Adversarial Robustness via Runtime Masking and Cleansing: Supplementary Materials

In this document, we give more details about our experiments and the behavior of RMC. The code of our experiments is available at `https://github.com/nthu-datalab/Runtime-Masking-and-Cleansing`.

A Training Details

This section details the trained models and the local-adaptation procedures of RMC for different datasets.

A.1 MNIST.

Training. We use Adam as the optimizer. We set the learning rate to 1e$-$3 for all models paired up with different train-time defense mentioned in Section 4 of the main paper. For the adversarial training, we set the batch size to 128, and the maximum allowable perturbations, denoted by ϵ, to 0.3. When the attack model is PGD in adversarial training, we run PGD for 10 iterations with step size 0.01 to generate an adversarial example. For regularization-based defenses, we set the weights of the regularization terms to 0.1 and 0.05 for Jacobian and Cross-Lipschitz regularizations, respectively.

Adaptation. We follow the settings used by the training procedure of the corresponding model, except setting the learning rate to 2e $-$ 4.

A.2 CIFAR-10.

Training. We use Adam with a learning rate of 1e $-$ 4 as the optimizer and set the batch size to 128 for all types of training methods. For the PGD attack model, we set $\epsilon = 8/255$ and iterate it for 10 steps with the step size 2/255 to generate an adversarial example. For regularization-based defenses, we set the weights of the regularization terms to 0.001 and 0.05 for Jacobian and Cross-Lipschitz regularizations, respectively.

Adaptation. The settings follow the ones used by the training procedure of the corresponding model, except that the learning rate is reduced to 2.5e$-$5.

A.3 ImageNet.

Training. We use the pretrained ResNet-152 publicly available on the TensorFlow GitHub repository.

Adaptation. We use Adam with a learning rate of 1e−5. For the adversarial training and denoising block (DB), we report their accuracy and robustness from the paper [3]. For other methods, all experiments are conducted with a batch size of 256. Note that the ResNet-152 has batch normalization layers whose means and variances at training time and runtime may be inconsistent because the $\mathbb{N}'(\hat{\boldsymbol{x}})$ in Algorithm 1 of the main paper does not contain i.i.d. samples. To avoid the discrepancy, we fix the means and variances of the batch-normalization layers at the corresponding means and variances recorded in the training phase, respectively, when running the local adaptation procedure of RMC.

B MNIST Experiment

In this section, we show the performance of different defenses under different train-time white-box attacks on MNIST, which was omitted in Section 4 of the main paper due to space limitation.

We set $N' = 2|\mathbb{D}|$, that is, half of the examples in $\mathbb{D}'$ are adversarial examples. For gradient-based attacks (FGSM, BIM, and PGD), the maximum allowable perturbation, ϵ, is set to 0.3. The learning rate for CW-L2 attack is 0.02. Table 1 shows the results. As we can see, the RMC achieves state-of-the-art robustness and performs significantly better than all other baseline defenses in all combinations of the trained networks and attacks. This is consistent with the results on CIFAR-10 shown in Table 1 of the main paper.

C Defense-Aware Attacks

In this section, we further investigate the behavior of RMC under runtime white-box attacks where the $\mathbb{D}'$ and $\mathbb{U}$ can be exploited by an adversary. Unless mentioned specifically, we run the following experiments with a regularly trained network on CIFAR-10 and use the settings described in Sections 4.1 and 5 of the main text.

C.1 PGD-NN2

In addition to the PGD-NN described in Section 5.1 of the main text, we devise another attack, called PGD-NN2, whose goal is to mislead RMC into obtaining $\mathbb{N}'(\hat{\boldsymbol{x}})$ consisting of "wrong" examples by using the knowledge about $\mathbb{D}'$. The PGD-NN2 computes the adversarial perturbation $\boldsymbol{\delta}^*$ for the test instance $\hat{\boldsymbol{x}} \in \mathbb{U}$ using

$$\boldsymbol{\delta}^* = \arg\min_{\boldsymbol{\delta}\in\mathbb{A}(\hat{\boldsymbol{x}})} H(\tfrac{1}{K}\textstyle\sum_{\boldsymbol{x}\in\mathbb{N}(\hat{\boldsymbol{x}}+\boldsymbol{\delta})} f(\boldsymbol{x};\boldsymbol{\theta}^*)_{\setminus\mathrm{idx}(\boldsymbol{y})}) - \|f(\boldsymbol{x};\boldsymbol{\theta}^*)_{\setminus\mathrm{idx}(\boldsymbol{y})}\|^2,$$

Table 1: The performance of different defenses under different train-time white-box attacks ($\epsilon = 0.3$) on MNIST.

	Acc.	**Robustness**				
		FGSM	**BIM**	**PGD**	**CW-L2**	**JSMA**
Regularly Trained						
None	**99.3**	11.6	0.6	0.5	0.7	14.1
DeepNN	99.2	12.3	0.6	0.5	75.3	58.2
WebNN	98.2	70.4	82.6	85.3	87.4	87.1
RMC	**99.3**	**99.3**	**99.3**	**99.3**	**99.3**	**99.1**
Adversarially Trained w. FGSM						
None	99	94	51.4	0.7	16.3	42.9
DeepNN	98.8	94	56.9	1.7	85.9	77.2
WebNN	98.6	94.3	85.2	90.8	89.1	87.9
RMC	**99.2**	**98.6**	**98.9**	**98.9**	**98.7**	**98.8**
Adversarially Trained w. PGD						
None	99.1	96.6	93	94.8	65.6	94.6
DeepNN	98.8	96.4	94.5	95.8	91	95.4
WebNN	98.7	96.5	94.5	95.8	91	97.5
RMC	**99.2**	**98.2**	**97.5**	**97.8**	**99.1**	**98.9**
Regularly Trained w. Jacobbian Reg.						
None	94.8	22.1	7.6	8	13.7	26.5
DeepNN	95.9	21.1	8.9	9.6	55.7	41
WebNN	94.2	55.5	55.6	58.3	79	66.4
RMC	**99.3**	**98.9**	**98.9**	**99.1**	**99.2**	**98**
Regularly Trained w. Cross-Lipschitz Reg.						
None	**99.3**	70.6	30.7	19.3	23.8	48.6
DeepNN	99.2	73.2	37.5	22.3	72.7	73.4
WebNN	97	79.8	75.1	74.4	82.8	85.5
RMC	**99.3**	**99.2**	**99.2**	**99.3**	**99.2**	**98.2**

Table 2: The robustness of runtime defenses under the PGD-NN and PGD-NN2 attacks.

	PGD	**PGD-NN**	**PGD-NN2**
None	6.7	11.3	59.6
DeepNN	8	11.4	59.7
WebNN	48.6	1.5	7.4
RMC	**86.6**	**75.4**	**79**

Table 3: The robustness of RMC under the (a) PGD-Skip-Delayed and (b) PGD-Skip-Partial attacks, which degenerate to PGD-Skip when $q = 0$ and when $\mathbb{U}$ is 100% known, respectively.

q	**0**	**50**	**100**
$p = 100$	14.9	19.8	20.8

(a) PGD-Skip-Delayed

known	**30%**	**50%**	**70%**	**100%**
$p = 100$	18.1	16.4	16.5	14.9

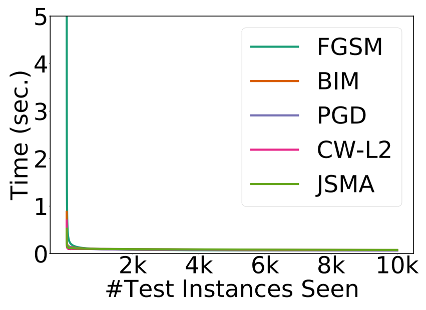
(b) PGD-Skip-Partial

where $H(\cdot)$ is the entropy function and $[\cdot]_{\backslash\mathrm{idx}(\boldsymbol{y})}$ is the normalized vector of the original $[\cdot]$ excluding the dimension that the one-hot label vector $\boldsymbol{y}$ indicates.[1] We then add the perturbation $\boldsymbol{\delta}^*$ back to $\hat{\boldsymbol{x}}$. The perturbed $\hat{\boldsymbol{x}}$ would let $\mathbb{N}(\hat{\boldsymbol{x}})$ contain examples of the same class but $\boldsymbol{y}$. One can think PGD-NN2 as a "targeted" analogy to the "non-targeted" PGD-NN presented in the main paper. Table 2 shows the performance of different runtime defenses RMC against this attack. We can see that the RMC still significantly outperforms the DeepNN and WebNN. This suggests that the RMC, unlike the adversarial training, does *not* rely on the correct labels in $\mathbb{N}'(\hat{\boldsymbol{x}})$ to work. Instead, it uses the "messily" labeled adversarial examples in $\mathbb{N}'(\hat{\boldsymbol{x}})$ to cleanse the model of non-robust features.

C.2 PGD-Skip

The PGD-Skip attack, which uses PGD to compute an adversarial point $\hat{\boldsymbol{x}}^{(p+q+1)} \in \mathbb{U}$ (i.e., the $(p+q+1)$-th test input seen at runtime) against the network that has been adapted to $\hat{\boldsymbol{x}}^{(1)}, \cdots, \hat{\boldsymbol{x}}^{(p)} \in \mathbb{U}$, is a strong attack because it bypasses all RMC effects due to $\hat{\boldsymbol{x}}^{(1)}, \cdots, \hat{\boldsymbol{x}}^{(p)}$. The plain RMC (without a changing $\mathbb{D}'$ in Algorithm 2 of the main text) has degraded performance under the PGD-Skip attack, as shown in Table 3. When q (the delay in obtaining an attackable test instance) equals to zero and the instances $\hat{\boldsymbol{x}}^{(1)}, \cdots, \hat{\boldsymbol{x}}^{(p)}$ are 100% known, the attacker can get all information used by the RMC, and the RMC can only defend the attack using one round of local adaptation for $\mathbb{N}(\hat{\boldsymbol{x}}^{(p+q+1)})$. In this extreme case, the RMC gives 14.9% robustness. Although not being very impressive, this

[1] For example, given $\boldsymbol{y} = [0, 0, 1]^\top$, we have $[0.2, 0.3, 0.5]^\top_{\backslash\mathrm{idx}(\boldsymbol{y})} = [0.4, 0.6]^\top$.



MNIST

Figure 1: The inference delay over different numbers of test instances seen by the network at runtime on the MNIST dataset.

number is better than the 6.7% given by the regularly trained network without any runtime defense. Furthermore, the robustness can be improved when the delay q becomes longer or when fewer instances in $\mathbb{U}$ are known by the attacker.

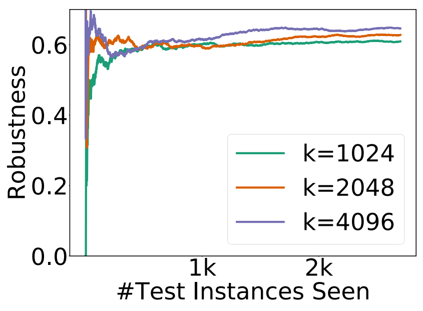
D More Experiments

Here, we investigate the behavior of RMC in more dimensions. Unless mentioned specifically, we follow the settings in Section 4 of the main text and use the non-targeted PGD attack with $\epsilon = 0.3$ and $8/255$ on the MNIST and CIFAR-10 datasets, respectively, while using the targeted PGD attack with $\epsilon = 16/255$ on ImageNet.

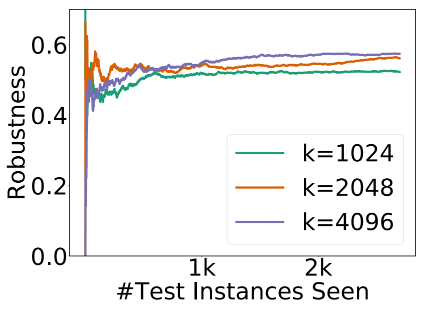
How long is the delay incurred by RMC at runtime on the MNIST dataset? We run RMC on a machine with an NVIDIA V100 GPU. The batch size for local adaptation is set to 128. Figure 1 shows the inference time of RMC on MNIST. The delay of making an inference is around 0.1 second under all types of attacks. Together with the results in Figure 2 in the main paper, the RMC can be a practical, high-performing defense for non-realtime applications.

How does the hyperparameter K affect the performance of RMC on ImageNet? We empirically study the effect of the hyperparameter K on the ImageNet dataset. Figure 2 shows the results. Similar to what we have seen in Figure 3 in the main paper, a larger K leads to less warm-up time and higher long-term robustness. Furthermore, all the values of K here give satisfactory robustness but are relatively small as compared to the ones used in WebNN [1], which justifies the practical advantage of RMC.

Can RMC withstand non-targeted attacks on ImageNet? Most of the existing defenses evaluate the robustness of a model on ImageNet by considering only the targeted attacks. However, a recent study [2] points out that the performance of the adversarial training could drop significantly under a non-targeted attack. To see whether the RMC can withstand a non-targeted attack, we compare the performance of RMC under targeted and non-targeted attacks



(a)

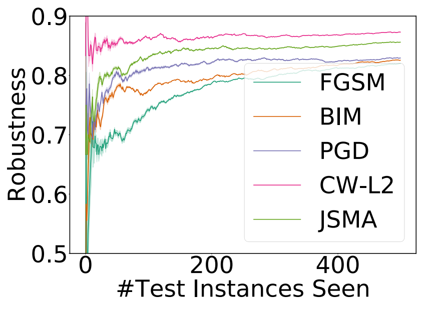


(b)

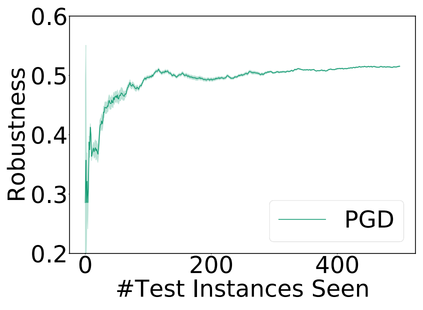
Figure 2: Sensitivity of performance to the hyperparameter K on ImageNet with the maximum allowable perturbation (a) $\epsilon = 8/255$ and (b) $\epsilon = 16/255$.

Table 4: The performance of different defenses under targeted and non-targeted train-time white-box attacks ($\epsilon = 16/255$) on ImageNet.

	Acc.	**Robustness**	
		Targeted	**Non-targeted**
None	72.9	5.2	8.6
Adv. Trained	62.3	52.5	17.1
DB	65.3	55.7	N/A
DeepNN	26.6	8.7	10.8
WebNN	27.8	15.2	19.9
RMC	**73.6**	**55.9**	**37.6**



(a) CIFAR-10



(b) ImageNet

Figure 3: Variance of RMC on the CIFAR-10 and ImageNet datasets. The solid lines represent the mean robustness of RMC with 6 different runs, while the translucent areas are the variance.

on ImageNet. Table 4 shows the results. We use PGD with $\epsilon = 16/255$ as the attack model. As we can see, the performance of RMC drops under the non-target attack. However, it still achieves the state-of-the-art robustness and performs much better than the other baselines. Note that both the DeepNN and WebNN give relatively better performance under the non-targeted attack. In particular, the WebNN achieves better performance than the adversarial training. The runtime defenses seem to be a promising direction of fighting against the non-targeted attacks in a large-scale learning problem, which deserves in-depth study in the future.

What are the variances of the reported performance? We report the performance of RMC by averaging the results of 6 runs. In general, we observe small variances at later runtime stages and therefore omit them in the main paper for simplicity. The shaded areas in Figure 3 show the variances of the reported robustness under different attacks on the CIFAR-10 and ImageNet datasets. As we can see, the variances are small except at the warm-up stage. On both the datasets, the RMC gives a stable performance after seeing about 200 test data points.

References

[1] Abhimanyu Dubey, Laurens van der Maaten, Zeki Yalniz, Yixuan Li, and Dhruv Mahajan. Defense against adversarial images using web-scale nearest-neighbor search. In *Proc. of CVPR*, 2019.

[2] Jianyu Wang and Haichao Zhang. Bilateral adversarial training: Towards fast training of more robust models against adversarial attacks. In *Proc. of ICCV*, 2019.

[3] Cihang Xie, Yuxin Wu, Laurens van der Maaten, Alan L Yuille, and Kaiming He. Feature denoising for improving adversarial robustness. In *Proc. of CVPR*, 2019.