RobArch: Designing Robust Architectures against Adversarial Attacks

ShengYun Peng[1], Weilin Xu[2], Cory Cornelius[2], Kevin Li[1], Rahul Duggal[1],
Duen Horng Chau[1], and Jason Martin[2]

[1]Georgia Institute of Technology, Atlanta, GA, USA

{speng65,kevin.li,rahulduggal,polo}@gatech.edu

[2] Intel Corporation, Hillsboro, OR, USA

{weilin.xu,cory.cornelius,jason.martin}@intel.com

Abstract

Adversarial Training is the most effective approach for improving the robustness of Deep Neural Networks (DNNs). However, compared to the large body of research in optimizing the adversarial training process, there are few investigations into how architecture components affect robustness, and they rarely constrain model capacity. Thus, it is unclear where robustness precisely comes from. In this work, we present the first large-scale systematic study on the robustness of DNN architecture components under fixed parameter budgets. Through our investigation, we distill 18 actionable robust network design guidelines that empower model developers to gain deep insights. We demonstrate these guidelines' effectiveness by introducing the novel Robust Architecture (RobArch) model that instantiates the guidelines to build a family of top-performing models across parameter capacities against strong adversarial attacks. RobArch achieves the new state-of-the-art AutoAttack accuracy on the RobustBench ImageNet leaderboard.

1. Introduction

Deep Neural Networks (DNNs) are vulnerable to adversarial attacks [6,17,29,32,47]. Many defense methods have been proposed to mitigate this pitfall [2,10,44,50,58,59,64], and among them, Adversarial Training (AT) [36] is the most effective way to defend against adversarial attacks. Compared to the large body of research devoted to improving the loss function [23, 31] and optimizing the AT procedure [14, 54, 64], few studies investigate how architectural components affect robustness despite its importance.

Yet DNN architectures have been dominating generalization improvements [16, 19, 34]. Recent research has started to highlight the potential significant impact architecture choices could have on robustness [13,46], and showed that adjusting widths [55] or depths [26] could robustify a
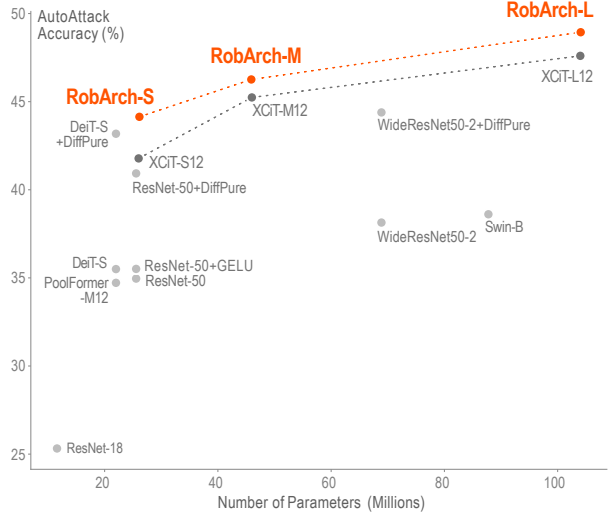

Figure 1. Our RobArch model family outperforms the SOTA XCiT family on RobustBench ImageNet leaderboard [7]. Every RobArch model outperforms its XCiT counterparts at a similar capacity. RobArch-S outperforms ResNet-50 by 9.18 percentage points, and is even more robust than WideResNet50-2 despite having 2.6× fewer parameters. The robustness continues to increase as capacity increases. RobArch-L achieves the new SOTA AA accuracy on RobustBench. Table 3 presents accuracy details.

network.

However, those studies did not constrain the model capacity, making it hard to attribute the robustness gains to those adjustments, because increasing model capacity alone could already improve robustness [26, 36]. Thus, controlling for model capacity while assessing robustness is important, and recent research has provided supporting evidence. For example, despite the popular belief that transformer models might be more robust than CNNs [5, 42], Bai *et al*. [3] demonstrated that Data-efficient image Trans-

1

formers (DeiT) [49] and ResNet [19] with Gaussian Error Linear Unit (GELU) activations [21] attained comparable robustness if the model scales were balanced. Therefore, it remains unclear how these previously studied architectural components precisely affect robustness. Our research filled this critical research gap by making three key contributions:

- **The first large-scale systematic study on the robustness of DNN architecture components.** To the best of our knowledge, our work is the first to comprehensively investigate and compare the robustness impacts of a wide range of architecture components on a large dataset such as ImageNet. Advancing over prior work, we carefully constrain the parameter budget to isolate and hone in on the benefit of each component. Such a systematic study enables us to discover a family of new architectures that outperform state-of-the-art (SOTA) networks. (Figure 1).

- **18 actionable robust network design guidelines.** Our systematic investigation for component robustness, through training over 150 models on ImageNet [12], enables us to distill 18 generalizable, actionable guidelines that empower model developers to gain deep insights and design networks with higher robustness. The guidelines present significant new knowledge and discoveries for our computer vision community. For example, we have discovered (1) deepening a network is more effective than widening it, and there is a sweet spot; (2) specific modifications such as adding Squeeze and Excitation (SE) block, removing the first normalization layer in a block, and reducing the downsampling factor in the stem stage effectively boosts robustness; and (3) architecture designs that harm robustness include inverted bottleneck, large dilation factor, Instance Normalization (IN), parametric activation functions [9], and reducing activation layers.

- **Top performance against strong adversarial attacks.** We demonstrate our guidelines' effectiveness by introducing the novel Robust Architecture (RobArch) model that instantiates the guidelines to build a family of top-performing models across parameter capacities against strong adversarial attacks. In particular, we compare our RobArch family with the Cross-Covariance Image Transformers (XCiT) family [1] that is the SOTA on RobustBench [7]. Every RobArch model outperform its XCiT counterpart with a similar model capacity (Figure 1). RobArch-S surpasses ResNet-50's AutoAttack (AA) accuracy by 9.18 percentage points, and is even more robust than WideResNet50-2 despite having $2.6\times$ fewer parameters. The robustness continues to increase as capacity increases. RobArch-L achieves the new SOTA AutoAttack (AA) [8] accuracy on the RobustBench ImageNet leaderboard. RobArch's performance advantage extrapolates to the Projected Gradient Descent (PGD) attack. Overall, the proposed RobArchs outperform both ConvNets and Transformers with similar total parameters.

2. Robust Architecture Design

We carefully select architectural components from off-the-shelf DNNs (ResNet [19], RegNet [40], DenseNet [25], and ConvNeXt [34]) that improve generalization accuracy. Based on the commonalities in these network designs, we group the components into three modification categories:

- **Network-level**: *depth*, *width*
- **Stage-level**: *stem stage*, *dense connection*
- **Block-level**: *kernel size*, *dilation*, *activation*, *SE*, *normalization*

Since ResNet [19] is a milestone in the history of DNN architecture, we choose its most popular instantiation, ResNet-50 ($\sim$26M parameters) as the base architecture, which consists of a stem stage, $n = 4$ body stages, and a classifier head, as our starting point. Each body stage contains multiple residual blocks with various depth and width configurations. Appendix A provides details of ResNet-50 configurations.

Notation and symbols used throughout this paper.

- We denote D-d_1-...-d_n as the depth of each stage in an n-stage network ($n \in \{3, 4, 5, 6\}$).
- For stage i, w_i and w_{b_i} are the numbers of channels in the pointwise and non-pointwise convolutions, respectively.
- Bottleneck multiplier b_i is the ratio of channels in pointwise to non-pointwise convolution, $b_i = w_i / w_{b_i}$.
- Assuming w_{g_i} is the group convolution width, g_i is the total number of groups in the non-pointwise convolution layer: $g_i = \lfloor w_{b_i} / w_{g_i} \rceil = \lfloor w_i / (b_i \times w_{g_i}) \rceil$.
- Width expansion ratio is $e = w_{i+1} / w_i$, $i \leq n - 1$.
- We use W-w_1-...-w_n, G-g_1-...-g_n, BM-b_1-...-b_n to represent the number of channels, group convolution groups, and bottleneck multiplier in an n-stage network.

Experimental settings. We train all models on ImageNet [12] with the recipes specified in Sec. 2.1. When studying a single architecture component (Sec. 2.2 - 2.4) and building cumulative networks (Sec. 3.1 & 3.2), we use 10-step PGD (PGD10) with different attack budgets ϵ ($\epsilon \in \{2, 4, 8\}$) for fast evaluations. After finalizing the model structures of the RobArch, we test all RobArchs against PGD100 and AA. All attacks are ℓ_∞ bounded. To control for the effect of model capacity, we constrain the networks' total parameters, *i.e.*, similar to ResNet-50 ($\sim$26M), throughout the exploration.

2.1. Training Techniques

Standard-AT. Adversarial Training (AT) is the most reliable defense to obtain robust DNNs [17, 36]. Standard-AT is formulated as a min-max optimization framework [36]. Given a network f_θ parameterized by θ, a dataset with samples (x_i, y_i), and a loss function $\mathcal{L}$, the robust optimization

(a) 4-stage networks attain top accuracies at much lower Giga Multiply–Accumulates (GMACs) than 3-stage networks. 5-stage and 6-stage networks are significantly less robust.

(b) Higher accuracy when guideline 2 is followed: depth rule $d_1 < d_2 < d_3 > c \times d_4$. We plot the mean accuracy (solid line) and 95% confidence interval (the shading).

(c) Robustness generally improves in all stages as depth increases and width decreases, until catastrophic overfitting happens with significantly reduced robustness.
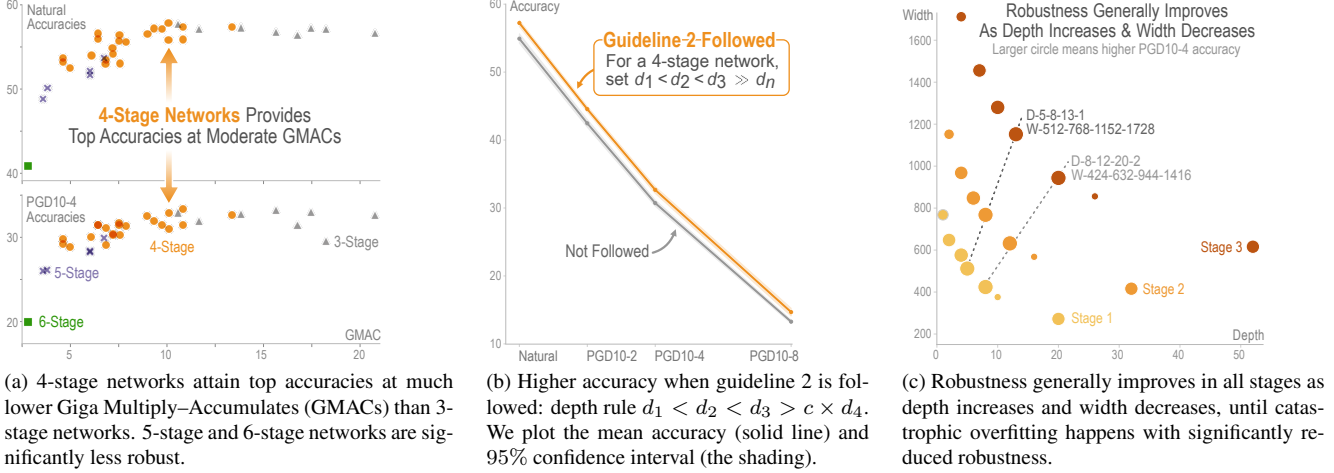
Figure 2. For network-level design, following guideline 2 to increase depth and decrease width in a 4-stage network provides the optimal robustness. We study (a) how the number of stages affect accuracies, (b) stage depth settings, and (c) depth-width trade-off. We only plot the first three stages of a 4-stage network in (c) for better visualization since the last stage is much shallower as per the optimal depth configurations in guideline 2. These observations also apply to other PGD attack budgets, as shown in Appendix D.

problem is formulated as:

$$\underset{\theta}{\text{argmin}}\, \mathbb{E}_{(x_i, y_i) \sim \mathbb{D}} \left[\max_{x'} \mathcal{L}\left(f_\theta, x', y\right) \right], \quad (1)$$

The inner adversarial example x' is generated on the fly during the training process, which aims to find an adversarial perturbation of a given data point x that achieves a high loss,

$$x'_{k+1} = \prod_{x+\Delta} \left(x'_k + \alpha \text{sgn}\left(\nabla_x \mathcal{L}(\theta, x'_k, y)\right)\right). \quad (2)$$

$\text{sgn}(\cdot)$ is the sign function, α is the step size, x'_k is the adversarial example generated after k steps ($1 \leq k \leq K$), $\Delta = \{\delta : \|\delta\|_\infty \leq \epsilon\}$ is the threat mode, and $\prod_{x+\Delta}$ is a projection operation that clips the perturbation back to the ϵ-ball centered on x if it goes beyond the attack budget.

Fast-AT. Fast-AT speeds up the Standard-AT and can robustify a ResNet-50 in under 13 hours [54]. It not only adopts Fast Gradient Sign Method (FGSM) [17] to generate adversarial samples during the training but also incorporates a cyclic learning rate [43] and mixed-precision arithmetic [37] to fully accelerate the AT with just 15 epochs. A line of research improves the performance and mitigates the catastrophic overfitting problem discovered in the Fast-AT, *e.g.*, YOPO [63], GradAlign [2], GAT [45], Sub-AT [30], *etc.*, but there are limited explorations on whether these recipes are compatible with the full ImageNet [12].

Although Fast-AT provides competitive PGD results, its resulting robustness on ResNet-50 is inferior to that of Standard-AT's as per the AA accuracy on the RobustBench leaderboard [7]. Therefore, we use Fast-AT as a rapid indicator while exploring different architecture components and building the RobArch family, and use Standard-AT to robustify all members in the RobArch family.

2.2. Network-level Design

Depth. In the standard ResNet-50 (D-3-4-6-3), each stage downsamples the input features by 2. The downsampling in the first stage is replaced by a max-pooling layer in the stem stage. We sample 36 architectures based on the depth relationship between each pair of stages, *i.e.*, $d_i \leq d_{i+1}$ and $d_i > d_{i+1}$. The widths in all stages are the same as ResNet-50, and when $n > 4$, we reuse the width in stage 4. For $n = 6$, even setting $d_i = 1, i \leq n$ leads to 1.83M more parameters than ResNet-50. Hence, there is only 1 data point for the 6-stage network, and we do not continue increasing the total stages. Fig. 2a shows the results after AT. 4-stage networks attain top natural and adversarial accuracies at much lower GMACs than 3-stage networks. 5-stage and 6-stage networks are significantly less robust. These results are expected since shallow stages, in general, compute on higher resolutions, and the depth of a 3-stage network in shallow stages is deeper than a 4-stage network by a large margin for similar total parameters. Hence, we select 4-stage networks and further explore the depth relationship between stages.

Huang *et al.* [26] found that reducing depth in the last stage of a 3-stage WideResNet34-10 improves robustness. Upon further inspection of our 4-stage models, we observe that increasing the stage depths d_i along with i, then significantly decreasing the depth in the last stage, leads to higher robustness. Fig. 2b shows that following such a rule ($d_1 < d_2 < d_3 > c \times d_4$) leads to a higher accuracy than not following it. We set $c = 3$ and leave the finetuning of a larger c to further research. RegNet [40] first discovered the depth pattern and applied it to improve benign accuracy. Our results extend this discovery to adversarial settings and

3

show that it helps robustifying architectures without incurring extra parameters. Overall, we found the optimal stage depth ratio is D-5-8-13-1 and listed its performance in Table 1 row 2.

Guideline 1: *3-stage $\approx$ 4-stage $>$ 5-stage $\gg$ 6-stage network in terms of robustness.*

Guideline 2: *For a 4-stage network, set $d_1 < d_2 < d_3 \gg d_n$, and D-5-8-13-1 provides the optimal robustness.*

Width. Factors that affect the stage width are pointwise convolution channels w_i, group convolution groups g_i, and bottleneck multiplier b_i. The width configurations of the standard ResNet-50 are W-256-512-1024-2048, G-1-1-1-1, BM-0.25-0.25-0.25-0.25. Unless otherwise specified, all configurations are kept consistent with ResNet-50 when studying one of the factors.

For $b_i \in \{0.125, 0.25, 0.5, 1, 2, 4\}$, we first test a constant $b_i = b$ in all stages. The accuracy reaches the peak when $b = 0.25$ or 0.5 and significantly decreases when increasing b from 0.5 to 4, which shows the inverted bottleneck is harmful to robustness. $b = 0.25$ (ResNet-50) has higher natural and PGD10-2 accuracy, while $b = 0.5$ has higher PGD10-4 and PGD10-8 accuracy. Both results are shown in Table 1 (rows 1 and 3). Then, we vary b_i for different stages, $b_{1,2} < b_{3,4}$ and $b_{1,2} > b_{3,4}$. The robustness of BM-0.25-0.25-2-2 is better than $b_i = 2$ but worse than $b_i = 0.25$. Surprisingly, BM-4-4-0.25-0.25 outperforms both $b_i = 0.25$ and $b_i = 4$. We further combine the two optimal bottleneck multipliers and set $b_{1,2} = 0.5, b_{3,4} = 0.25$. As shown in Table 1 row 4, this setting attains higher accuracy than both $b_i = 0.5$ and 0.25.

Next, we study the group convolution groups $g_i \in \{1, 2, 4, 8, 16, w_{b_i}\}$. $g_i = w_{b_i}$ is equivalent to the depth convolution. The pointwise convolution width w_i is adjusted to reach the controlled parameter budget, but b_i is always 0.25. For a constant $g_i = g$, we observe a significant increase from $g = 1$ (ResNet-50) to $g = 2$, but then the accuracy gradually decreases if we continue to increase g. Similar to the bottleneck multiplier study, we vary g_i for different stages. However, there is no further robustness gain. We list the results of $g = 2$ in Table 1 row 5.

For the width expansion ratio, we evaluate $e \in \{1, 1.5, 2, 2.5, 3\}$. The robustness rises and saturates at $e = 1.5$ and falls for a larger e. We show $e = 1.5$ in Table 1 row 6. Finally, we combine the optimal configurations for all three factors, *i.e.*, $b_{1,2} = 0.5, b_{3,4} = 0.25, g_i = g = 2, e = 1.5$. However, the robustness is inferior to that of just using the individual optimal settings. After a close look at all the results, we find setting a constant $b_i = b = 0.25$ works favorably with g and e. In addition, we observe $g = 2, e = 2$ and $g = 1, e = 1.5$ achieve the best two accuracies. The phenomenon also demonstrates that directly combining multiple individual optimal architectural settings does not transfer to a better model.

Guideline 3: *Inverted bottleneck harms robustness, especially when added to deeper stages.*

Guideline 4: *For a single modification, $b_{1,2} = 0.5, b_{3,4} = 0.25$, $g_i = 2$, and $e = 1.5$ all show promising improvements. However, merging all three configurations makes the model less robust, and the optimal width configurations are $e = 2, g = 2$ or $e = 1.5, g = 1$ with $b = 0.25$.*

Combining Depth and Width. In this part, we answer the following question: *Under a fixed model capacity, does increasing widths while decreasing depths, or vice versa, improve robustness?*

We use the optimal depth ratio, D-5-8-13-1. To provide a more general understanding and avoid overfitting to specific optimal settings, we cross-select $e = 1.5, g = 2, b = 0.25$ from the two optimal width configurations from guideline 4. We proportionally adjust depths and widths to accommodate the fixed budget. Fig. 2c displays the relationship between depths and widths using PGD10 accuracy. A larger bubble size means higher accuracy. The results show that increasing depth while decreasing width improves robustness in all stages. It is important to note that if we continue the trend, catastrophic overfitting [2] occurs during training. Since catastrophic overfitting drastically decreases the robustness, we should deepen the network but balance the depth and the width to stabilize the AT process. Comparing the top 2 models (dotted lines), both PGD10-2 and PGD10-4 accuracies of the deeper model are $0.10pp$ (percentage points) higher, but the PGD10-8 accuracy is $0.49pp$ lower, which is a sign of unstable training. Overall, D-5-8-13-1 is selected as the starting point of our cumulative model in Sec. 3.1. Compared to ResNet-50 (D-3-4-6-3), D-5-8-13-1 is much deeper and slimmer with significantly higher robustness: $\uparrow 1.15pp$ for natural accuracy, $\uparrow 2.03pp$ for PGD10-2, $\uparrow 2.62pp$ for PGD10-4, and $\uparrow 2.75pp$ for PGD10-8. We observe a similar depth-width relationship when scaling up the model in Sec. 3.2.

Guideline 5: *Under a fixed model capacity, first increase the network depth proportionally to the optimal depth until catastrophic overfitting happens,* i.e., *a sudden drop in loss and increase in training accuracy. The width is adjusted to fill the total parameter budget.*

2.3. Stage-level Design

Stem Stage. The stem stage in a standard ResNet-50 consists of a convolution layer and a max-pooling layer, each of which has a downsampling factor of 2. All 4 tandemly-connected body stages downsample the input resolution by 2 except the first stage. The convolution layer uses a 7×7 kernel and outputs 64-layer features.

In the stem stage, we modify the following architectural components: channel width, kernel size, "patchify" stem, and downsampling factor. First, we test channel width $\in \{32, 64, 96\}$ and kernel size $\in \{3, 5, 7\}$. With less than

Table 1. PGD10 robustness of architecture components. All configurations trained with Fast-AT and evaluated on full ImageNet validation set. We provide ResNet-50 as baseline. Appendix D shows detailed results, including PGD10-2 and PGD10-8.

Idx.	Configurations	Natural	PGD10-4
1	ResNet-50	56.09%	30.43%
Network-level Design			
2	D-5-8-13-1	57.35%	33.33%
3	BM-0.5-0.5-0.5-0.5	55.31%	30.52%
4	BM-0.5-0.5-0.25-0.25	56.11%	31.26%
5	G-2-2-2-2	57.31%	32.09%
6	W-512-768-1152-1728	57.17%	32.04%
	G-2-2-2-2		
7	BM-05-05-025-025	56.64%	31.04%
	W-512-768-1152-1728		
Stage-level Design			
8	Stem width 96	57.29%	32.06%
9	Move down ($\downarrow$) downsampling	57.08%	33.08%
10	Dense ratio 2	55.93%	30.73%
Block-level Design			
11	Kernel size 5	56.73%	32.77%
12	Kernel size 7	59.70%	34.67%
13	Dilation 2	52.98%	28.38%
14	Dilation 3	52.10%	27.97%
15	Act. GELU	57.48%	33.12%
16	Act. SiLU	58.19%	34.07%
17	Act. PSiLU	56.38%	33.76%
18	SE (ReLU)	57.83%	32.64%
19	Norm-BN-BN-0	54.15%	29.59%
20	Norm-BN-0-BN	56.04%	31.34%
21	Norm-0-BN-BN	56.18%	31.61%

0.01M increase in total parameters, switching convolution layer width from 32 to 64 and 64 to 96 improve the PGD10-4 accuracy by 0.7 and 1.65 percentage points, respectively. The "stem width 96" is located in Table 1 row 8. For kernel size = 3 or = 5, the training overfits to FGSM and leads to a completely non-robust model. The original kernel size is 7 in ResNet-50, and increasing it to 9 improves the PGD accuracy but leads to a drop in the natural accuracy.

We study the downsampling factor next. RegNet [40] is built based on ResNet, but the max-pooling layer in the stem stage is replaced by a stride 2 convolution shortcut connection in the first stage. We denote this operation as "move down ($\downarrow$) downsampling." The evaluation result (Table 1 row 9) manifests 0.99 and 2.65 percentage points increments in natural and PGD10-4 accuracy. We further disassemble the operation by only discarding the max-pooling layer without adding the stride 2 convolution shortcut. Although the robustness is slightly lower than "move $\downarrow$ downsampling," it still outperforms ResNet-50 by a large margin.

Vision Transformer (ViT) [16] first introduced the "patchify stem," and ConvNeXt [34] also incorporated the design to improve generalization. Motivated by those works, we replace the original stem with a 4×4 patch, *i.e.*, kernel size = stride = 4, and observe a slight increment in robustness. Since moving down the downsampling layer boosts robustness, we continue to test a smaller 2×2 patch. The accuracy increases as expected, but the gain is slightly lower than directly moving down the downsampling layer in a ResNet-style stem. Since a small kernel size in the early convolution layer leads to a smaller receptive field, a moderate kernel size of 7×7 is preferred. Overall, we select "stem width 96" and "move $\downarrow$ downsampling" as potential candidates while building the cumulative model in Sec. 3.1.

Guideline 6: *Replacing the max-pooling in the stem stage with a downsampling shortcut in the first stage significantly improves robustness.*

Guideline 7: *For the convolution layer in the stem stage, directly replacing it with a "patchify" stem design contributes to the robustness. However, the optimal configurations are increasing the channel width and setting kernel size = 7.*

Dense Connection. Huang *et al*. [25] introduced the dense connection in DenseNet that concatenates the feature maps of all preceding blocks within the stage as the input to the current block. We extend the definition and experiment with different dense ratios i ($i \in \{1, 2, 3, 4, 5\}$), *i.e.*, i preceding feature maps are used to construct the input. Only $i = 2$ shows minor improvements in PGD accuracy, and no strong benefits are observed (Table 1 row 10). We further remove the last Rectified Linear Unit (ReLU) since the original DenseNet uses the Pre-Activation (PreAct) operation [20]. However, the robustness is further degraded, and we assume the poor performance of reducing the last activation itself (discussed in 2.4) is a potential reason.

Guideline 8: *Dense connection is not beneficial to robustness.*

2.4. Block-level Design

Kernel Size. In this part, we study the kernel size in all body stages. Inspired by the large local window size in Swin-T [33], ConvNeXt [34] boosts the generalization accuracy via increasing the kernel size from 3×3 to 7×7. A large kernel size can extract more semantic information but implicitly increases the attack area during backpropagation. It is unclear whether a larger kernel size can bring higher robustness. We evaluate kernel size $\in \{3, 5, 7\}$ and find the accuracy grows along with the kernel size (Table 1 row 1, 11 & 12), but the total parameters also increase significantly: kernel = 3 (25.56M), kernel = 5 (45.68M), and kernel = 7 (75.86M). Thus, using a large kernel size is a potential candidate to optimize the robustness when scal-

ing up the model. We will revisit the design in Sec. 3.2.

Guideline 9: *Purely increasing the kernel size raises the model capacity but robustness significantly improves. Thus, it is a prospective option when scaling up the network.*

Dilation. Dilated convolution supports the exponential expansion of the receptive field without loss of resolution [62]. The operation offers a wider field of view at a similar computational cost. However, the results in Table 1 (row 1, 13 & 14) show that a larger dilation factor significantly decreases both natural and PGD accuracy after AT. Connecting to the previous kernel size section, we hypothesize that a larger receptive field facilitates the attacker. We still observe the robustness gain in using a large kernel size because the huge model capacity mitigates the effect, yet the accuracy drops when adjusting dilation since the operation does not change the model capacity. In Sec. 3.2, we also notice the kernel size is not effective in optimizing robustness if all other modifications are considered at the same scale.

Guideline 10: *Increasing dilation factor enlarges the attacking area, which leads to inferior robustness.*

Activation. We study two factors in the activation layer: the activation function and the number of activation layers in a block. For the activation function, we replace ReLU, which is used in ResNet-50, with two smoother functions, GELU and Sigmoid Linear Unit (SiLU). GELU alone significantly improves the robustness (Table 1 row 15), which echoes the result in [3]. SiLU further improves the accuracy (Table 1 row 16), which echoes the result in [57]. Recently, Dai *et al.* [9] added learnable parameters to original non-parametric functions, and proposed the parametric counterparts, *e.g.*, ReLU to Parametric ReLU (PReLU) and SiLU to Parametric SiLU (PSiLU) or Parametric Shifted SiLU (PSSiLU). These parametric functions outperform the non-parametric ones on CIFAR-10 [28]. We test these functions on ImageNet and observed PSiLU has the highest robustness among all parametric functions, as shown in Table 1 row 17. However, compared to the non-parametric versions, all parametric functions are less robust. Since the original paper only tested on the small-scale dataset, we believe such learnable functions are not compatible with the large-scale dataset. Next, we reduce the activation layers in each block. Neither reducing one nor reducing two activation layers show extra benefits to the robustness. The more activation layer we reduce, the worse the performance is.

Guideline 11: *Activation function significantly affects robustness. The non-parametric SiLU provides a competitive improvement.*

Guideline 12: *Reducing activation layers in a residual block severely hurts the robustness.*

Squeeze and Excitation (SE). Hu *et al.* [24] first introduced the SE block that explicitly explored interdependencies between channels, and adaptively recalibrated channel-wise feature responses. Inspired by RegNet

[40], we place the SE block between the last two convolutions in each block and set the reduction ratio as $1/4$. Compared to ResNet-50, Table 1 row 18 shows that adding SE significantly improves the robustness. Directly adding the SE module slightly increases the model capacity by 2.17M, but in Sec. 3.1, we show that sacrificing the parameters in other components by adopting the SE module can still improve the robustness, which proves the effectiveness of SE.

Since switching activation functions shows significant differences, we also replace ReLU in the SE block with SiLU, GELU and their parametric versions. We still observe that non-parametric activation functions are better than their parametric counterparts. The SILU is again the optimal activation for SE module. However, in Sec. 3.1, we find that replacing the activation function in activation layers and SE at the same time causes inferior robustness.

Guideline 13: *The SE module significantly contributes to robustness.*

Guideline 14: *The robustness improves if we just replace the activation function in the SE block. But the modification does not work favorably with switching the activation function in the residual block.*

Normalization. Similar to the activation layer, we examine both normalization functions and the number of normalization layers in a block. For the normalization function, we switch the original Batch Normalization (BN) [27] in ResNet-50 to IN [51]. The training is extremely hard to converge and thus leading to an almost non-robust model (PGD10-4: 8.54%). Then, we attempt to reduce the total normalization layers in a residual block. In Table 1, row 19 to 21 show that reducing the first normalization layer in a residual block optimizes the robustness. We keep reducing 2 BNs, and no further benefits are observed.

Guideline 15: *Switching BN to IN harms robustness.*

Guideline 16: *Reducing the first BN in a residual block benefits robustness.*

3. Experiments

In this section, we provide a roadmap that outlines the path we take to construct the RobArch using the guidelines in Sec. 2. Our roadmap combines architecture components such that for each combination we only keep components that increase robustness. Then, we scale up the resulting model and proposed a family of RobArch models. Finally, we compare RobArch with other SOTA architectures. See Appendix B for the full experimental setup. We also ablate Fast-AT and Standard-AT in Appendix C.

3.1. A Roadmap from ResNet-50 to RobArch-S

In this section, we cumulatively construct RobArch-S from ResNet-50 based on the proposed guidelines. Table 2 (upper) presents the procedures and results at each step

Table 2. The roadmap outlines the path we take to cumulatively improve the robustness and construct RobArch-S ($\sim$26M), RobArch-M ($\sim$46M), and RobArch-L ($\sim$104M) based on our guidelines. PGD10-2 and PGD10-8 show a similar trend of accuracy improvement as PGD10-4, and detailed results are shown in Appendix E.

	Configurations	Natural	PGD10-4
	Small: ResNet-50 → RobArch-S ($\mathcal{S}_7$)		
$\mathcal{S}_0$	ResNet-50	56.09%	30.43%
$\mathcal{S}_1$	$\mathcal{S}_0$ + D-5-8-13-1	57.35%	33.33%
$\mathcal{S}_{2a}$	$\mathcal{S}_1$ + $g=2, e=2, b=0.25$	57.98%	33.94%
$\mathcal{S}_{2b}$	$\mathcal{S}_1$ + $g=1, e=1.5, b=0.25$	57.52%	32.83%
$\mathcal{S}_3$	$\mathcal{S}_{2a}$ + Stem width 96 + Move down ($\downarrow$) downsampling	57.82%	34.86%
$\mathcal{S}_4$	$\mathcal{S}_3$ + SE (ReLU)	60.57%	36.61%
$\mathcal{S}_5$	$\mathcal{S}_4$ + Act. SiLU	62.04%	39.48%
$\mathcal{S}_6$	$\mathcal{S}_5$ + SE (SiLU)	60.32%	38.24%
$\mathcal{S}_7$	$\mathcal{S}_5$ + Norm-0-BN-BN	62.27%	39.88%
	Medium: RobArch-S ($\mathcal{S}_7$) → RobArch-M ($\mathcal{M}_2$)		
$\mathcal{M}_1$	$\mathcal{S}_7$ + Kernel size 5	63.82%	41.00%
$\mathcal{M}_2$	$\mathcal{S}_7$ + D-7-11-18-1	64.40%	42.06%
$\mathcal{M}_3$	$\mathcal{S}_7$ + W-384-760-1504-2944	63.52%	41.43%
	Large: RobArch-M ($\mathcal{M}_2$) → RobArch-L ($\mathcal{L}_2$)		
$\mathcal{L}_1$	$\mathcal{M}_2$ + Kernel size 7	64.08%	40.70%
$\mathcal{L}_2$	$\mathcal{M}_2$ + W-512-1024-2016-4032	66.08%	43.81%
$\mathcal{L}_3$	$\mathcal{M}_2$ + D-8-13-21-2	64.91%	43.09%
$\mathcal{L}_4$	$\mathcal{M}_2$ + D-10-16-26-2	65.28%	42.85%

of network modification. We start with network depth and width. Combining guideline 2 and guideline 5, model $\mathcal{S}_1$ selects the optimal depth configuration D-5-8-13-1. For width, we test the two optimal width configurations in guideline 4 and select $g = 2, e = 2$ ($\mathcal{S}_{2a}$). For the stem stage, model $\mathcal{S}_3$ increases the width to 96 and replace the max-pooling in the stem stage with a downsampling shortcut in the first stage according to guidelines 6 and 7. Then, we optimize the block settings in each stage. Guideline 13 suggests inserting a SE block between the last 2 convolutions. To accommodate the extra parameters in the modification, we reduce the width in all stages and build model $\mathcal{S}_4$. Next, $\mathcal{S}_5$ substitutes SiLU for ReLU in all 3 activation layers. However, we find that continuing to replace the activation function in the SE block lowers the robustness. Thus, we discard the modification, reduce the first BN layer, and construct $\mathcal{S}_7$. The resulting model is named RobArch-S. The guidelines are verified by the consistent increase in robustness along the network construction process. The total model capacity is comparable to ResNet-50, but both natural and PGD-4 accuracies have increased by 6.18 and 9.45 percentage points, respectively.
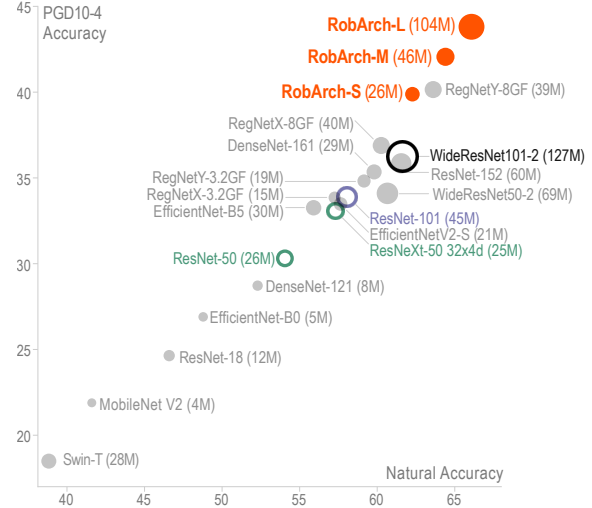

Figure 3. Our **RobArch** model family outperforms SOTA architectures under the same Fast-AT training method. With a similar model capacity, **RobArch-S** outperforms **ResNet-50** [19] and **ResNeXt-50 32×4d** [61] by 9.45 and 6.80 percentage points, respectively. **RobArch-M** outperforms **ResNet-101** by 8.16 percentage points. Compared to the models with larger parameters, **RobArch-S** is even more robust than **WideResNet101-2** despite having $4.85\times$ fewer parameters (highlighted in black). Appendix E shows detailed results, including other PGD attack budgets.

3.2. Scaling Up: The RobArch Family

We extend our investigation to optimize the robustness when scaling up the parameter budget. The budgets align with the XCiT [1] family since it is the current SOTA on the RobustBench ImageNet leaderboard [7]. Guideline 9 suggests increasing kernel size as a potential improvement when scaling up the model. Increasing total depth and width are another 2 promising directions [26, 60]. For the medium-sized budget ($\sim$46M), model $\mathcal{M}_1$ enlarges the kernel size from 3 to 5, model $\mathcal{M}_2$ proportionally deepens the network by a factor of 1.4, and model $\mathcal{M}_3$ widens the channels while keeping the depth same as RobArch-S. The training results of $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_3$ are shown in Table 2 (middle). In general, all three models are more robust than RobArch-S. But in terms of accuracy, increasing depth ($\mathcal{M}_2$) > increasing width ($\mathcal{M}_3$) > increasing kernel size ($\mathcal{M}_1$). Therefore, we set $\mathcal{M}_2$ as RobArch-M.

For the large-sized budget ($\sim$104M), model $\mathcal{L}_1$ increases the kernel size from 3 to 7, but leads to a drop in robustness, as shown in Table 2 (bottom). RobArch-M increases the depth of $\mathcal{S}_7$, and according to the depth-width trade-off in Fig. 2c, consistently increasing the depth can lead to unstable training. Therefore, model $\mathcal{L}_2$ increases the width in RobArch-M, and the robustness rises by a large margin.

Table 3. Our RobArch model outperforms *ConvNets* and *Transformers* with similar total parameters against $\ell_\infty = 4/255$ AA. Using the same training configurations as Salman *et al.* [41], our model outperforms both ResNet-50 and WideResNet50-2. Every RobArch model outperforms its XCiT counterpart at a similar capacity. Appendix F.2 shows the detailed results including PGD100 for $\epsilon \in \{2, 4, 8\}$.

Architecture	#Param	AutoAttack	Natural
ResNet-18 [41]	12M	25.32%	52.49%
PoolFormer-M12 [11]	22M	34.72%	66.16%
DeiT-S [3]	22M	35.50%	66.50%
DeiT-S+DiffPure [39]	22M	43.18%	73.63%
ResNet-50 [41]	26M	34.96%	63.87%
ResNet-50+DiffPure [39]	26M	40.93%	67.79%
ResNet-50+GELU [3]	26M	35.51%	67.38%
XCiT-S12 [11]	26M	41.78%	72.34%
RobArch-S	26M	**44.14%**	70.17%
XCiT-M12 [11]	46M	45.24%	74.04%
RobArch-M	46M	**46.26%**	71.88%
WideResNet50-2 [41]	69M	38.14%	68.41%
WideResNet50-2 +DiffPure [39]	69M	44.39%	71.16%
Swin-B [38]	88M	38.61%	74.36%
XCiT-L12 [11]	104M	47.60%	73.76%
RobArch-L	104M	**48.94%**	73.44%

We further deepen $\mathcal{L}_2$ to explore whether guideline 5 holds true when scaling up the model budget. $\mathcal{L}_3$ and $\mathcal{L}_4$ increase the depth by $1.6\times$ and $2\times$ and reduce the width to fit the total parameters. The results in Table 2 (bottom) show a decline in accuracy along with an increase in depth. The phenomenon extends guideline 5 that the depth-width relationship also applies to scaling up the models. Finally, we set $\mathcal{L}_2$ as RobArch-L based on the above discussions, and provide the following guidelines:

Guideline 17: *When scaling up the model, increasing the kernel size, depth, and width all contribute to the robustness. But proportionally increasing the optimal depth configuration is most effective.*

Guideline 18: *There exists a saturation point for purely increasing the depth to fill the parameter budget. We should enlarge channel widths when such a degradation happens.*

3.3. Results

In Fig. 3, we compare RobArch with a series of SOTA architectures. All architectures are trained with Fast-AT for a fair comparison, and we discover a similar trend for PGD10-2, PGD10-4, and PGD10-8. Below we provide a few observations based on PGD10-4 accuracy:

1) Under a similar model capacity, **RobArch-S** outperforms ResNet-50 [19] and ResNeXt-50 32×4d [61] by 9.45 and 6.80 percentage points, respectively. **RobArch-M** out-

performs ResNet-101 by 8.16 percentage points.

2) Compared to the models with larger parameters, **RobArch-S** is even more robust than WideResNet101-2 despite having $4.85\times$ fewer parameters.

3) Increasing the total parameters in general leads to higher robustness, and the natural accuracy is positively correlated with the adversarial accuracy after AT. Lightweight models, *e.g.*, MobileNet V2 and SqueezeNet-1.1, are among the least robust. The accuracies of **RobArchs** consistently grow when scaling up the model sizes.

4) Transformers, *e.g.*, Swin-T [33], and Transformer-based architectures, *e.g.*, ConvNeXt-T [34], are non-robust using Fast-AT. The phenomenon can be attributed to the differences in optimizers and learning rates, where most Transformer-related architectures use AdamW [35] and tiny learning rates.

As introduced in Sec. 2.1, we then train all RobArchs using Standard-AT. All three RobArchs outperform their XCiT [1] counterparts (Table 3). Using the same training configurations as Salman *et al.* [41], **RobArch-S** surpasses ResNet-50 AA accuracy by 9.18 percentage points, and is even more robust than WideResNet50-2 with $2.6\times$ fewer parameters. The robustness continues to improve when scaling up the model, and **RobArch-L** achieves the new SOTA AA [8] accuracy on RobustBench. RobArch's performance advantage also extrapolates to the PGD attack. Overall, the proposed RobArchs outperform both *ConvNets* and *Transformers* with a similar total parameters.

4. Related Work

A huge number of AT variants have been proposed, *e.g.*, TRADES [64], AWP [56], ADT [15], DART [52], MART [53], CAS [4], Max-Margin AT [14], *etc*. For the robust DNN research, only a few studies explored how architectures affect robustness [13, 36, 46, 48], *e.g.*, depths [60], widths [55] and activation functions [9, 57]. However, the total model capacity is unconstrained along with the architecture modifications. Besides, simply combining multiple individual optimal architectures does not transfer to a better model, *e.g.*, Huang *et al.* [26] studied depths and widths, and found the combination of the optimal depth and width ratios is less robust than just using the optimal width ratio.

5. Conclusion

In this work, we present the first large-scale systematic study on the robustness of architecture components under fixed parameter budgets. Through our investigation, we distill 18 actionable robust network design guidelines that empower model developers to gain deep insights. Our RobArch models instantiate the guidelines to build a family of top-performing models across parameter capacities against strong adversarial attacks.

References

[1] Alaaeldin Ali, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *Advances in neural information processing systems*, 34:20014–20027, 2021. 2, 7, 8

[2] Maksym Andriushchenko and Nicolas Flammarion. Understanding and improving fast adversarial training. *Advances in Neural Information Processing Systems*, 33:16048–16059, 2020. 1, 3, 4

[3] Yutong Bai, Jieru Mei, Alan L Yuille, and Cihang Xie. Are transformers more robust than cnns? *Advances in Neural Information Processing Systems*, 34:26831–26843, 2021. 1, 6, 8, 16

[4] Yang Bai, Yuyuan Zeng, Yong Jiang, Shu-Tao Xia, Xingjun Ma, and Yisen Wang. Improving adversarial robustness via channel-wise activation suppressing. *arXiv preprint arXiv:2103.08307*, 2021. 8

[5] Srinadh Bhojanapalli, Ayan Chakrabarti, Daniel Glasner, Daliang Li, Thomas Unterthiner, and Andreas Veit. Understanding robustness of transformers for image classification. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10231–10241, 2021. 1

[6] Tom B Brown, Dandelion Mané, Aurko Roy, Martín Abadi, and Justin Gilmer. Adversarial patch. *arXiv preprint arXiv:1712.09665*, 2017. 1

[7] Francesco Croce, Maksym Andriushchenko, Vikash Sehwag, Edoardo Debenedetti, Nicolas Flammarion, Mung Chiang, Prateek Mittal, and Matthias Hein. Robustbench: a standardized adversarial robustness benchmark. *arXiv preprint arXiv:2010.09670*, 2020. 1, 2, 3, 7, 13

[8] Francesco Croce and Matthias Hein. Reliable evaluation of adversarial robustness with an ensemble of diverse parameter-free attacks. In *International conference on machine learning*, pages 2206–2216. PMLR, 2020. 2, 8, 13

[9] Sihui Dai, Saeed Mahloujifar, and Prateek Mittal. Parameterizing activation functions for adversarial robustness. In *2022 IEEE Security and Privacy Workshops (SPW)*, pages 80–87. IEEE, 2022. 2, 6, 8

[10] Nilaksh Das, Sheng-Yun Peng, and Duen Horng Chau. Skelevision: Towards adversarial resiliency of person tracking with multi-task learning. *arXiv preprint arXiv:2204.00734*, 2022. 1

[11] Edoardo Debenedetti, Vikash Sehwag, and Prateek Mittal. A light recipe to train robust vision transformers. *arXiv preprint arXiv:2209.07399*, 2022. 8, 16

[12] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 2, 3

[13] Chaitanya Devaguptapu, Devansh Agarwal, Gaurav Mittal, Pulkit Gopalani, and Vineeth N Balasubramanian. On adversarial robustness: A neural architecture search perspective. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 152–161, 2021. 1, 8

[14] Gavin Weiguang Ding, Yash Sharma, Kry Yik Chau Lui, and Ruitong Huang. Mma training: Direct input space margin maximization through adversarial training. *arXiv preprint arXiv:1812.02637*, 2018. 1, 8

[15] Yinpeng Dong, Zhijie Deng, Tianyu Pang, Jun Zhu, and Hang Su. Adversarial distributional training for robust deep learning. *Advances in Neural Information Processing Systems*, 33:8270–8283, 2020. 8

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 5

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 1, 2, 3

[18] Minghao Guo, Yuzhe Yang, Rui Xu, Ziwei Liu, and Dahua Lin. When nas meets robustness: In search of robust architectures against adversarial attacks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 631–640, 2020. 16

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 1, 2, 7, 8, 12

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *European conference on computer vision*, pages 630–645. Springer, 2016. 5

[21] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 2

[22] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in Neural Information Processing Systems*, 33:6840–6851, 2020. 15

[23] Ramtin Hosseini, Xingyi Yang, and Pengtao Xie. Dsrna: Differentiable search of robust neural architectures. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6196–6205, 2021. 1

[24] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 6

[25] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2, 5

[26] Hanxun Huang, Yisen Wang, Sarah Erfani, Quanquan Gu, James Bailey, and Xingjun Ma. Exploring architectural ingredients of adversarially robust deep neural networks. *Advances in Neural Information Processing Systems*, 34:5545–5559, 2021. 1, 3, 7, 8, 16

[27] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 6

[28] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

9

[29] Alexey Kurakin, Ian Goodfellow, and Samy Bengio. Adversarial machine learning at scale. *arXiv preprint arXiv:1611.01236*, 2016. 1

[30] Tao Li, Yingwen Wu, Sizhe Chen, Kun Fang, and Xiaolin Huang. Subspace adversarial training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13409–13418, 2022. 3

[31] Chen Liu, Mathieu Salzmann, Tao Lin, Ryota Tomioka, and Sabine Süsstrunk. On the loss landscape of adversarial training: Identifying challenges and how to overcome them. *Advances in Neural Information Processing Systems*, 33:21476–21487, 2020. 1

[32] Xin Liu, Huanrui Yang, Ziwei Liu, Linghao Song, Hai Li, and Yiran Chen. Dpatch: An adversarial patch attack on object detectors. *arXiv preprint arXiv:1806.02299*, 2018. 1

[33] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 5, 8

[34] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 1, 2, 5, 8

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 8

[36] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations*, 2018. 1, 2, 8, 13

[37] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, et al. Mixed precision training. *arXiv preprint arXiv:1710.03740*, 2017. 3

[38] Yichuan Mo, Dongxian Wu, Yifei Wang, Yiwen Guo, and Yisen Wang. When adversarial training meets vision transformers: Recipes from training to architecture. *arXiv preprint arXiv:2210.07540*, 2022. 8, 16

[39] Weili Nie, Brandon Guo, Yujia Huang, Chaowei Xiao, Arash Vahdat, and Anima Anandkumar. Diffusion models for adversarial purification. *arXiv preprint arXiv:2205.07460*, 2022. 8, 15, 16

[40] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436, 2020. 2, 3, 5, 6, 12

[41] Hadi Salman, Andrew Ilyas, Logan Engstrom, Ashish Kapoor, and Aleksander Madry. Do adversarially robust imagenet models transfer better? *Advances in Neural Information Processing Systems*, 33:3533–3545, 2020. 8, 13, 15, 16

[42] Rulin Shao, Zhouxing Shi, Jinfeng Yi, Pin-Yu Chen, and Cho-Jui Hsieh. On the adversarial robustness of vision transformers. *arXiv preprint arXiv:2103.15670*, 2021. 1

[43] Leslie N Smith. Cyclical learning rates for training neural networks. In *2017 IEEE winter conference on applications of computer vision (WACV)*, pages 464–472. IEEE, 2017. 3

[44] Yang Song, Taesup Kim, Sebastian Nowozin, Stefano Ermon, and Nate Kushman. Pixeldefend: Leveraging generative models to understand and defend against adversarial examples. *arXiv preprint arXiv:1710.10766*, 2017. 1

[45] Gaurang Sriramanan, Sravanti Addepalli, Arya Baburaj, et al. Guided adversarial attack for evaluating and enhancing adversarial defenses. *Advances in Neural Information Processing Systems*, 33:20297–20308, 2020. 3

[46] Dong Su, Huan Zhang, Hongge Chen, Jinfeng Yi, Pin-Yu Chen, and Yupeng Gao. Is robustness the cost of accuracy?– a comprehensive study on the robustness of 18 deep image classification models. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 631–648, 2018. 1, 8

[47] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*, 2013. 1

[48] Shiyu Tang, Ruihao Gong, Yan Wang, Aishan Liu, Jiakai Wang, Xinyun Chen, Fengwei Yu, Xianglong Liu, Dawn Song, Alan Yuille, et al. Robustart: Benchmarking robustness on architecture design and training techniques. *arXiv preprint arXiv:2109.05211*, 2021. 8

[49] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2

[50] James Tu, Mengye Ren, Sivabalan Manivasagam, Ming Liang, Bin Yang, Richard Du, Frank Cheng, and Raquel Urtasun. Physically realizable adversarial examples for lidar object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13716–13725, 2020. 1

[51] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Instance normalization: The missing ingredient for fast stylization. *arXiv preprint arXiv:1607.08022*, 2016. 6

[52] Yisen Wang, Xingjun Ma, James Bailey, Jinfeng Yi, Bowen Zhou, and Quanquan Gu. On the convergence and robustness of adversarial training. *arXiv preprint arXiv:2112.08304*, 2021. 8

[53] Yisen Wang, Difan Zou, Jinfeng Yi, James Bailey, Xingjun Ma, and Quanquan Gu. Improving adversarial robustness requires revisiting misclassified examples. In *International Conference on Learning Representations*, 2019. 8

[54] Eric Wong, Leslie Rice, and J Zico Kolter. Fast is better than free: Revisiting adversarial training. *arXiv preprint arXiv:2001.03994*, 2020. 1, 3, 12

[55] Boxi Wu, Jinghui Chen, Deng Cai, Xiaofei He, and Quanquan Gu. Do wider neural networks really help adversarial robustness? *Advances in Neural Information Processing Systems*, 34:7054–7067, 2021. 1, 8

[56] Dongxian Wu, Shu-Tao Xia, and Yisen Wang. Adversarial weight perturbation helps robust generalization. *Advances*

10

in Neural Information Processing Systems, 33:2958–2969, 2020. 8

[57] Cihang Xie, Mingxing Tan, Boqing Gong, Alan Yuille, and Quoc V Le. Smooth adversarial training. *arXiv preprint arXiv:2006.14536*, 2020. 6, 8, 16

[58] Cihang Xie, Jianyu Wang, Zhishuai Zhang, Zhou Ren, and Alan Yuille. Mitigating adversarial effects through randomization. *arXiv preprint arXiv:1711.01991*, 2017. 1

[59] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 501–509, 2019. 1

[60] Cihang Xie and Alan Yuille. Intriguing properties of adversarial training at scale. In *International Conference on Learning Representations*, 2019. 7, 8

[61] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 7, 8

[62] Fisher Yu and Vladlen Koltun. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*, 2015. 6

[63] Dinghuai Zhang, Tianyuan Zhang, Yiping Lu, Zhanxing Zhu, and Bin Dong. You only propagate once: Accelerating adversarial training via maximal principle. *Advances in Neural Information Processing Systems*, 32, 2019. 3

[64] Hongyang Zhang, Yaodong Yu, Jiantao Jiao, Eric Xing, Laurent El Ghaoui, and Michael Jordan. Theoretically principled trade-off between robustness and accuracy. In *International conference on machine learning*, pages 7472–7482. PMLR, 2019. 1, 8

A. Network Configurations

A.1. Overview of ResNet-style ConvNets

A standard ResNet-style ConvNet includes a stem stage, several body stages, and a classification head, as shown in Fig. 4. A typical body stage consists of multiple residual blocks, where each of them has a shortcut connection that skips other layers and feeds the output of the previous layer to the current output of the block [19]. The stem stage proceeds the input image through a convolution layer and a max-pooling that downsample the resolution by 4 in total. The final classification head passes the extracted features from body stages through an average pooling and a linear layer that outputs the predictions. Table 4 lists ResNet-50 configurations written in notations defined in the paper.

A.2. RobArch Architecture

The RobArch follows the ResNet-style ConvNet design. We display block designs for ResNet-50 and RobArch-S in Fig. 5. Following the RegNet design [40], we add the SE block after the 3×3 convolution layer in each block. The SE reduction ratio is 0.25.

ResNet-style ConvNets
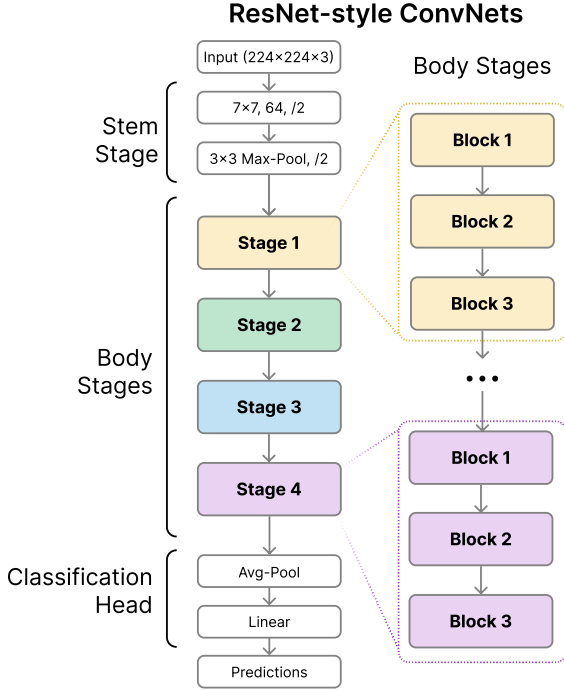
Figure 4. An overview of ResNet-style ConvNet design, which includes a stem stage, several body stages, and a classification head.

B. Experimental Settings

We use Fast-AT as a rapid indicator while exploring different architecture components and building the RobArch

Table 4. ResNet-50 configurations written in notations defined in the paper. The left column lists architecture components, and the right column shows notations. ResNet-50 does not have SE block, so the configuration is "N/A". For activation, ReLU-ReLU-ReLU represents the three activation layers in a residual block. The same also applies to normalization.

	Notation
Depth	D-3-4-6-3
Width	W-256-512-1024-2048
	G-1-1-1-1
	BM-0.25-0.25-0.25-0.25
Stem stage	Stem width 64
	Stem kernel 7
	Downsample factor 4
Dense connection	Dense ratio 1
Kernel size	Kernel size 3
Dilation	Dilation 1
Activation	Act. ReLU
	ReLU-ReLU-ReLU
SE	N/A
Normalization	Norm. BN
	BN-BN-BN
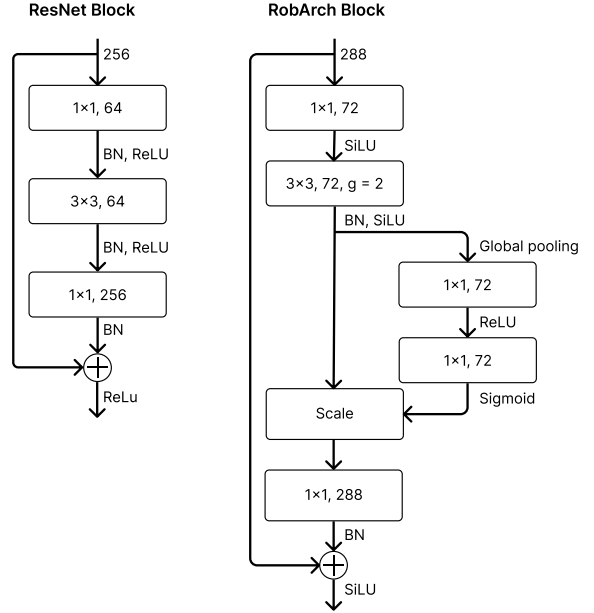
Figure 5. Block designs for a ResNet and a RobArch. For simplicity, "$1 \times 1, 64$" means pointwise convolution with 64-layer output channels. "$g = 2$" means 2 group convolution groups, and the default group is 1.

family. We follow the same 3-phase training as proposed in the Fast-AT paper [54]. Fast-AT sets training $\epsilon = 1.25\times$ test ϵ and finds catastrophic overfitting happens when training ϵ goes beyond 10. Therefore, we set training $\epsilon \in \{2.5, 5.0, 7.5\}$ corresponding to test $\epsilon \in \{2, 4, 6\}$ and show

Table 5. Determine training ϵ for Fast-AT using ResNet-50. Training $\epsilon = 2.5$ shows the highest natural and PGD10-2 accuracies, while training $\epsilon = 7.5$ shows the highest PGD10-8 accuracy. Overall, training $\epsilon = 5.0$ is selected for all Fast-AT experiments since it exhibits a balanced performance on all natural and attack budgets.

Training ϵ	Natural	PGD10-2	PGD10-4	PGD10-8
2.5	60.04%	43.06%	25.34%	6.49%
5.0	56.09%	42.66%	30.43%	12.61%
7.5	49.80%	36.86%	26.95%	13.87%

the results in Table 5. Larger training ϵ exhibits higher robustness against strong attacks at the cost of lowering the accuracies of natural and weak attacks. We select training $\epsilon = 5.0$ for its balanced performance on natural and various attack budgets. We use Standard-AT to robustify all members in the RobArch family, and follow the same training configurations as Salman *et al.* [41].

Our RobArchs are evaluated against the two strongest adversarial attacks, PGD [36] and AA [8]. All PGD attacks are tested on the full ImageNet validation set. AA is an ensemble of four different parameter-free attacks, three white- and one black-box. We use the same 5000 ImageNet validation subset provided by the RobustBench [7] for AA comparison.

C. Ablations on Adversarial Training

We ablate Fast-AT and Standard-AT for two purposes: 1) verify the robustness order is consistent under two different AT methods, 2) compute whether the two approaches exhibit comparable robustness increases when subjected to the same ablation.

Since Standard-AT incurs longer training time, we randomly select one small budget model $\mathcal{S}_4$ and show the results in Table 6. For natural, PGD10-4 and AA runs, $\mathcal{S}_4$ outperforms ResNet-50 but is inferior to RobArch-S, which demonstrates the robustness order is consistent under Fast-AT and Standard-AT. Then, we compute the robustness gain using PGD10-4 as an example. From ResNet-50, $\mathcal{S}_4$ to RobArch-S, accuracy increases by 6.18 and 3.27 percentage points under Fast-AT, and increases by 6.01 and 2.52 percentage points under Standard-AT. Both training methods show comparable robustness increases on the same architecture against the same attack. The observation also extrapolates to natural and AA accuracies. As expected, Standard-AT displays higher robustness than Fast-AT. Hence, we conclude that Fast-AT serves as a good indicator when exploring different architecture components and building the RobArch family. Standard-AT can fully robustify all members in the RobArch family after finalizing the architectures.

Table 6. Ablations on Fast-AT and Standard-AT. We randomly select one small budget model $\mathcal{S}_4$ from the roadmap and train it with both methods. The results show that the robustness order is consistent under two different AT methods, and the scales of robustness increment are also comparable.

Model	Fast-AT		Standard-AT		
	Natural	PGD10-4	Natural	PGD10-4	AA
$\mathcal{S}_0$ (ResNet-50)	56.09%	30.43%	63.87%	39.66%	34.96%
$\mathcal{S}_4$	60.57%	36.61%	68.88%	45.67%	41.44%
$\mathcal{S}_7$ (RobArch-S)	62.27%	39.88%	70.17%	48.19%	44.14%

Table 7. Our RobArch model family outperforms SOTA architectures under the same Fast-AT training method. The results are consistent across natural and different attack budgets. We highlight all three RobArchs for easy comparisons.

Architecture	#Param	Natural	PGD10-2	PGD10-4	PGD10-8
SqueezeNet 1.1	1 M	0.10 %	0.10 %	0.10 %	0.10 %
MobileNet V2	4 M	41.60%	31.23%	21.89%	8.94 %
EfficientNet-B0	5 M	48.78%	37.74%	26.90%	10.92%
ShuffleNet V2 2.0×	7 M	49.99%	0.01 %	0.01 %	0.02 %
DenseNet-121	8 M	52.29%	40.06%	28.72%	12.23%
ResNet-18	12 M	46.59%	35.05%	24.64%	9.95 %
RegNetX-3.2GF	15 M	57.26%	45.74%	33.85%	15.37%
RegNetY-3.2GF	19 M	59.15%	47.09%	34.82%	15.51%
EfficientNetV2-S	21 M	57.64%	45.89%	33.48%	14.03%
ResNeXt-50 32×4d	25M	57.33%	45.46%	33.08%	14.45%
ResNet-50	26 M	56.09%	42.66%	30.43%	12.61%
RobArch-S	26 M	62.27%	51.67%	39.88%	18.99%
Swin-T	28 M	38.83%	28.08%	18.49%	6.20 %
ConvNeXt-T	29 M	21.35%	15.39%	10.51%	4.07 %
DenseNet-161	29 M	59.80%	47.60%	35.35%	15.77%
EfficientNet-B5	30 M	55.90%	44.80%	33.26%	14.53%
RegNetY-8GF	39 M	63.61%	52.26%	40.15%	19.21%
RegNetX-8GF	40 M	60.26%	48.98%	36.89%	17.22%
ResNet-101	45 M	58.04%	45.72%	33.90%	15.93%
RobArch-M	46 M	64.40%	53.97%	42.06%	20.98%
ResNet-152	60 M	61.55%	48.50%	35.85%	15.87%
WideResNet50-2	69 M	60.66%	46.99%	34.10%	15.37%
RobArch-L	104M	66.08%	55.52%	43.81%	22.50%
WideResNet101-2	127M	61.63%	49.10%	36.23%	16.14%

D. Robust Architecture Design Results

This section presents the detailed results for all architecture components, using five tables. In each table, we use a bold font to highlight the results that have been presented in the paper, and in the caption, we describe the additional information that we are introducing here. Table 8 is depth-only, Table 13 is width-only, Table 14 is depth-width combination, Table 9 includes all stage-level designs, and Table 10 includes all block-level designs. For each component, its table includes architecture configurations, total parameters,

Table 8. PGD10 robustness of depth. Bold font means the results have been presented in the paper. All configurations are trained with Fast-AT and evaluated on full ImageNet validation set. ResNet-50 serves as the baseline. We presented D-5-8-13-1 in the main paper, and provide results for all 3-, 4-, 5- and 6-stage networks here.

Config	#Param	Natural	PGD10-2	PGD10-4	PGD10-8
ResNet-50	25.56M	56.09%	42.66%	30.43%	12.61%
3-stage Network					
D-16-16-16	25.02M	57.15%	41.35%	29.57%	14.37%
D-10-18-16	25.15M	56.47%	43.32%	31.52%	14.57%
D-3-22-16	25.78M	56.77%	44.99%	33.24%	15.14%
D-16-25-14	25.30M	56.69%	44.53%	32.63%	14.39%
D-2-16-18	26.26M	57.31%	44.97%	32.72%	14.00%
D-3-29-14	25.51M	57.27%	44.74%	33.02%	14.88%
D-3-4-20	25.21M	57.69%	45.27%	32.95%	14.46%
D-8-2-20	25.00M	57.12%	44.32%	31.90%	13.50%
4-stage Network					
D-1-5-6-3	25.70M	55.98%	43.54%	31.46%	13.54%
D-5-2-6-3	25.14M	52.93%	40.41%	29.14%	12.60%
D-1-4-7-3	26.53M	56.60%	43.62%	31.51%	13.76%
D-6-4-4-3	23.53M	54.19%	42.11%	30.40%	13.30%
D-3-5-2-4	25.83M	53.98%	41.44%	30.08%	13.13%
D-4-3-10-2	25.35M	55.62%	43.15%	31.32%	14.03%
D-2-7-13-1	25.22M	57.19%	44.16%	31.91%	13.89%
D-2-9-13-1	25.78M	57.89%	45.08%	32.84%	14.63%
D-2-13-8-2	25.78M	55.86%	42.91%	30.96%	13.40%
D-1-1-15-1	25.71M	55.74%	43.41%	31.45%	13.51%
D-2-5-14-1	25.78M	56.49%	44.13%	32.58%	14.73%
D-5-8-13-1	25.71M	**57.35%**	44.83%	**33.33%**	15.46%
D-2-12-12-1	25.51M	55.89%	43.39%	31.45%	13.56%
D-4-8-1-4	25.62M	54.84%	42.44%	30.23%	12.86%
D-1-4-2-4	25.41M	52.46%	40.25%	28.80%	12.22%
D-2-1-3-4	25.76M	53.23%	41.50%	29.76%	12.51%
D-3-24-5-2	25.58M	57.41%	44.66%	32.65%	14.42%
D-2-8-5-3	25.49M	56.43%	43.65%	31.70%	13.62%
D-6-4-2-4	25.76M	53.48%	42.04%	31.07%	13.70%
D-10-6-5-3	25.49M	57.17%	43.65%	31.45%	13.25%
D-10-2-2-4	25.48M	53.03%	41.01%	30.32%	13.45%
D-1-2-3-4	25.97M	53.68%	41.05%	29.21%	11.92%
5-stage Network					
D-1-1-3-1-2	25.42M	48.85%	36.89%	25.98%	10.37%
D-1-1-3-2-1	25.42M	50.14%	37.33%	26.11%	10.35%
D-3-6-2-2-1	25.85M	51.64%	39.12%	28.23%	12.24%
D-2-3-7-1-1	26.06M	52.16%	39.79%	28.40%	11.72%
D-3-4-6-2-1	29.76M	53.67%	41.25%	29.88%	12.65%
6-stage Network					
D-1-1-1-1-1-1	27.39M	40.82%	29.46%	20.00%	7.52%

natural, PGD10-2, PGD10-4, and PGD10-8 accuracies.

Table 9. PGD10 robustness of all stage-level designs. Bold font means the results have been presented in the paper. All configurations are trained with Fast-AT and evaluated on full ImageNet validation set. ResNet-50 serves as the baseline. We presented "Stem width 96" and "Move down ($\downarrow$) downsampling" for the stem stage, and "Dense ratio 2" for the dense connection in the main paper. We complete the results by providing all other configurations, and PGD attack budgets here.

Config	#Param	Natural	PGD10-2	PGD10-4	PGD10-8
ResNet-50	25.56M	56.09%	42.66%	30.43%	12.61%
Stem Stage					
Stem width 32	25.54M	55.89%	41.64%	29.73%	13.25%
Stem width 96	25.57M	**57.29%**	44.55%	**32.06%**	13.74%
Stem kernel 3	25.55M	38.93%	0.46%	0.55%	0.30%
Stem kernel 5	25.55M	59.59%	0.38%	0.09%	0.04%
Stem kernel 9	25.56M	55.75%	43.00%	31.19%	13.63%
Move down ($\downarrow$) downsampling	25.56M	**57.08%**	45.19%	**33.08%**	14.50%
Downsample factor 2	25.56M	56.03%	44.48%	32.86%	14.71%
"Patchify 4"	25.55M	55.40%	43.45%	31.68%	13.80%
"Patchify 2"	25.55M	56.38%	44.21%	31.91%	13.48%
Dense Connection					
Dense ratio 2	25.56M	**55.93%**	42.85%	**30.73%**	12.67%
Dense ratio 3	25.56M	53.45%	40.70%	29.39%	12.84%
Dense ratio 4	25.56M	55.02%	42.44%	30.52%	12.98%
Dense ratio 5	25.56M	54.45%	41.96%	30.07%	12.49%
Dense ratio 5 ReLU-ReLU-0	25.56M	49.68%	37.32%	26.15%	10.28%

E. Roadmap Results

This section presents detailed results for the roadmap we take to construct the RobArch family using Table 11. We demonstrate each architecture component in the cumulative RobArch construction process improves natural and PGD10-4 in the main paper. In Table 11, we show that the accuracy gain is also consistent on PGD10-2 and PGD10-8.

F. SOTA Architecture Comparisons

F.1. Fast-AT Comparisons

This section presents the detailed results of RobArch and other SOTA architectures after Fast-AT using Table 7. With a similar model capacity, RobArch-S outperforms ResNet-50 and ResNeXt-50 43×4d, and RobArch-M outperforms ResNet-101. Compared to models with larger parameters, RobArch-S is even more robust than WideResNet101-2 despite having $4.85\times$ fewer parameters. The accuracy continues to increase while scaling up the RobArch models, with RobArch-L achieving the highest natural and adver-

Table 10. PGD10 robustness of all block-level designs. Bold font means the results have been presented in the paper. All configurations are trained with Fast-AT and evaluated on full ImageNet validation set. ResNet-50 serves as the baseline. Bold means results have already appeared in the main paper. We complete the results by providing all other configurations and PGD attack budgets here. For activation, 0-0-ReLU means only the last activation layer is preserved in a block and the first two are discarded. The same also applies to normalization.

Config	#Param	Natural	PGD10-2	PGD10-4	PGD10-8
ResNet-50	25.56M	56.09%	42.66%	30.43%	12.61%
Kernel Size					
Kernel size 5	45.68M	**56.73%**	44.55%	**32.77%**	14.62%
Kernel size 7	75.86M	**59.70%**	47.28%	**34.67%**	14.99%
Dilation					
Dilation 2	25.56M	**52.98%**	40.38%	**28.38%**	11.79%
Dilation 3	25.56M	**52.10%**	39.69%	**27.97%**	11.05%
Activation					
Act. GELU	25.56M	**57.48%**	45.05%	**33.12%**	14.80%
Act. SiLU	25.56M	**58.19%**	46.21%	**34.07%**	14.68%
Act. PReLU	25.56M	55.81%	42.52%	30.38%	12.76%
Act. PSiLU	25.56M	**56.38%**	44.90%	**33.76%**	15.40%
Act. PSSiLU	25.56M	57.43%	44.44%	32.22%	13.71%
ReLU-ReLU-0	25.56M	51.54%	38.69%	27.05%	10.94%
ReLU-0-ReLU	25.56M	53.91%	41.22%	29.62%	12.30%
0-ReLU-ReLU	25.56M	54.81%	42.10%	30.34%	12.86%
0-0-ReLU	25.56M	51.03%	39.12%	28.15%	12.09%
0-ReLU-0	25.56M	47.18%	34.85%	24.12%	9.51%
ReLU-0-0	25.56M	44.21%	32.34%	22.24%	8.77%
Squeeze and Excitation (SE)					
SE (ReLU)	27.73M	**57.83%**	45.09%	**32.64%**	14.01%
SE (SiLU)	27.73M	58.49%	45.79%	33.63%	14.51%
SE (GELU)	27.73M	58.27%	45.66%	33.55%	14.56%
SE (PSiLU)	27.73M	56.98%	44.19%	32.19%	13.68%
SE (PSSiLU)	27.73M	57.55%	45.27%	33.33%	14.73%
Normalization					
Norm. IN	25.51M	17.15%	12.49%	8.54%	3.55%
BN-BN-0	25.53M	**54.15%**	41.12%	**29.59%**	12.36%
BN-0-BN	25.55M	**56.04%**	43.29%	**31.34%**	13.37%
0-BN-BN	25.55M	**56.18%**	43.64%	**31.61%**	13.47%
0-0-BN	25.54M	54.47%	41.91%	30.13%	12.65%
0-BN-0	25.52M	54.55%	41.94%	30.06%	12.62%
BN-0-0	25.52M	54.44%	41.47%	29.72%	12.50%

parts. Using the same training configurations as Salman *et al*. [41], RobArch-S surpasses ResNet-50 AA accuracy by 9.18 percentage points, and is even more robust than WideResNet50-2 with 2.6× fewer parameters. The robustness continues to scale with model capacity, and RobArch-L achieves the new SOTA AA accuracy on the RobustBench leaderboard. It is important to note that ResNet-50+DiffPure [39] designed a novel AT method via using diffusion models [22] for adversarial purification. Although the method improves the AA accuracy by 5.97 percentage points, our architecture modifications show stronger robustness even without finetuning the Standard-AT method. We believe a carefully designed training recipe can further improve RobArchs' robustness. For PGD, the RobArch-S again outperforms ResNet-50 and even WideResNet50-2 using the same Standard-AT configurations. Overall, our RobArchs outperform both *ConvNets* and *Transformers* with similar total parameters.

sarial accuracies.

F.2. Standard-AT Comparisons

This section compares our RobArchs with other SOTA models against both PGD and AA in Table 12. For AA, all three RobArchs outperform their XCiT counter-

Table 11. The roadmap outlines the path we take to cumulatively improve the robustness and construct RobArch-S ($\sim$26M), RobArch-M ($\sim$46M), and RobArch-L ($\sim$104M) based on our guidelines. Natural and PGD10-4 accuracies were already shown in the main paper. PGD10-2 and PGD10-8 show similar trends of accuracy improvement as PGD10-4.

	Configurations	#Param	Natural	PGD10-2	PGD10-4	PGD10-8
Small: ResNet-50 $\rightarrow$ RobArch-S ($\mathcal{S}_7$)						
$\mathcal{S}_0$	ResNet-50	25.71M	56.09%	42.66%	30.43%	12.61%
$\mathcal{S}_1$	$\mathcal{S}_0$ + D-5-8-13-1	25.56M	57.35%	44.83%	33.33%	15.46%
$\mathcal{S}_{2a}$	$\mathcal{S}_1$ + $g=2, e=2, b=0.25$	25.84M	57.98%	46.00%	33.94%	15.27%
$\mathcal{S}_{2b}$	$\mathcal{S}_1$ + $g=1, e=1.5, b=0.25$	25.53M	57.52%	44.60%	32.83%	14.23%
$\mathcal{S}_3$	$\mathcal{S}_{2a}$ + Stem width 96 + Move down ($\downarrow$) downsampling	25.85M	57.82%	46.37%	34.86%	15.92%
$\mathcal{S}_4$	$\mathcal{S}_3$ + SE (ReLU)	26.15M	60.57%	49.05%	36.61%	16.43%
$\mathcal{S}_5$	$\mathcal{S}_4$ + Act. SiLU	26.15M	62.04%	51.41%	39.48%	18.95%
$\mathcal{S}_6$	$\mathcal{S}_5$ + SE (SiLU)	26.15M	60.32%	49.74%	38.24%	18.18%
$\mathcal{S}_7$	$\mathcal{S}_5$ + Norm-0-BN-BN	26.14M	62.27%	51.67%	39.88%	18.99%
Medium: RobArch-S ($\mathcal{S}_7$) $\rightarrow$ RobArch-M ($\mathcal{M}_2$)						
$\mathcal{M}_1$	$\mathcal{S}_7$ + Kernel size 5	45.95M	63.82%	52.89%	41.00%	19.90%
$\mathcal{M}_2$	$\mathcal{S}_7$ + D-7-11-18-1	45.90M	64.40%	53.97%	42.06%	20.98%
$\mathcal{M}_3$	$\mathcal{S}_7$ + W-384-760-1504-2944	46.16M	63.52%	53.11%	41.43%	20.27%
Large: RobArch-M ($\mathcal{M}_2$) $\rightarrow$ RobArch-L ($\mathcal{L}_2$)						
$\mathcal{L}_1$	$\mathcal{M}_2$ + Kernel size 7	103.89M	64.08%	52.92%	40.70%	19.61%
$\mathcal{L}_2$	$\mathcal{M}_2$ + W-512-1024-2016-4032	104.07M	66.08%	55.52%	43.81%	22.50%
$\mathcal{L}_3$	$\mathcal{M}_2$ + D-8-13-21-2	104.13M	64.91%	54.64%	43.09%	21.81%
$\mathcal{L}_4$	$\mathcal{M}_2$ + D-10-16-26-2	104.14M	65.28%	54.49%	42.85%	21.42%

Table 12. Our RobArch model outperforms *ConvNets* and *Transformers* with similar total parameters against $\ell_\infty = 4/255$ AA and $\ell_\infty = 2/255, 4/255, 8/255$ PGD attacks. Using the same training configurations as Salman *et al*. [41], our model outperforms both ResNet-50 and WideResNet50-2. Every RobArch model outperforms its XCiT counterpart at a similar capacity.

Architecture	#Param	Natural	AA	PGD10-4	PGD50-4	PGD100-4	PGD100-2	PGD100-8
ResNet-18 [41]	12M	52.49%	25.32%	30.06%	29.61%	29.61%	40.98%	11.57%
RobNet-large [18]	13M	61.26%	-	37.16%	37.15%	37.14%	-	-
PoolFormer-M12 [11]	22M	66.16%	34.72%	-	-	-	-	-
DeiT-S [3]	22M	66.50%	35.50%	41.03%	40.34%	40.32%	-	-
DeiT-S+DiffPure [39]	22M	**73.63%**	43.18%	-	-	-	-	-
ResNet-50 [41]	26M	63.87%	34.96%	39.66%	38.98%	38.96%	52.15%	15.83%
ResNet-50+DiffPure [39]	26M	67.79%	40.93%	-	-	-	-	-
ResNet50+SiLU [57]	26M	69.70%	-	43.00%	41.90%	-	-	-
ResNet50+GELU [3]	26M	67.38%	35.51%	40.98%	40.28%	40.27%	-	-
ResNet-50-R [26]	26M	56.63%	-	-	31.14%	-	-	-
XCiT-S12 [11]	26M	72.34%	41.78%	-	-	-	-	-
RobArch-S	26M	70.17%	**44.14%**	**48.19%**	**47.78%**	**47.77%**	**60.06%**	**21.77%**
XCiT-M12 [11]	46M	**74.04%**	45.24%	-	-	-	-	-
RobArch-M	46M	71.88%	**46.26%**	**49.84%**	**49.32%**	**49.30%**	**61.89%**	**23.01%**
WideResNet50-2 [41]	69M	68.41%	38.14%	42.51%	41.33%	41.24%	55.86%	16.29%
WideResNet50-2+DiffPure [39]	69M	71.16%	44.39%	-	-	-	-	-
Swin-B [38]	88M	74.36%	38.61%	-	-	-	-	-
XCiT-L12 [11]	104M	**73.76%**	47.60%	-	-	-	-	-
RobArch-L	104M	73.44%	**48.94%**	**51.72%**	**51.04%**	**51.03%**	**63.49%**	**25.31%**

Table 13. PGD10 robustness of width. Bold font means the results have been presented in the paper. All configurations are trained with Fast-AT and evaluated on full ImageNet validation set. ResNet-50 serves as the baseline. In the main paper, we presented BM-0.5-0.5-0.5-0.5 and BM-0.5-0.5-0.25-0.25 for bottleneck multiplier, G-2-2-2-2 for group convolution groups, W-512-768-1152-1728 for expansion ratio, and the combined model. We complete the results by providing all other configurations, and PGD attack budgets here.

Channel	Group	Bottleneck Multiplier	#Param	Natural	PGD10-2	PGD10-4	PGD10-8
	ResNet-50		25.56M	56.09%	42.66%	30.43%	12.61%
Bottleneck Multiplier							
W-320-672-1456-3136	G-1-1-1-1	BM-0.125-0.125-0.125-0.125	25.47M	53.47%	41.42%	30.11%	13.40%
W-128-256-568-1304	G-1-1-1-1	BM-0.5-0.5-0.5-0.5	25.57M	**55.31%**	42.48%	**30.52%**	13.23%
W-64-144-320-720	G-1-1-1-1	BM-1-1-1-1	25.61M	53.07%	40.93%	29.54%	12.70%
W-32-72-168-384	G-1-1-1-1	BM-2-2-2-2	25.72M	51.17%	38.79%	27.32%	11.22%
W-16-32-88-200	G-1-1-1-1	BM-4-4-4-4	26.19M	47.67%	35.93%	25.30%	10.32%
W-256-512-168-384	G-1-1-1-1	BM-0.25-0.25-2-2	26.42M	52.33%	39.79%	28.52%	12.30%
W-24-48-1024-2048	G-1-1-1-1	BM-4-4-0.25-0.25	25.20M	55.78%	43.09%	30.79%	12.89%
W-128-256-1024-2048	G-1-1-1-1	BM-0.5-0.5-0.25-0.25	24.83M	**56.11%**	43.38%	**31.26%**	13.47%
Group Convolution Groups							
W-256-512-1080-2504	G-2-2-2-2	BM-0.25-0.25-0.25-0.25	26.02M	**57.31%**	44.25%	**32.09%**	13.91%
W-288-576-1248-2592	G-4-4-4-4	BM-0.25-0.25-0.25-0.25	25.58M	56.28%	44.00%	31.52%	13.33%
W-256-512-1280-2816	G-8-8-8-8	BM-0.25-0.25-0.25-0.25	25.81M	56.54%	42.49%	30.07%	12.86%
W-256-576-1344-2816	G-16-16-16-16	BM-0.25-0.25-0.25-0.25	25.61M	54.83%	42.92%	31.03%	13.28%
W-304-640-1384-2848	G-76-160-337-712	BM-0.25-0.25-0.25-0.25	25.52M	55.17%	42.34%	30.45%	12.72%
W-256-512-1040-2112	G-8-8-1-1	BM-0.25-0.25-0.25-0.25	26.13M	55.49%	42.42%	30.78%	12.82%
W-256-512-1248-2784	G-1-1-8-8	BM-0.25-0.25-0.25-0.25	25.69M	55.94%	43.28%	31.15%	13.92%
W-256-512-1248-2592	G-2-2-4-4	BM-0.25-0.25-0.25-0.25	25.41M	57.13%	43.88%	31.44%	13.48%
Channel / Expansion Ratio							
W-1112-1112-1112-1112	G-1-1-1-1	BM-0.25-0.25-0.25-0.25	25.70M	56.77%	43.18%	31.08%	13.68%
W-512-768-1152-1728	G-1-1-1-1	BM-0.25-0.25-0.25-0.25	25.95M	**57.17%**	44.05%	**32.04%**	14.06%
W-144-360-904-2264	G-1-1-1-1	BM-0.25-0.25-0.25-0.25	26.01M	53.89%	41.83%	30.33%	13.38%
W-88-264-792-2376	G-1-1-1-1	BM-0.25-0.25-0.25-0.25	25.81M	52.39%	40.60%	29.36%	12.58%
Combined							
W-512-768-1152-1728	G-2-2-2-2	BM-0.5-0.5-0.25-0.25	24.43M	**56.64%**	43.56%	**31.04%**	13.17%

Table 14. PGD10 robustness of combining depth and width. We use a bold font to highlight results that have been presented in the paper. Specifically, the paper uses a scatter plot to visualize how the PGD10-4 accuracy changes as we vary depth and width. Here, we additionally show the results for PGD10-2 and PGD10-8. All configurations are trained with Fast-AT and evaluated on full ImageNet validation set.

Depth	Width	#Param	Natural	PGD10-2	PGD10-4	PGD10-8
D-1-2-4-1	W-768-1152-1712-2560	25.69M	54.28%	41.16%	**29.10%**	11.83%
D-2-4-7-1	W-648-968-1456-2160	25.55M	57.25%	43.60%	**31.52%**	13.59%
D-4-6-10-1	W-576-848-1280-1904	25.51M	57.08%	44.18%	**32.32%**	14.46%
D-5-8-13-1	W-512-768-1152-1728	25.18M	57.24%	44.69%	**33.05%**	15.36%
D-8-12-20-2	W-424-632-944-1416	25.37M	57.74%	44.79%	**33.15%**	14.87%
D-10-16-26-2	W-376-568-856-1280	25.56M	61.36%	44.92%	**27.23%**	5.67%
D-20-32-52-4	W-272-416-616-928	25.52M	55.76%	43.28%	**31.31%**	13.03%