure 4. The results of block ensemble variants of DS-Net (under PGD-20 attack). (a) DS-Net with fixed and uniform attention weights. $\beta = 1$ is used for TRADES. (b) DS-Net with the same atomic blocks and uniform ensemble. Standard AT is used.

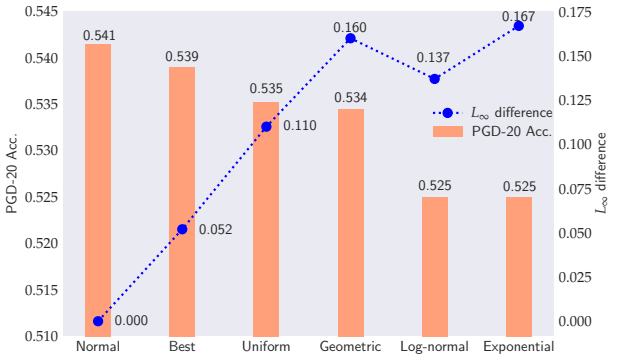

Figure 5. Comparative robustness of our DS-Net with different initializations (evaluated by PGD-20 attack).

4.2. Results on CIFAR-10 and SVHN

We presented the results of AT and MART in Tabs. 2 and 3. Results of TRADES are in Appendix C. We made several observations. First, the robustness of DS-Net is consistently better than baselines, which achieves promising results with a much smaller amount of parameters, e.g., 54.14% under PGD-20 attack with only 20.78M parameters compared to 51.95% for WRN-34-10 (46.16M) and 52.54% for RobNet (33.42M) using AT. Second, if we increase the amount of parameters to the same level of WRN-34-10 by setting the factor $k = 6$, DS-Net further improves its robustness (53.28% under CW attack). Meanwhile, DS-Net generalizes well (with a standard accuracy of 86.76%) and also performs well in terms of ensembles of white-box and black-box attacks (see AutoAttack). Third, DS-Net shows its effectiveness across different datasets and training styles, which provides better robustness and generalization ability.

4.3. Comparison with Block Ensemble

We compared the robustness of DS-Net with two variants: 1) The attention weights are uniformly distributed among different atomic blocks and fixed during training. 2) The four atomic blocks are the same whose outputs are uniformly ensembled. We compared them with DS-Net on CIFAR-10 with factor $k = 4$, which is shown in Fig. 4.

The above figure highlights two factors—a) Learnability of attention weights and (b) The diversity of atomic blocks—matter for network robustness in DS-Net. Therefore, the improvement of DS-Net does not solely come from a simple ensemble. Meanwhile, we found that the highest standard accuracy of these variants are lower than DS-Net, such as 85.31% vs. 87.89% for fixed attention weights and learnable weights in DS-Net on TRADES, which justifies that these two factors are also important for generalization.

4.4. Sensitivity to Weight Initialization

We investigated the sensitivity of DS-Net to the initialization of the attention weights. Note the results in Tabs. 2 and 3 are reported using the normal distribution ($\mu = 0, \sigma^2$=1), we further tested the uniform distribution (within $[0, 1]$), log-normal distribution ($\mu = 0, \sigma^2 = 1$), exponential distribution

Table 3. Evaluations (test accuracy) of deep models on CIFAR-10 and SVHN dataset using MART (Wang et al., 2020b). † means the results by our implementation. The perturbation bound ϵ is set to 0.031 for each architecture. The regularization factor β is set to 6.

Defense Architecture	Param (M)	Natural	FGSM	PGD-20	C&W$_\infty$	AA
CIFAR-10						
RobNet-large-v2 (Guo et al., 2020)†	33.42	80.23±0.129	60.23±0.203	51.07±0.290	48.37±0.365	48.14±0.317
WRN-34-10 (Zagoruyko & Komodakis, 2016)†	46.16	78.59±0.221	62.50±0.355	52.26±0.409	49.75±0.517	49.96±0.531
IE-WRN-34-10 (Li et al., 2020a)†	48.24	81.33±0.127	62.29±0.116	51.99±0.244	49.40±0.142	50.07±0.246
DS-Net-4-softmax(ours)	20.76	79.51±0.137	63.03±0.241	54.29±0.376	50.25±0.229	49.79±0.256
DS-Net-6-softmax(ours)	46.35	**81.64**±0.229	**66.40**±0.173	**55.23**±0.168	**51.48**±0.291	**52.74**±0.096
Improv.(%)	-	0.38%	6.09%	5.68%	3.48%	5.44%
SVHN						
WRN-34-10 (Zagoruyko & Komodakis, 2016)†	46.16	92.15±0.279	74.57±0.160	52.96±0.384	47.03±0.100	49.88±0.103
DS-Net-4-softmax (ours)	20.78	92.39±0.172	73.88±0.263	**56.08**±0.326	48.00±0.298	**51.39**±0.206
DS-Net-6-softmax (ours)	46.35	**93.77**±0.272	**76.23**±0.165	55.00±0.126	**48.84**±0.179	50.43±0.312
Improv.(%)	-	1.76%	2.23%	5.89%	3.85%	3.03%

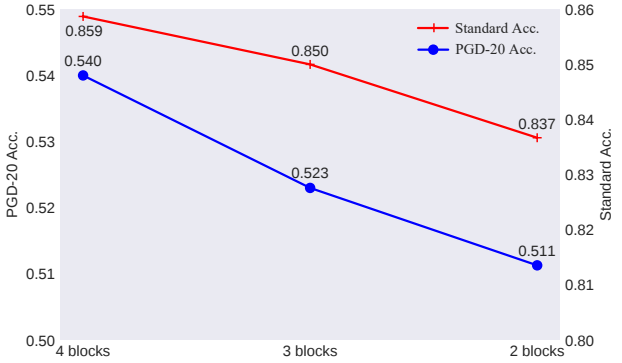

Figure 6. The performance of DS-Net with atomic block space reduction. Results are reported using seed 0.

$\lambda = 1$, geometric distribution ($p = 0.5$). We also used the optimized attention weights after training for initialization. We compared these initializations using standard AT on CIFAR-10 with factor $k = 4$, which are shown in Fig. 5.

From the above figure, DS-Net is not sensitive to different initializations, where the largest accuracy drop is $\leq 2\%$. The optimized weight for initialization obtains the closet performance to the original one, which illustrates its superiority. Besides, we compared the l_∞ difference between the learned weights initialized with the normal distribution and the others, which shows a similar trend to the PGD-20 Acc.

4.5. Effect of Reducing Atomic Block Space

To observe whether the number of atomic blocks matters to model performance, we trained our DS-Net under 2 and 3 atomic blocks for each robust module. Note that sampling 2 and 3 atomic blocks from 4 blocks has 6 and 4 options and we reported the average results. To ensure a fair comparison, we kept the same level of network parameters by changing the initial channel number across different models. We compared them with DS-Net-4-softmax using standard AT on CIFAR-10 with factor $k = 4$, which are shown in Fig. 6.

From Fig. 6, the performance of DS-Net slightly decreases in terms of robustness and generalization ability with smaller atomic block space, which shows a carefully-designed block space with higher diversity is beneficial. A principled approach to block selection is a promising future work.

4.6. Learned Attention Weight Visualization

To gain some insights on the location sensitivity of different atomic blocks, we visualized the learned attention weights from different layers in Fig. 8. We made several observations. First, the weight of atomic blocks is balanced in the former layers of DS-Net without obvious dominance, which implies that AT prefers a diverse structure. Second, the weight of residual block increases in the last layers, which shows DS-Net tends to tends to use the cleaner feature representations by favoring residual blocks in the later layers for classification. This phenomenon happens because the error of features learned by the early layers accumulates less than that learned by the later layers. Third, densely connected modules are more favorable in robust models. For instance, DS-Net-6-softmax pays more attention to the dense block compared to DS-Net-4-softmax, and DS-Net-6-softmax under TRADES ($\beta = 6$) gives more weights to dense block than DS-Net-6-softmax under TRADES ($\beta = 1$). Such findings align well with Guo et al. (2020). The trends may guide us to design different blocks for different layers of a robust model.

4.7. Empirical Convergence Analysis

We showed in Theorem 3.6 that Lipschitz smoothness and gradient variance are important for convergence. In this section, we empirically verified these two properties of DS-Net. Due to the substantial budget for calculating the Hessian matrix of the objective function in order to measure the global Lipschitz smoothness (Nesterov, 2004), we followed (Li et al., 2018) and used the loss landscape to measure the local smoothness of models, which is visualized as $f(\alpha, \beta) = L(W^* + \alpha\delta + \beta\eta)$ (δ, η are two random directions and W^* is the center point in the network parameter space). We compared the loss landscape among four models in Fig. 7 (a), and found that DS-Net empirically smooths the loss landscape around the optimized parameters, which makes the convergence faster and more stable.

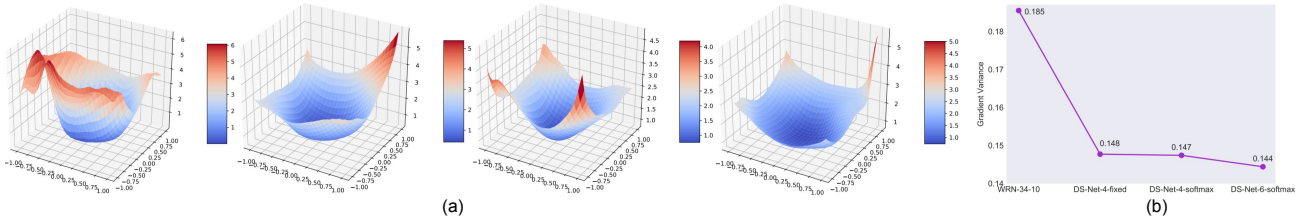

Figure 7. Empirical convergence analysis under TRADES ($\beta = 6$). (a) 3D loss landscape of four models, i.e., WRN-34-10, DS-Net-4-fixed, DS-Net-4-softmax and DS-Net-6-softmax from left to right where "fixed" means we use 0.25 as the weight for each atomic block. The height of the surface indicates the loss value. (b) Gradient variance for the same set of models.

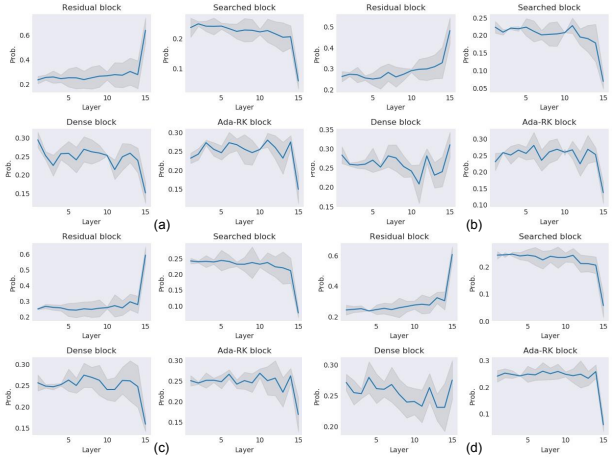

Figure 8. Visualization of the learned attention weights with variance in shaded area. (a) DS-Net-4-softmax under AT. (b) DS-Net-6-softmax under AT. (c) DS-Net-6-softmax under TRADES ($\beta = 1$). (d) DS-Net-6-softmax under TRADES ($\beta = 6$).

We computed the gradient variance of DS-Net by its definition $\mathbb{E}\|\nabla_{W_i} f - \mathbb{E}\nabla_{W_i} f\|^2$. To reduce computational cost, following Shu et al. (2020), the gradients over batches are regarded as the full gradients and the expected gradient is calculated by averaging over batch gradients. The variance is calculated w.r.t. the last fully connected layer. The comparison among the same set of models is shown in Fig. 7(b), where DS-Net holds a smaller gradient variance. Learnable attention weights and more network parameters are beneficial to convergence speed and stability.

4.8. Results after Block Pruning

To observe whether block pruning in search-based methods (Dong et al., 2020a) benefits DS-Net, we tested the performance of DS-Net by selecting 1,2,3 atomic blocks with higher probabilities after training, re-normalizing their weights and retraining the model from scratch by AT (The weights are fixed). We also tested 4 blocks with their optimized attention weights. To ensure a fair comparison, we kept the same level of network parameters by changing the initial channel number across different models. We conducted experiments using standard AT on CIFAR-10 with factor $k = 4$, which are shown in Tab. 4.

Table 4. Model robustness and generalization ability with different number of pruned blocks. The standard deviations (Std.) are shown below the mean row (Acc.).

Selected Blocks	1	2	3	4	Ours
PGD-20 Acc. (%)	42.03	51.88	50.70	52.19	**54.14**
Std.	0.339	0.214	0.213	0.291	0.100
Standard Acc. (%)	78.05	84.22	84.30	83.75	**85.39**
Std.	0.172	0.238	0.176	0.118	0.216

Tab. 4 shows that block pruning is not suitable for DS-Net, since attention weights and block parameters are co-adapted together. Therefore, discarding parameters that cooperates well with attention weights will lead to the accuracy drop. Additionally, such block pruning reduces the structure diversity during retraining, which may also hurt the model robustness and generalization ability.

5. Conclusion

Orthogonal to studies from the view of objective and optimizer for AT, we focus on NAs and propose a Diverse-Structured Network (DS-Net) to improve model robustness. DS-Net trades off exploring diverse structures and exploiting the best structures. Specifically, it learns a set of attention weights over predefined atomic blocks, where attention weights are jointly optimized with network parameters by the robust training objective that encourages structure exploration. We theoretically demonstrate the advantage of DS-Net in terms of robustness and convergence, and empirically justify our DS-Net on benchmark datasets. In the future, we will improve DS-Net by studying different combination of atomic blocks to further improve model robustness.

Acknowledgements

XFD and BH were supported by HKBU Tier-1 Start-up Grant and HKBU CSD Start-up Grant. BH was also supported by the RGC Early Career Scheme No. 22200720, NSFC Young Scientists Fund No. 62006202 and HKBU CSD Departmental Incentive Grant. TLL was supported by Australian Research Council Project DE-190101473. JZ, GN and MS were supported by JST AIP Acceleration Research Grant Number JPMJCR20U3, Japan. MS was also supported by the Institute for AI and Beyond, UTokyo.

References

Abbasi, M., Rajabi, A., Gagné, C., and Bobba, R. B. Toward adversarial robustness by diversity in an ensemble of specialized deep neural networks. In *Canadian Conference on Artificial Intelligence*, 2020.

Allen-Zhu, Z., Li, Y., and Song, Z. A convergence theory for deep learning via over-parameterization. In *ICML*, 2019.

Athalye, A., Carlini, N., and Wagner, D. A. Obfuscated gradients give a false sense of security: Circumventing defenses to adversarial examples. In *ICML*, 2018.

Bai, Y., Zeng, Y., Jiang, Y., Xia, S.-T., Ma, X., and Wang, Y. Improving adversarial robustness via channel-wise activation suppressing. In *ICLR*, 2021.

Beck, A. and Tetruashvili, L. On the convergence of block coordinate descent type methods. *SIAM Journal on Optimization*, 2013.

Bui, A., Le, T., Zhao, H., Montague, P., DeVel, O. Y., Abraham, T., and Phung, D. Q. Improving adversarial robustness by enforcing local and global compactness. In *ECCV*, 2020.

Cai, Q., Liu, C., and Song, D. Curriculum adversarial training. In *IJCAI*, 2018.

Cao, Y. and Gu, Q. A generalization theory of gradient descent for learning over-parameterized deep relu networks. *CoRR*, abs/1902.01384, 2019.

Carlini, N. and Wagner, D. A. Adversarial examples are not easily detected: Bypassing ten detection methods. In *ACM Workshop on Artificial Intelligence and Security*, 2017.

Cazenavette, G., Murdock, C., and Lucey, S. Architectural adversarial robustness: The case for deep pursuit. *CoRR*, abs/2011.14427, 2020.

Chen, H., Zhang, B., Xue, S., Gong, X., Liu, H., Ji, R., and Doermann, D. S. Anti-bandit neural architecture search for model defense. In *ECCV*, 2020.

Cissé, M., Bojanowski, P., Grave, E., Dauphin, Y. N., and Usunier, N. Parseval networks: Improving robustness to adversarial examples. In *ICML*, 2017.

Cohen, J. M., Rosenfeld, E., and Kolter, J. Z. Certified adversarial robustness via randomized smoothing. In *ICML*, 2019.

Croce, F. and Hein, M. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *ICML*, 2020.

Cubuk, E. D., Zoph, B., Schoenholz, S. S., and Le, Q. V. Intriguing properties of adversarial examples. In *ICLR Workshop*, 2018.

Danskin, J. M. The theory of max-min and its application to weapons allocation problems. In *Springer Science & Business Media*, 2012.

Dong, M., Li, Y., Wang, Y., and Xu, C. Adversarially robust neural architectures. *CoRR*, abs/2009.00902, 2020a.

Dong, Y., Deng, Z., Pang, T., Zhu, J., and Su, H. Adversarial distributional training for robust deep learning. In *NeurIPS*, 2020b.

Du, S. S., Lee, J. D., Li, H., Wang, L., and Zhai, X. Gradient descent finds global minima of deep neural networks. In *ICML*, 2019.

Dziugaite, G. K., Ghahramani, Z., and Roy, D. M. A study of the effect of JPG compression on adversarial images. *CoRR*, abs/1608.00853, 2016.

Ghadimi, S. and Lan, G. Stochastic first- and zeroth-order methods for nonconvex stochastic programming. *SIAM Journal on Optimization*, 2013.

Goodfellow, I. J., Shlens, J., and Szegedy, C. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

Guo, M., Yang, Y., Xu, R., Liu, Z., and Lin, D. When NAS meets robustness: In search of robust architectures against adversarial attacks. In *CVPR*, 2020.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *CVPR*, 2016.

Hein, M. and Andriushchenko, M. Formal guarantees on the robustness of a classifier against adversarial manipulation. In *NeurIPS*, 2017.

Hosseini, R., Yang, X., and Xie, P. Dsrna: Differentiable search of robust neural architectures. *CoRR*, abs/2012.06122, 2020.

Hu, J., Shen, L., and Sun, G. Squeeze-and-excitation networks. In *CVPR*, 2018.

Huang, G., Liu, Z., van der Maaten, L., and Weinberger, K. Q. Densely connected convolutional networks. In *CVPR*, 2017.

Kariyappa, S. and Qureshi, M. K. Improving adversarial robustness of ensembles with diversity training. *CoRR*, abs/1901.09981, 2019.

Kim, B., Chudomelka, B., Park, J., Kang, J., Hong, Y., and Kim, H. J. Robust neural networks inspired by strong stability preserving runge-kutta methods. In *ECCV*, 2020.

Kingma, D. and Ba, J. Adam: A method for stochastic optimization. *ICLR*, 2014.

Li, H., Xu, Z., Taylor, G., Studer, C., and Goldstein, T. Visualizing the loss landscape of neural nets. In *NeurIPS*, 2018.

Li, M., He, L., and Lin, Z. Implicit euler skip connections: Enhancing adversarial robustness via numerical stability. In *ICML*, 2020a.

Li, P., Yi, J., Zhou, B., and Zhang, L. Improving the robustness of deep neural networks via adversarial training with triplet loss. In *IJCAI*, 2019.

Li, Y., Dong, M., Wang, Y., and Xu, C. Neural architecture search in A proxy validation loss landscape. In *ICML*, 2020b.

Liu, X., Cheng, M., Zhang, H., and Hsieh, C. Towards robust neural networks via random self-ensemble. In *ECCV*, 2018.

Ma, A., Faghri, F., and Farahmand, A. Adversarial robustness through regularization: A second-order approach. *CoRR*, abs/2004.01832, 2020.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018.

Nair, V. and Hinton, G. E. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

Naseer, M., Khan, S., Hayat, M., Khan, F. S., and Porikli, F. A self-supervised approach for adversarial robustness. In *CVPR*, 2020.

Nesterov, Y. E. *Introductory Lectures on Convex Optimization - A Basic Course*. 2004.

Ning, X., Zhao, J., Li, W., Zhao, T., Yang, H., and Wang, Y. Multi-shot NAS for discovering adversarially robust convolutional neural architectures at targeted capacities. *CoRR*, abs/2012.11835, 2020.

Pang, T., Xu, K., Du, C., Chen, N., and Zhu, J. Improving adversarial robustness via promoting ensemble diversity. In *ICML*, 2019.

Pang, T., Yang, X., Dong, Y., Su, H., and Zhu, J. Bag of tricks for adversarial training. In *ICLR*, 2021.

Pascanu, R., Mikolov, T., and Bengio, Y. On the difficulty of training recurrent neural networks. In *ICML*, 2013.

Paulavičius, R. and Žilinskas, J. Analysis of different norms and corresponding lipschitz constants for global optimization. *Ukio Technologinis ir Ekonominis Vystymas*, 12(4): 301–306, 2006.

Qian, H. and Wegman, M. N. L2-nonexpansive neural networks. In *ICLR*, 2019.

Rahnama, A., Nguyen, A. T., and Raff, E. Robust design of deep neural networks against adversarial attacks based on lyapunov theory. In *CVPR*, 2020.

Ramachandran, P., Zoph, B., and Le, Q. V. Searching for activation functions. In *ICLR Workshop*, 2018.

Rice, L., Wong, E., and Kolter, J. Z. Overfitting in adversarially robust deep learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119, pp. 8093–8104, 2020.

S., V. B. and Babu, R. V. Single-step adversarial training with dropout scheduling. In *CVPR*, 2020.

Shu, Y., Wang, W., and Cai, S. Understanding architectures learnt by cell-based neural architecture search. In *ICLR*, 2020.

Tan, M., Chen, B., Pang, R., Vasudevan, V., Sandler, M., Howard, A., and Le, Q. V. Mnasnet: Platform-aware neural architecture search for mobile. In *CVPR*, 2019.

Tsuzuku, Y., Sato, I., and Sugiyama, M. Lipschitz-Margin training: Scalable certification of perturbation invariance for deep neural networks. In *NeurIPS*, pp. 6541–6550, 2018.

Vargas, D. V. and Kotyan, S. Evolving robust neural architectures to defend from adversarial attacks. *CoRR*, abs/1906.11667, 2019.

Wang, B., Shi, Z., and Osher, S. J. Resnets ensemble via the feynman-kac formalism to improve natural and robust accuracies. In *NeurIPS*, 2019a.

Wang, H. and Yu, C. A direct approach to robust deep learning using adversarial networks. In *ICLR*, 2019.

Wang, H., Chen, T., Gui, S., Hu, T., Liu, J., and Wang, Z. Once-for-all adversarial training: In-situ tradeoff between robustness and accuracy for free. In *NeurIPS*, 2020a.

Wang, Y., Ma, X., Bailey, J., Yi, J., Zhou, B., and Gu, Q. On the convergence and robustness of adversarial training. In *ICML*, 2019b.

Wang, Y., Zou, D., Yi, J., Bailey, J., Ma, X., and Gu, Q. Improving adversarial robustness requires revisiting misclassified examples. In *ICLR*, 2020b.

Weng, T., Zhang, H., Chen, P., Yi, J., Su, D., Gao, Y., Hsieh, C., and Daniel, L. Evaluating the robustness of neural networks: An extreme value theory approach. In *ICLR*, 2018.

Wong, E. and Kolter, J. Z. Provable defenses against adversarial examples via the convex outer adversarial polytope. In *ICML*, 2018.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. In *ICLR*, 2020.

Wu, D., Xia, S., and Wang, Y. Adversarial weight perturbation helps robust generalization. In *NeurIPS*, 2020.

Xie, C., Wang, J., Zhang, Z., Ren, Z., and Yuille, A. L. Mitigating adversarial effects through randomization. In *ICLR*, 2018.

Xie, C., Tan, M., Gong, B., Yuille, A. L., and Le, Q. V. Smooth adversarial training. *CoRR*, abs/2006.14536, 2020a.

Xie, L., Chen, X., Bi, K., Wei, L., Xu, Y., Chen, Z., Wang, L., Xiao, A., Chang, J., Zhang, X., and Tian, Q. Weight-sharing neural architecture search: A battle to shrink the optimization gap. *CoRR*, abs/2008.01475, 2020b.

Yue, Z., Lin, B., Huang, X., and Zhang, Y. Effective, efficient and robust neural architecture search. *CoRR*, abs/2011.09820, 2020.

Zagoruyko, S. and Komodakis, N. Wide residual networks. In *BMVC*, 2016.

Zhai, R., Cai, T., He, D., Dan, C., He, K., Hopcroft, J. E., and Wang, L. Adversarially robust generalization just requires more unlabeled data. *CoRR*, abs/1906.00555, 2019.

Zhang, D., Zhang, T., Lu, Y., Zhu, Z., and Dong, B. You only propagate once: Accelerating adversarial training via maximal principle. In *NeurIPS*, 2019a.

Zhang, H., Yu, Y., Jiao, J., Xing, E. P., Ghaoui, L. E., and Jordan, M. I. Theoretically principled trade-off between robustness and accuracy. In *ICML*, 2019b.

Zhang, H., Chen, H., Xiao, C., Gowal, S., Stanforth, R., Li, B., Boning, D. S., and Hsieh, C. Towards stable and efficient training of verifiably robust neural networks. In *ICLR*, 2020a.

Zhang, J., Xu, X., Han, B., Niu, G., Cui, L., Sugiyama, M., and Kankanhalli, M. Attacks which do not kill training make adversarial learning stronger. In *ICML*, 2020b.

Zhang, Y. and Liang, P. Defending against whitebox adversarial attacks via randomized discretization. In *AISTATS*, 2019.

A. Proof of Proposition 3.3

Before we give the proof for Proposition 3.3, we first illustrate how we obtain Lemma 3.2.

Proof of Lemma 3.2 For the first module f_1 in DS-Net, the Lipschitz smoothness is represented as

$$|f_1(x) - f_1(x+\delta)| \leq \sum_{j=1}^{N} w_1^j \cdot L_1^j \cdot \|\delta\|_p, \tag{16}$$

which is obtained by the definition of Lipschitz smoothness for each atomic block. By composing different layers behind together, we get

$$|f_K(x) - f_K(x+\delta)| \leq \prod_{i=1}^{K} \sum_{j=1}^{N} w_i^j \cdot L_i^j \cdot \|\delta\|_p, \tag{17}$$

Therefore, the Lipschitz constant of DS-Net is decomposed as $L_f = \prod_{i=1}^{K} \sum_{j=1}^{N} w_i^j \cdot L_i^j$. Following the same decomposition process, the Lipschitz constant of the common network architecture is decomposed as $L_{\hat{f}} = \prod_{j=1}^{NK} L_j$ with the same number of network parameters.

Proof of Proposition 3.3 For the Lipschitz constant of the parameterized convolutional layers, we focus on the L_2 bounded perturbations. According to the definition of spectral norm, the Lipschitz constant of these layers is the spectral norm of its weight matrix. Mathematically, we get $L_i^j = \prod_{k=1}^{M} \|W_k\|_2$ where M is the number of convolutional layers in the current block. The spectral norm is also the maximum singular value of W_k. According to Pascanu et al. (2013), we need $\|W_k\|_2 \geq 1$ in order to prevent vanishing gradient problem during adversarial training. Therefore, we get $L_i^j \geq 1$ for all the blocks.

Comparing L_f and $L_{\hat{f}}$ then degenerates to comparing $\sum_{j=1}^{N} w_i^j \cdot L_i^j$ for DS-Net and $\prod_{j=1}^{N} L_j$ for the common network architecture since we can reorder the blocks and do not change the Lipschitz constant. And we have the following comparison results

$$\sum_{j=1}^{N} w_i^j \cdot L_i^j \leq max_j L_i^j \leq \prod_{j=1}^{N} L_j, \tag{18}$$

which obtained by using the fact that $L_i^j \geq 1$. Note that although the perturbation is L_2 bounded, the robustness against L_∞ can be also achieved, as stated by (Qian & Wegman, 2019).

Next, we prove DS-Net with a learnable attention weight is more robust than DS-Net with an arbitrary fixed set of attention weight. According to Cissé et al. (2017); Ma et al. (2020), the robust training objective $\ell(f(x+\delta, y, w))$ can be approximated by

$$\ell(f(x+\delta, y, w)) \approx \ell(f(x, y, w)) + L\varepsilon, \tag{19}$$

which is the standard classification loss on the natural images plus a term that is linearly correlated with the global Lipschitz constant.

Given a fixed number of network parameters, the standard classification loss function for the DS-Net with a learnable or fixed set of attention weights does not vary much. Therefore, adversarial training minimizes the global Lipschitz constant L of DS-Net implicitly. Recall that the global Lipschitz constant L is a function of the attention weight w_i^j and the Lipschitz constant L_i^j for each block, if we assume w_i^j is learnable and adversarial training leads to a global minimization of L, then changing the optimized w_i^j in DS-Net will cause the global Lipschitz constant L to increase, which validates our claim $L_f \leq L'_f$.

B. Proof of Theorem 3.7

The proof of Theorem 3.7 is inspired by Shu et al. (2020).

Proof of Theorem 3.7 Following Fig. 3 in the main paper, we first explain how we obtain the bound of the block-wise Lipschitz constant for the common network architecture given a bounded block-wise Lipschitz constant of DS-Net.

To begin with, the derivative w.r.t. the parameter W_i in DS-Net is calculated as:

$$\nabla_{W_i} f = \nabla_{x_i} f x^T. \tag{20}$$

For the common network architecture, we get $\hat{x}_i = \prod_{k=1}^{i} W_k x$. Similarly, the gradient w.r.t. the parameter matrix W_i is calculated by the chain rule as

$$\begin{aligned}\nabla_{W_i}\hat{f} &= \sum_{k=i}^{N}\left(\prod_{j=i+1}^{k} W_j\right)^T \nabla_{\hat{x}_k}\hat{f}\left(\prod_{j=1}^{i-1} W_j x\right)^T \\ &= \sum_{k=i}^{N}\left(\prod_{j=i+1}^{k} W_j\right)^T \nabla_{\hat{x}_k}\hat{f}x^T\left(\prod_{j=1}^{i-1} W_j\right)^T.\end{aligned} \tag{21}$$

Using the fact that $\nabla_{\hat{x}_k}\hat{f} = \nabla_{x_k} f$, we replace $\nabla_{\hat{x}_k}\hat{f}x^T$ with $\nabla_{W_k} f$ according to Eqn. (20) and we get

$$\nabla_{W_i}\hat{f} = \sum_{k=i}^{N}\left(\prod_{j=i+1}^{k} W_j\right)^T \nabla_{W_k} f\left(\prod_{j=1}^{i-1} W_j\right)^T. \tag{22}$$

To avoid the complexity of using the standard Lipschitz constant of the smoothness for analysis, we explore and compare the block-wise Lipschitz constant (Beck & Tetruashvili, 2013) for DS-Net and the common network architecture. Specifically, we analyze for each parameter matrix W_i while fixing others. Currently, we have the block-wise Lipschitz constant bound for DS-Net, which is $\|\nabla_{W_i^1} f - \nabla_{W_i^2} f\| \leq L_i \left\|W_i^1 - W_i^2\right\|$. W_i^1, W_i^2 are any two possible assignments of W_i.

Denote λ_i as the largest eigenvalue of the parameter matrix W_i, assume we use a 2-norm for the parameter matrix W_i, then we get $\lambda_i = \|W_i\|$ where λ_i is the largest eigenvalue of $\|W_i\|$. The local smoothness w.r.t. the network parameters of the common network architectures is shown as

$$\begin{aligned}\left\|\nabla_{W_i^1}\hat{f} - \nabla_{W_i^2}\hat{f}\right\| &= \left\|\sum_{k=i}^{N}\left(\prod_{j=i+1}^{k} W_j\right)^T \left(\nabla_{W_k^1} f - \nabla_{W_k^2} f\right)\left(\prod_{j=1}^{i-1} W_j\right)^T\right\| \\ &\leq \sum_{k=i}^{N}\left\|\left(\prod_{j=i+1}^{k} W_j\right)^T \left(\nabla_{W_k^1} f - \nabla_{W_k^2} f\right)\left(\prod_{j=1}^{i-1} W_j\right)^T\right\| \\ &\leq \sum_{k=i}^{N}\left(\frac{1}{\lambda_i}\prod_{j=1}^{k}\lambda_j\right) L_k \left\|W_k^1 - W_k^2\right\| \\ &\leq \left(\prod_{j=1}^{i-1}\lambda_j\right) L_i \left\|W_i^1 - W_i^2\right\|.\end{aligned} \tag{23}$$

The first line of Eqn. (23) is obtained based on Eqn. (22) and the fact W_j is the same when we focus on the investigating the block-wise Lipschitz smoothness of W_i. The second line of Eqn. (23) is based on triangle inequality of norm. The third line is obtained by the inequality $\|WV\| \leq \|W\|\|V\|$ and the given block-wise smoothness of our DS-Net. The last line is obtained by using the fact that $W_k^1 = W_k^2$ if $k \neq i$.

Similarly, for the gradient variance bound of the common network architecture, we start from the gradient variance bound for DS-Net as follows

$$\mathbb{E}\left\|\nabla_{W_i} f - \mathbb{E}\nabla_{W_i} f\right\|^2 \leq \sigma_i^2. \tag{24}$$

Given such a bound for DS-Net, the bound for the common network architectures is shown as follows:

$$\begin{aligned}\mathbb{E}\left\|\nabla_{W_i}\hat{f} - \mathbb{E}\nabla_{W_i}\hat{f}\right\|^2 &= \mathbb{E}\left\|\sum_{k=i}^{N}\left(\prod_{j=i+1}^{k} W_j\right)^T \left(\nabla_{W_k} f - \mathbb{E}\nabla_{W_k} f\right)\left(\prod_{j=1}^{i-1} W_j\right)^T\right\|^2 \\ &\leq N\mathbb{E}\sum_{k=i}^{N}\left\|\left(\prod_{j=i+1}^{k} W_j\right)^T \left(\nabla_{W_k} f - \mathbb{E}\nabla_{W_k} f\right)\left(\prod_{j=1}^{i-1} W_j\right)^T\right\|^2 \\ &\leq N\sum_{k=i}^{N}\left(\frac{\sigma_k}{\lambda_i}\prod_{j=1}^{k}\lambda_j\right)^2.\end{aligned} \tag{25}$$

The first line of Eqn. (25) is obtained by using Eqn. (22). The second line of Eqn. (23) is obtained by Cauchy-Schwarz inequality. The last line is obtained based on the inequality $\|WV\| \leq \|W\|\|V\|$ and the bounded gradient variance of DS-Net.

C. Additional Results on TRADES

To further illustrate the effectiveness of DS-Net on different adversarial training styles, we test its robustness and standard accuracy on TRADES with both $\beta = 1$ and $\beta = 6$ in Tab. 5. The detailed experimental setting is described in Section 4.1 of the main paper. Tab. 5 shows a similar trend as the DS-Net trained under standard AT and MART as stated in the main paper.

Table 5. Evaluations (test accuracy) of deep models on CIFAR-10 and SVHN dataset using TRADES (Zhang et al., 2019b). [1] means $\beta = 1.0$ and [2] means $\beta = 6.0$. † means the results by our implementation. The perturbation bound ϵ is set to 0.031 for each architecture.

Defense Architecture	Param (M)	Natural	FGSM	PGD-20	C&W$_\infty$	AA
CIFAR-10						
RobNet-large-v2[1] (Guo et al., 2020)†	33.42	87.90±0.132	57.01±0.258	49.27±0.315	49.00±0.124	46.84±0.130
WRN-34-10[1] (Zagoruyko & Komodakis, 2016)†	46.16	88.07±0.231	56.03±0.120	49.27±0.200	48.98±0.119	46.62±0.288
IE-WRN-34-10[1] (Li et al., 2020a)†	48.24	88.31±0.303	54.32±0.129	50.22±0.100	**50.37**±0.331	48.92±0.138
DS-Net-4-softmax[1](ours)	20.78	87.89±0.176	64.38±0.218	50.70±0.322	46.78±0.303	48.10±0.200
DS-Net-6-softmax[1](ours)	46.35	**88.44**±0.301	**65.00**±0.120	**52.50**±0.286	49.75±0.174	**50.00**±0.166
Improv.(%)	-	0.15%	14.02%	4.54%	-	2.21%
RobNet-large-v2[2] (Guo et al., 2020)†	33.42	81.95±0.119	60.31±0.320	53.21±0.166	50.09±0.300	50.13±0.263
WRN-34-10[2] (Zagoruyko & Komodakis, 2016)†	46.16	83.88±0.110	62.28±0.206	55.49±0.231	53.94±0.158	52.21±0.145
IE-WRN-34-10[2] (Li et al., 2020a)†	48.24	83.23±0.134	61.28±0.209	56.03±0.099	**61.72**±0.201	52.73±0.273
DS-Net-4-softmax[2](ours)	20.78	82.81±0.375	64.69±0.154	54.61±0.272	50.63±0.219	52.02±0.116
DS-Net-6-softmax[2](ours)	46.35	**83.98**±0.177	**66.56**±0.208	**56.87**±0.311	54.12±0.272	**53.33**±0.256
Improv.(%)	-	0.12%	6.87%	1.50%	-	1.14%
SVHN						
WRN-34-10[1] (Zhang et al., 2019b)†	46.16	94.23±0.117	72.76±0.287	52.42±0.300	48.65±0.216	48.86±0.183
DS-Net-4-softmax[1](ours)	20.78	94.77±0.213	72.85±0.340	**55.69**±0.272	48.90±0.136	**51.37**±0.401
DS-Net-6-softmax[1](ours)	46.35	**95.73**±0.197	**76.61**±0.362	54.92±0.351	**49.12**±0.272	51.26±0.228
Improv.(%)	-	1.59%	5.29%	6.24%	0.97%	5.14%
WRN-34-10[2] (Zhang et al., 2019b)†	46.16	91.92±0.223	73.65±0.128	57.46±0.125	50.34±0.231	54.11±0.231
DS-Net-4-softmax[2](ours)	20.78	91.74±0.370	**73.83**±0.414	59.84±0.351	53.92±0.184	56.54±0.306
DS-Net-6-softmax[2](ours)	46.35	**92.54**±0.217	73.04±0.361	**60.54**±0.212	**54.58**±0.153	**56.75**±0.065
Improv.(%)	-	0.67%	0.24%	5.36%	8.42%	4.88%

D. Effect of Weight Decay

To demonstrate the effect of weight decay in adversarial training, we change the weight cay to 3e-4, 4e-4, 6e-4 and 7e-4 and report the average performance in terms of robustness and generalization ability for DS-Net. We conduct experiments using standard AT on CIFAR-10 with factor $k = 4$, which are shown in Tab. 6. The results demonstrate the importance of weight decay in adversarial training (align well with the empirical findings in Pang et al. (2021)), which should be carefully selected.

Table 6. Model robustness and generalization ability with different weight decay. The perturbation bound for evaluation is set to 0.031.

Weight decay	3e-4	4e-4	5e-4	6e-4	7e-4
PGD-20 Acc. (%)	48.28	49.69	**54.14**	52.67	49.69
Standard Acc. (%)	82.50	85.00	**85.39**	84.06	83.75

E. Comparison with using different optimizers for DS-Net

SGD is commonly used in AT literature. We tried other optimizers such as Adam, RMSprop, Adadelta and Adagrad. The PGD-20 accuracy is listed in Tab. 7 for DS-Net-4-softmax on CIFAR-10 (vs. 54.14% by SGD).

Table 7. PGD-20 accuracy of DS-Net trained with different optimizers for block parameters.

Optimizer	Adam	Adadelta	Adagrad	RMSprop
PGD-20 Acc.	40.76%	41.12%	39.28%	37.46%