Supplementary Material

A. Detail of Existing Methods of Comparison Experiments

In order to evaluate the performance of ARREST, we selected many existing methods for comparison. Table 6 lists the DNNs trained on existing methods, with 22 and 13 DNN models for CIFAR-10 and CIFAR-100, respectively. The standard and AutoAttack accuracies, Sum, ARDist, and sources of their accuracies are depicted. Almost all models used WideResNet-34-10 as an architecture, though several ones used WideResNet-28-10. We found that AWP [63], LAS-AT [23], S^2O [24], and LBGAT [12] had the four best scores in terms of Sum and ARDist on both CIFAR-10 and CIFAR-100.

Table 6. Detailed information on existing methods. The source of each accuracy is also described in the rightmost column. The methods are sorted using the score of ARDist in ascending order. * indicates a result obtained with WideResNet-28-10; the other results were obtained with WideResNet-34-10.

	Standard	AutoAttack	Sum	ARDist	Source of performance
CIFAR-10 - $\varepsilon = 8/255$					
AT [37]	87.14%	44.04%	131.18	-1.500	RobustBench
DAT [61]	86.20%	45.38%	132.10	-1.991	Directly copied from [52]
TLA [34]	86.21%	47.41%	133.62	-1.282	RobustBench
CAT [5]	89.61%	34.78%	124.39	-1.259	Directly copied from [52]
MART [62]	83.62%	50.98%	134.60	-0.922	Provided model
BAT [59]*	91.20%	29.35%	120.55	-0.690	RobustBench
FS [68]*	90.00%	36.64%	126.64	-0.502	RobustBench
AIT [69]*	90.25%	36.45%	126.70	-0.297	RobustBench
BS [8]	85.32%	51.12%	136.44	-0.230	RobustBench
AGKD-BML [58]*	86.25%	50.59%	136.84	0.184	Directly copied from [58]
FAT [71]	89.34%	43.05%	132.39	0.343	Provided model
SAL [26]	91.51%	34.22%	125.73	0.496	RobustBench
IAD [72]	85.09%	52.29%	137.38	0.535	Directly copied from [72]
SAT [52]	86.84%	50.75%	137.59	0.766	Directly copied from [52]
LAT [29]	87.80%	49.12%	136.92	0.853	RobustBench
TRADES [70]	84.92%	53.08%	138.00	1.180	RobustBench
ST [31]	84.92%	53.54%	138.46	1.616	Directly copied from [31]
Bag of Tricks [40]	84.24%	53.88%	138.12	1.829	Directly copied from [40]
LAS-AT [23]	86.23%	53.58%	139.81	2.236	Directly copied from [23]
AWP [63]	85.57%	54.04%	139.61	2.314	Directly copied from [63]
S^2O [24]	85.67%	54.10%	139.77	2.410	Directly copied from [24]
LBGAT [12]	88.22%	52.18%	140.40	2.706	Provided model
CIFAR-100 - $\varepsilon = 8/255$					
AT [37]	59.59%	22.86%	82.45	-3.268	Our reimplementation
AIT [69]*	69.51%	2.80%	72.31	-7.403	Provided model
CAT [5]	62.84%	16.82%	79.66	-5.487	Directly copied from [52]
FS [68]*	73.94%	0.04%	73.98	-4.728	Provided model
FAT [71]	65.51%	21.17%	86.68	-0.618	Provided model
TRADES [70]	56.50%	26.87%	83.37	-0.449	Directly copied from [12]
SAT [52]	62.95%	24.56%	87.51	0.074	Directly copied from [52]
BS [8]	62.15%	26.94%	89.09	1.549	RobustBench
IAD [72]	60.72%	27.89%	88.61	1.687	Directly copied from [72]
AWP [63]	60.38%	28.86%	89.24	2.424	Directly copied from [63]
S^2O [24]	63.40%	27.60%	91.00	2.786	Directly copied from [24]
LAS-AT [23]	61.80%	29.03%	90.83	3.189	Directly copied from [23]
LBGAT [12]	70.25%	26.73%	96.98	6.639	Provided model

B. Detail of ARDist

As described in Subsection 5.1, ARDist approximates a curve representing the accuracy-robustness tradeoff. The red dashed lines in Figs. 1 and 3 show the approximated curves. ARDist evaluates the mitigation by calculating the distance between the approximated curve and a point given by the method being evaluated. We can calculate the distance by finding the approximated curve's normal through the evaluated point. This can be done with numerical calculation. The source codes for the calculation of ARDist are as follows. One can easily use these codes by only setting the standard and AutoAttack accuracies of a method that is required evaluation.

```
## ARDist for CIFAR-10 ##
from scipy import optimize, exp
import numpy as np

p=90.24 # please set standard accuracy
q=50.20 # please set AutoAttack accuracy

def f(x):
    return 9.877e-05*x**3 - 0.3922*x**2 + 63.82*x - 2600

def df(x):
    return 9.877e-05*3*x**2 - 0.3922*2*x + 63.82

def h(x):
    return q-f(x) + (p-x)/df(x)

x_sol = float(optimize.fsolve(h,90))
sign = np.sign(q-f(p))
print("x = " + str(x_sol))
print("y = " + str(f(x_sol)))
print("Sign: " + str(sign))
print(sign * np.sqrt((p-x_sol)**2+(q-f(x_sol))**2))
```

```
## ARDist for CIFAR-100 ##
from scipy import optimize, exp
import numpy as np

p=73.05 # please set the standard accuracy
q=24.32 # please set the AutoAttack accuracy

def f(x):
    return 0.0005615*x**3 - 0.1582*x**2 + 12.44*x - 271.8

def df(x):
    return 0.0005615*3*x**2 - 0.1582*2*x + 12.44

def h(x):
    return q-f(x) + (p-x)/df(x)

x_sol = float(optimize.fsolve(h,70))
sign = np.sign(q-f(p))
print("x = " + str(x_sol))
print("y = " + str(f(x_sol)))
print("Sign: " + str(sign))
print(sign * np.sqrt((p-x_sol)**2+(q-f(x_sol))**2))
```

C. Results with Other Distance Functions

In the experiment, we used the angular distance as $d(\cdot)$ in RGKD. However, RGKD also performs well with other distance functions. Table 7 lists the results of RGKD depending on the distance functions. As shown, using the mean squared error (MSE) and mean absolute error (MAE) achieves almost the same performance of the angular distance. Since using the angular distance achieves slightly better than the other two functions, we chose it in the experiments.

Table 7. Comparison of three distance functions in RGKD.

	Standard	AutoAttack
MSE	88.34%	50.06%
MAE	88.31%	50.13%
Angular distance	**88.52%**	**50.20%**

D. Comparison with TRADES and S^{2}O using Varied Hyperparameters

As discussed in Section 2, TRADES [70] can achieve various accuracy-robustness tradeoffs by adjusting the hyperparameter β. Although we used the results of TRADES with $\beta = 6.0$ for comparison in Figs. 1 and 3, it is also important to compare ARREST with TRADES when other hyperparameters are used. Here, we compared ARREST with not only TRADES but also a recent state-of-the-art method called S^{2}O [24], which is integrated with TRADES. We set three values $\{0.5, 1.0, 6.0\}$ as the hyperparameter β. Figures 7 and 8 show the results for CIFAR-10 and CIFAR-100, respectively. In these figures, the results of ARREST clearly appear on the right side relative to those of TRADES and S^{2}O, thus indicating that ARREST can achieve higher standard accuracy while maintaining the same robustness as these methods. Finally, in both figures, we also show the approximated curve of the tradeoff for ARDist, the same as in Figs. 1 and 3. As we can see, these curves are in good agreement with the results of ARREST and S^{2}O.

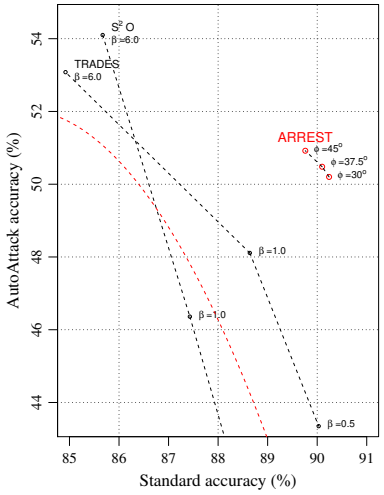

Figure 7. Comparison results between ARREST, TRADE, and S^{2}O for CIFAR-10.

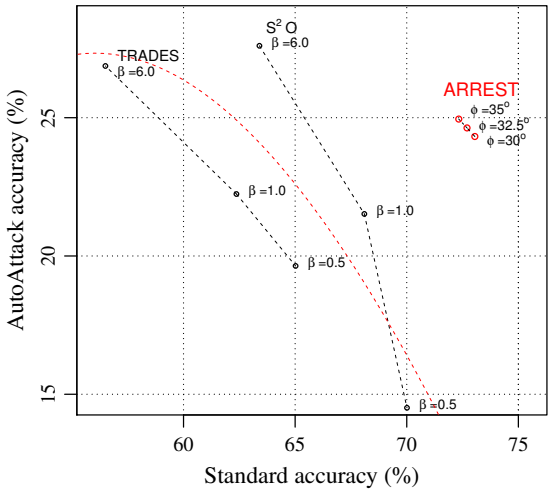

Figure 8. Comparison results between ARREST, TRADE, and S^{2}O for CIFAR-100.

E. Results of Other Attacks

In Section 5, we mainly evaluated the robustness using AutoAttack [11] since it is common and reliable. In this appendix, we evaluated the robustness of ARREST using other attacks. Table 8 lists the results obtained with four variations of our method (only AFT, AFT with RGKD, AFT with NR, and all of them, *i.e.*, ARREST) and two baselines (standard training and AT). We evaluated them using four types of adversarial attacks: FGSM [15], PGD (7 and 30) [37], and CW (30) [6]. The number in parentheses indicates the number of iterative steps for each attack. As seen in Table 8, the robustness trend of the variations of our method is consistent with those in Table 2. Namely, ARREST also exhibits high adversarial robustness against various attacks, not only AutoAttack.

Table 8. Results obtained with four variations of our method (only AFT, AFT with RGKD, AFT with NR, ARREST) and two baselines (standard training and AT). Evaluation of robustness is done with FGSM [15], PGD (7 and 30) [37], and CW (30) [6]. Number in parentheses indicates number of iterative steps for each attack.

	ResNet-18					WideResNet-34-10				
	Standard	FGSM	PGD (7)	PGD (20)	CW (30)	Standard	FGSM	PGD (7)	PGD (20)	CW (30)
ST	**94.5**%	28.3%	0%	0%	0%	**95.4**%	38.6%	0%	0%	0%
AT [37]	84.7%	56.0%	50.6%	46.7%	46.9%	87.1%	56.1%	50.0%	45.8%	46.8%
AFT	84.1%	56.2%	52.4%	49.2%	48.0%	87.5%	60.0%	55.0%	51.3%	51.3%
AFT + RGKD	85.1%	57.4%	53.0%	**49.6**%	49.0%	88.5%	61.5%	56.0%	52.3%	52.9%
AFT + NR	85.5%	56.5%	52.4%	49.0%	48.1%	88.9%	60.1%	54.7%	51.0%	51.4%
ARREST	86.6%	**57.7**%	**53.3**%	49.4%	**49.3**%	90.2%	**63.0**%	**56.8**%	**52.4**%	**53.4**%

F. Detail of Other Knowledge Distillation Methods

In this appendix, we formulate the other knowledge distillation methods used in Table 4. We used two methods for our comparisons, (i) logit and (ii) attention map. Hereafter, we formulate them in order.

First, logit [12] guides the DNN with the final output of the pretrained DNN θ_{s}^{*}. It replaces $\mathcal{L}_{\mathrm{RGKD}}$ in Eq. (4) to $\mathcal{L}_{\mathrm{logit}}$ that is formulated as

$$\mathcal{L}_{\mathrm{logit}}(\boldsymbol{x}, \boldsymbol{\delta}, \theta_{\mathrm{r}}) = d\left(f(\boldsymbol{x}+\boldsymbol{\delta}; \theta_{\mathrm{r}}), f(\boldsymbol{x}; \theta_{\mathrm{s}}^{*})\right). \tag{6}$$

We set the hyperparameter $\lambda = 1$ in accordance with Cui *et al*. [12].

Second, attention map [58] guides the DNN with a spatial attention map computed from the latent representations [67]. It replaces $\mathcal{L}_{\mathrm{RGKD}}$ in Eq. (4) to $\mathcal{L}_{\mathrm{AT}}$ that is formulated as

$$\mathcal{L}_{\mathrm{AT}} = d(\mathrm{AT}(h(\boldsymbol{x}+\boldsymbol{\delta}; \theta_{\mathrm{r}}), \mathrm{AT}(h(\boldsymbol{x}; \theta_{\mathrm{s}}^{*}))). \tag{7}$$

$$\text{Here, } \mathrm{AT}(A) = \frac{F(A)}{||F(A)||_2}, \ F(A) = \sum_{i}^{C} |A_i|. \tag{8}$$

Note that this loss function assumes that the latent representations $h(\boldsymbol{x}+\boldsymbol{\delta}; \theta_{\mathrm{r}})$ and $h(\boldsymbol{x}; \theta_{\mathrm{s}}^{*})$ are obtained from convolutional network [18, 28, 51, 55, 66], and tensors in $\mathbb{R}^{C\times H\times W}$. A_i ($\in \mathbb{R}^{H\times W}$) indicates the ith tensor along C (channel) direction in the representation. Therefore, in Eq. (8), $F(A)$ means simply summing up the representation along C direction, and $\mathrm{AT}(A)$ normalizes it. We set the hyperparameter $\lambda = 2$ in accordance with Wang *et al*. [58].

We used the same experimental settings in AFT and RGKD except for the settings described above.

G. Detail of Optimization with Additional Examples

In this appendix, we explain the setup of our experiments with additional examples (listed in Table 5). In these experiments, we utilized the framework in RST [43]. The experimental settings were mostly the same as those explained in Subsection 5.2. Thus, here, we describe only the settings that differed from those settings.

In the experiments, we utilized the additional unlabeled examples that were originally provided by Carmon *et al*. [7]. These examples were pseudo labeled in advance [7], and we used them as ground-truth labels. The batch size was set to 256, and the fraction of unlabeled examples in the batch was 0.5, *i.e*., 128 unlabeled examples were used in each batch. In RST in Table 5, we set the number of training epochs to 200. The learning rate started at 0.1 and then decayed by $\times 0.1$ with transition epochs $\{150, 180\}$. These settings follow Raghunathan *et al*. [43]. In ARREST, we set the number of training epochs to 100. The learning rate started at 0.025, decayed to 0.02 at 50 epochs, and then decayed by half every 10 epochs thereafter.

Finally, in accordance with Raghunathan *et al*. [43], $\mathcal{L}_{\mathrm{CE}}$ is calculated with not only adversarial but also clean examples in both RST and ARREST. The resulting loss function is denoted as $\mathcal{L}_{\mathrm{CE\_RST}}$,

$$\mathcal{L}_{\mathrm{CE\_RST}}(\boldsymbol{x}, \boldsymbol{\delta}, y, \theta_{\mathrm{r}}) = 0.5\, \mathcal{L}_{\mathrm{CE}}(f(\boldsymbol{x}+\boldsymbol{\delta}; \theta_{\mathrm{r}}), y) + 0.5\, \mathcal{L}_{\mathrm{CE}}(f(\boldsymbol{x}; \theta_{\mathrm{r}}), y). \tag{9}$$

We replaced $\mathcal{L}_{\mathrm{CE}}$ with $\mathcal{L}_{\mathrm{CE\_RST}}$ in optimization. Note that the PGD objective function was maintaining $\mathcal{L}_{\mathrm{CE}}(f(\boldsymbol{x}+\boldsymbol{\delta}; \theta_{\mathrm{r}}), y)$ alone, without $\mathcal{L}_{\mathrm{CE}}(f(\boldsymbol{x}; \theta_{\mathrm{r}}), y)$.