# Appendices

## A1. AdvRush Implementation Details

Following DARTS, the search space of AdvRush includes following operations:

- Zero Operation
- Skip Connect
- 3x3 Average Pooling
- 3x3 Max Pooling
- 3x3 Separable Conv
- 5x5 Separable Conv
- 3x3 Dilated Conv
- 5x5 Dilated Conv

The pseudo code is presented in Algorithm 1.

**Algorithm 1:** AdvRush (Search)

**Input :** $E$ = total number of epochs for search
$E_{\text{warmup}}$ = number of epochs to warm-up
$\gamma$ = regularization strength
**Output:** $f_A^*(\cdot)$ = final architecture

Initialize $f_{\text{super}}(\omega_0, \alpha_0)$
**For** $i$ = from 1 to $E$:
  **If** $i \leq E_{\text{warmup}}$
    Update $\omega_i$ using $\nabla_\omega \mathcal{L}_{\text{train}}(\omega_{i-1}, \alpha_{i-1})$ (SGD)
    Update $\alpha_i$ using $\nabla_\alpha \mathcal{L}_{\text{val}}(\omega_i, \alpha_{i-1})$ (Adam)
  **Else**
    Update $\omega_i$ using $\nabla_\omega \mathcal{L}_{\text{train}}(\omega_{i-1}, \alpha_{i-1})$ (SGD)
    Update $\alpha_i$ using $\nabla_\alpha [\mathcal{L}_{\text{val}}(\omega_i, \alpha_{i-1}) + \gamma \mathcal{L}_\lambda]$ (Adam)
  **End**
**End**
Derive $f_A^*(\cdot)$ through discretization rule of DARTS from $f_{\text{super}}(\omega_E, \alpha_E)$

## A2. Hyperparameters and Datasets

For adversarial training, a different set of hyperparameters for each dataset. Hyperparameter settings for the datasets used in our experiments are provided in Table A1. Details regarding the datasets are provided in Table A6.

Table A1. Details of hyperpameters used for adversarial training on different datasets. Learning rate is decayed by the factor of 0.1 at selected epochs. CIFAR refers to both CIFAR-10 and -100.

| | CIFAR | SVHN | Tiny-ImageNet |
|---|---|---|---|
| Optimizer | SGD | SGD | SGD |
| Momentum | 0.9 | 0.9 | 0.9 |
| Weight decay | 1e-4 | 1e-4 | 1e-4 |
| Epochs | 200 | 200 | 90 |
| LR | 0.1 | 0.01 | 0.1 |
| LR decay | (100, 150) | (100, 150) | (30, 60) |

## A3. Difference in $\mathcal{L}_\lambda$

In Figure A1, we visualize how the difference in $\alpha$ update rule between DARTS and AdvRush affects the value of $\mathcal{L}_\lambda$. We use $\gamma$ of 0.01 for AdvRush, and $\mathcal{L}_\lambda$ is introduced at the $50^{th}$ epoch. Notice that AdvRush experiences a steeper drop in $\mathcal{L}_\lambda$; this phenomenon implies that the final supernet of AdvRush is indeed smoother than that of DARTS. Since the only difference in the two compared search algorithms is the learning objective of $\alpha$, it is safe to assume that this extra smoothness is induced solely by the difference in $\alpha$.

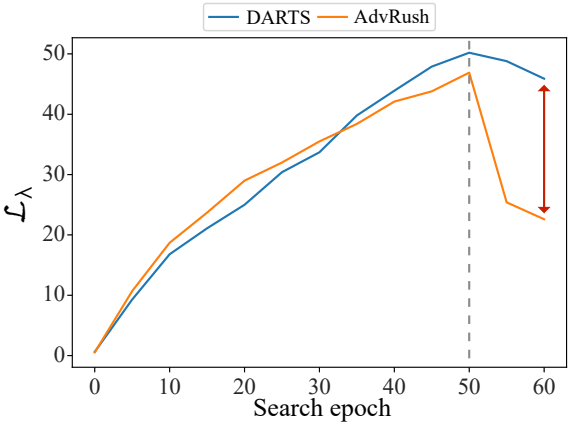



Figure A1. The difference in value of $\mathcal{L}_\lambda$ between DARTS and AdvRush. $\mathcal{L}_\lambda$ is introduced at Epoch 50, noted in a gray dotted line. The red arrow points out the gap in $\mathcal{L}_\lambda$ between the two algorithms, caused by different $\alpha$ update rules.

## A4. More Input Loss Landscapes

We provide additional input loss landscapes of other tested architectures after standard training (Figure A4) and adversarial training (Figure A5). All loss landscapes are drawn using the CIFAR-10 dataset. In both figures, $\vec{n}$ and $\vec{r}$ denote perturbations in normal and random directions, respectively. Degrees of perturbation in input data for standard trained and adversarially trained architectures are set differently to account for the discrepancy in their sensitivity to perturbation.

## A5. Architecture Visualization

The architectures used for the analysis in Section 6.3 of the main text are visualized in Figure A3 and A2.

## A6. HRS Calculation

HRS is calculated as follows:

$$HRS = \frac{2CR}{C+R} \tag{A1}$$

Table A2. Effect of the change in the magnitude of $\gamma$. Baseline refers to the AdvRush with default $\gamma$ of 0.01. The best result in each column is in bold, and the second best result is underlined.

| Arch | Std. Tr. | | | Adv. Tr. | | |
|---|---|---|---|---|---|---|
| | Clean | FGSM | HRS | Clean | $PGD^{20}$ | HRS |
| PDARTS | 97.49% | 54.51% | 69.92 | 85.37% | 51.32% | 64.10 |
| Arch 0 | 97.58% | 55.91% | 71.09 | 87.30% | 53.07% | 66.01 |
| Arch 1 | 95.59% | 47.54% | 63.50 | 85.65% | 52.70% | 65.25 |
| Arch 2 | 95.55% | 56.57% | 71.06 | 85.68% | 52.93% | 65.44 |
| Arch 3 | 95.24% | 48.46% | 64.24 | 83.15% | 53.42% | 64.05 |
| Arch 4 | 95.45% | 50.75% | 66.27 | 83.03% | 53.67% | 65.20 |
| Arch 5 | 95.05% | 45.53% | 61.57 | 83.04% | 48.76% | 61.44 |
| Arch 6 | 94.93% | 52.20% | 67.36 | 82.82% | 48.41% | 61.10 |
| Arch 7 | 94.01% | 29.19% | 44.55 | 82.78% | 46.48% | 59.53 |
| Arch 8 | 93.24% | 24.55% | 38.87 | 81.13% | 46.37% | 59.01 |
| Arch 9 | 93.08% | 21.19% | 34.52 | 81.87% | 46.22% | 59.08 |
| Arch 10 | 93.88% | 23.94% | 38.15 | 82.26% | 46.81% | 59.66 |

Table A3. Comparison with additional variations of DARTS. The results below further consolidate the effectiveness of AdvRush.

| Search | Epoch | Clean(%) | FGSM(%) | PGD20(%) |
|---|---|---|---|---|
| PGD Search | 50 | 82.25 | 55.31 | 46.87 |
| DARTS | 60 | 84.01 | 59.40 | 52.03 |
| AdvRush | 60 | **87.30** | **60.87** | **53.07** |

Table A4. Results of using weight decay=5e-4 for adversarial training show that the robustness of the AdvRush architecture is unaffected by the choice of training hyperparameters.

| Model | PGD20(%) | Model | PGD20(%) |
|---|---|---|---|
| ResNet-18 | 47.06 | DARTS | 53.14 |
| DenseNet | 51.15 | PDARTS | 50.07 |
| RACL | 52.75 | RobNet-Free | 50.38 |
| AdvRush | **54.61** | | |

where C denotes the clean accuracy, and R denotes the robust accuracy. For standard-trained architectures, we use the robust accuracy under FGSM attack for R because the robust accuracy of standard-trained architectures under PGD attack reaches near-zero. For adversarially-trained architectures, we use the robust accuracy under $PGD^{20}$ attack for R. C, R, and HRS values for all architectures used in Section 6.3 of the main text can be found in Table A2.

## A7. Additional Baselines

We compare AdvRush against two additional variations of DARTS, which could provide more meaningful baselines to further clarify the effectiveness of AdvRush. The first variation, named "PGD Search," applies PGD adversarial training to both the architecture parameters and the weight parameters of the supernet and is thus far more computationally expensive than AdvRush. In the second variation, we report the search result of removing the regulrization term in AdvRush; this is equivalent to running DARTS for 60 epochs. The results are reported in Table A3.

## A8. Change in Weight Decay

To see whether AdvRush achieves achieves a high robust accuracy even when the set of hyperparameters for adversarial training is changed, we adversarially train all tested architectures using the weight decay value of $5e-4$. All other hyperparameters are kept the same. The results are reported in Table A4. Although all the architectures experience some amount of increase in the robust accuracy after changing the value of the weight decay factor, AdvRush once again achieves the highest robust accuracy among them.

## A9. Extended Results on Other Datasets

In addition to the results in the main text, we conduct PGD attack with different number of iterations and evaluate the robust accuracy on CIFAR-100, SVHN, and Tiny-ImageNet. We use varying iterations from 7 to 1000. The results are visualized in Figure A6. AdvRush consistently outperforms other architectures, regardless of the strength of the adversary.

## A10. Full Ablation Results

In Table A5, the full ablation analysis of AdvRush including the robust accuracies under AutoAttack is presented. No matter the value of $\gamma$ used, architectures searched with AdvRush achieve high robust accuracies under AutoAttack. Such results suggest that AdvRush does not require excessive tuning of $\gamma$ to search for a robust neural architecture.

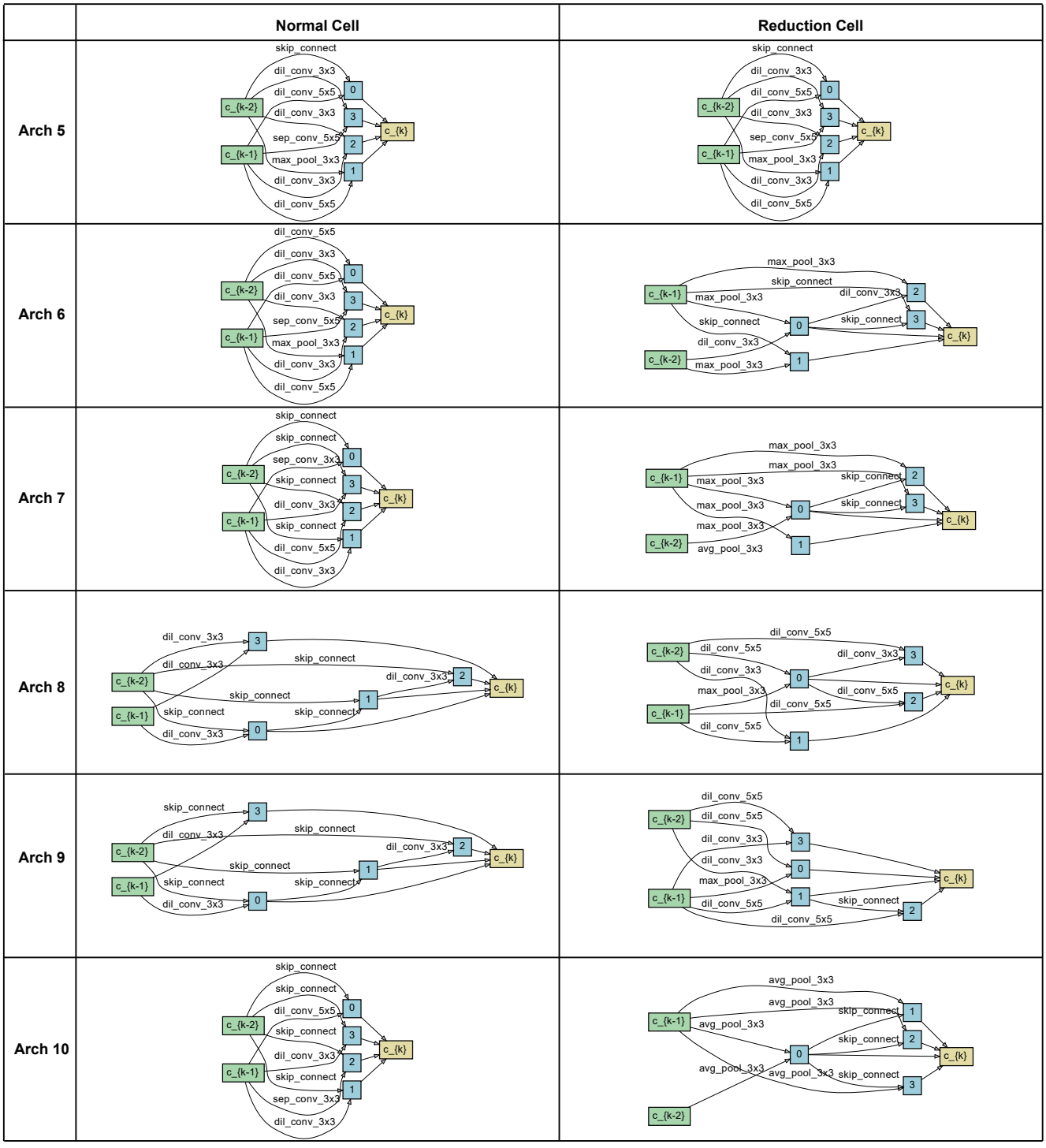



Figure A2. Architectures searched by adversarial training of the supernet. A normal cell maintains the dimension of input feature maps, while a reduction cell reduces it. The entire neural network is constructed by stacking 18 normal cells and 2 reduction cells following the DARTS convention. The reduction cell is placed at the $1/3$ and $2/3$ points of the network.

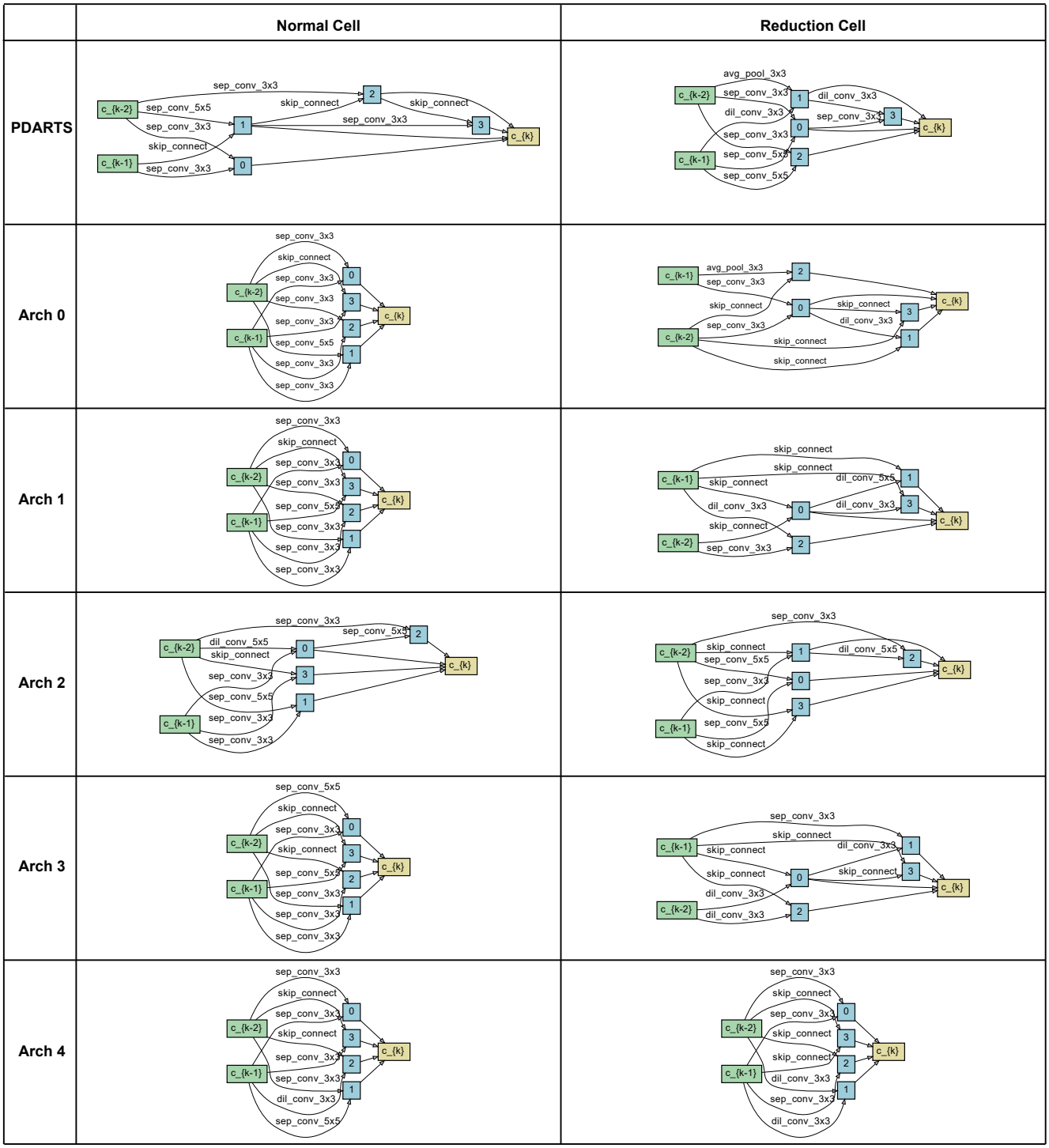
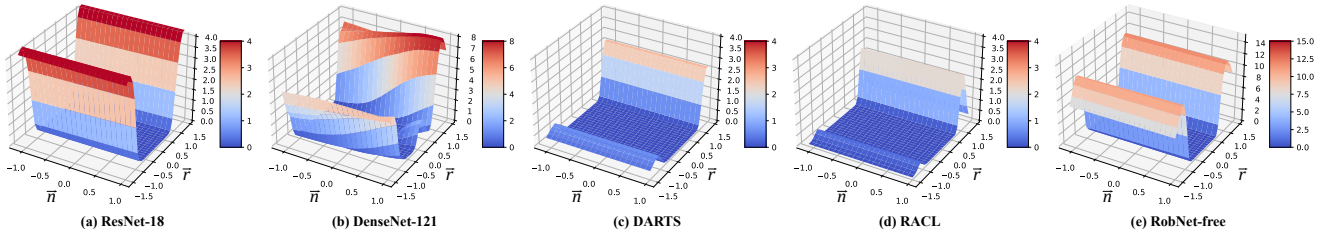
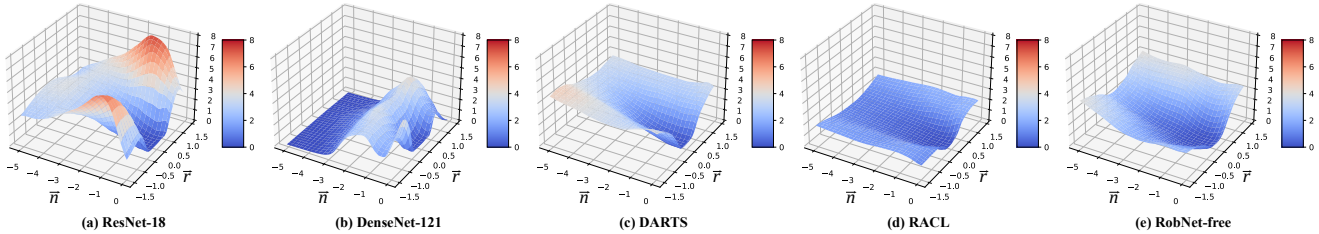



Figure A3. Architectures searched by PDARTS and AdvRush. A normal cell maintains the dimension of input feature maps, while a reduction cell reduces it. The entire neural network is constructed by stacking 18 normal cells and 2 reduction cells following the DARTS convention. The reduction cell is placed at the $1/3$ and $2/3$ points of the network.



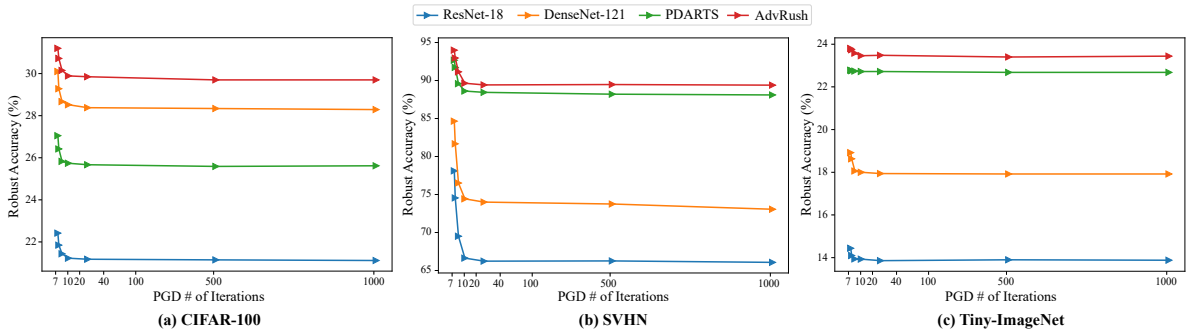
(a) ResNet-18 (b) DenseNet-121 (c) DARTS (d) RACL (e) RobNet-free

Figure A4. Input loss landscapes after standard training on CIFAR-10.



(a) ResNet-18 (b) DenseNet-121 (c) DARTS (d) RACL (e) RobNet-free

Figure A5. Input loss landscapes after adversarial training on CIFAR-10.




(a) CIFAR-100 (b) SVHN (c) Tiny-ImageNet

Figure A6. Robust accuracies on (a) CIFAR-100, (b) SVHN, and (c) Tiny-ImageNet under different iterations of PGD attack.

Table A5. Effect of the change in the magnitude of $\gamma$. Baseline refers to AdvRush with default $\gamma$ of 0.01. The best result in each column is in bold, and the second best result is underlined.

| $\gamma$-value | Clean | FGSM | $PGD^{20}$ | $PGD^{100}$ | $APGD_{CE}$ | $APGD^{T}$ | $FAB^{T}$ | Square | AA |
|---|---|---|---|---|---|---|---|---|---|
| 0.001 (x 0.1) | 85.65% | 60.04% | 52.70% | 52.39% | 51.39% | 49.16% | 49.13% | 49.13% | 49.13% |
| 0.005 (x 0.5) | <u>85.68%</u> | <u>60.31%</u> | 52.93% | 52.61% | <u>51.77%</u> | <u>49.63%</u> | <u>49.62%</u> | <u>49.62%</u> | <u>49.62%</u> |
| 0.01 (Ours.) | **87.30%** | **60.87%** | 53.07% | <u>52.80%</u> | 50.05% | **50.04%** | **50.04%** | **50.04%** | **50.04%** |
| 0.02 (x 2) | 83.15% | 59.34% | <u>53.42%</u> | **53.19%** | **52.22%** | 49.60% | 49.60% | 49.60% | 49.60% |
| 0.1 (x 10) | 83.03% | 59.69% | **53.67%** | 52.20% | 51.22% | 49.12% | 49.11% | 49.11% | 49.11% |

Table A6. Summary of datasets used for the experiments. CIFAR-10, CIFAR-100, and SVHN, which are available directly from torchvision, are split only into train and test sets by default.

| Dataset | # of Train Data | # of Validation Data | # of Test Data | # of Classes | Image Size |
|---|---|---|---|---|---|
| CIFAR-10 | 50,000 | - | 10,000 | 10 | $(32 \times 32)$ |
| CIFAR-100 | 50,000 | - | 10,000 | 100 | $(32 \times 32)$ |
| SVHN | 73,257 | - | 26,032 | 10 | $(32 \times 32)$ |
| Tiny-ImageNet | 100,000 | 5,000 | 5,000 | 200 | $(64 \times 64)$ |