A. Defenses and Evaluation Configurations

A.1. Attack Configurations

We use PGD, BPDA, and AutoAttack for the evaluation on CIFAR-10. PGD uses 200 update iterations, and BPDA and AutoAttack use 100 update iterations. All the attacks use 20 EOT samples. The step size of PGD and BPDA is 0.007. For randomized defenses, such as DiffPure [24], we use the random version of AutoAttack, and for static defenses, such as SODEF [19] and DISCO [16], we use the standard version. For diffusion-based purification methods, following the settings in DiffPure, we use a fixed subset of 512 randomly sampled images for all experiments.

A.2. Diffusion-Based Purification

Diffusion-based purification methods follow the algorithm proposed by DiffPure [24]. Diffusion-based purification partialy utilizes the forward and denoising processes. Algorithm 1 displays the complete forward process of diffusion-based purification. Using the "notable property" of the forward process [17], we can sample $\mathbf{x}_{t^*}$ in a single step:

$$q(\mathbf{x}_{t^*}|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t^*}; \sqrt{\alpha_{t^*}}\mathbf{x}_0, (1-\alpha_{t^*})\mathbf{I}), \tag{7}$$

where $\alpha_{t^*} := \prod_{i=1}^{t^*}(1-\beta_i)$. Algorithm 2 displays the complete denoising process of diffusion-based purification. Song et al. [33] find that denoising process can be accelerated by a linearly increasing sub-sequence $\{\tau_0, \ldots, \tau_s\}$ of $[0, \ldots, t^*]$ with $\tau_0 = 0$ and $\tau_s = t^*$ where $s \leq t^*$. Throughout all experiments, we only consider sub-sequences having uniform step size. Given $\sigma_{\tau_i}(\eta) = \eta\sqrt{(1-\alpha_{\tau_{i-1}})/(1-\alpha_{\tau_i})}\sqrt{1-\alpha_{\tau_i}/\alpha_{\tau_{i-1}}}$ for all timesteps, the denoising process is DDPM when $\eta = 1$ and DDIM when $\eta = 0$.

Algorithm 1 Forward process of diffusion-based purification

1: **Input**: image $\mathbf{x}_0$, maximum timestep t^*
2: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
3: $\mathbf{x}_{t^*} = \sqrt{\alpha_{t^*}}\mathbf{x}_0 + \sqrt{1-\alpha_{t^*}}\boldsymbol{\epsilon}$
4: **Return** $\mathbf{x}_{t^*}$

Algorithm 2 Denoising process of diffusion-based purification

1: **Input**: noisy image $\mathbf{x}_{t^*}$, timestep schedule $\{\tau_0, \tau_1, \ldots, \tau_s\}$
2: **for** $i = s, \ldots, 1$ **do**
3: $\quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\quad \mathbf{x}_{\tau_{i-1}} = \sqrt{\alpha_{\tau_{i-1}}}\left(\frac{\mathbf{x}_{\tau_i} - \sqrt{1-\alpha_{\tau_i}}\epsilon_\theta(\mathbf{x}_{\tau_i})}{\sqrt{\alpha_{\tau_i}}}\right) + \sqrt{1-\alpha_{\tau_{i-1}}-\sigma_{\tau_i}^2}\cdot\epsilon_\theta(\mathbf{x}_{\tau_i}) + \sigma_{\tau_i}\boldsymbol{\epsilon}$
5: **end for**
6: **Return** $\mathbf{x}_0$

A.3. ADP [39]

ADP uses a score-based model trained with denoising score matching:

$$\mathbb{E}_t\mathbb{E}_{q(\mathbf{x}_t|\mathbf{x})p_{\text{data}}(\mathbf{x})}\left[\frac{1}{2}\|\mathbf{s}_\theta(\mathbf{x}_t) - \nabla_{\mathbf{x}_t}\log q(\mathbf{x}_t|\mathbf{x})\|^2\right], \tag{8}$$

where p_{data} is a data distribution, t is a scale of perturbation, and $\mathbf{s}_\theta$ is a score-based model. To clean an adversarial example, ADP uses a deterministic version of Langevin dynamics:

$$\mathbf{x}_t = \mathbf{x}_{t-1} + \alpha_{t-1}\mathbf{s}_\theta(\mathbf{x}_{t-1}), \tag{9}$$

where α_{t-1} is a step size. Before its purification process, ADP adds Gaussian noise to the input, which can improve robustness. In its original implementation, ADP is evaluated on BPDA. However, since ADP uses up to eight steps for purification, calculating the full gradients of the defense is possible. Therefore, we use the full gradients to generate adversarial examples in our evaluation.

A.4. DiffPure [24]

Given a forward diffusion process $\mathbf{x}(t)_{t\in[0,1]}$, DiffPure uses $t^* = 0.1$ and $t^* = 0.075$ on CIFAR-10 against threat models $\ell_\infty(\epsilon = 8/255)$ and $\ell_2(\epsilon = 0.5)$, respectively (with step size 0.001). Since it is impossible to calculate the gradients using back-propagation, in the original evaluation, DiffPure uses an adjoint method of its underlying numerical SDE (ODE) solver for calculating gradients. In our evaluation, we use gradients of a surrogate process calculated by direct back-propagation. To overcome memory constraints, we increase the step size of the surrogate denoising process in attack to 0.005.

A.5. GDMP [36]

The basic approach of GDMP is the same as DiffPure [24], however, it uses two additional techniques: guidance and multiple purification steps. GDMP proposes to use gradients of a distance between an initial input and a target being processed to preserve semantic information:

$$\mathbf{x}_{t-1} \sim \mathcal{N}(\boldsymbol{\mu}_\theta - s\boldsymbol{\Sigma}_\theta \nabla_{\mathbf{x}_t} \mathcal{D}(\mathbf{x}_t, \mathbf{x}_{\text{adv},t}), \boldsymbol{\Sigma}_\theta), \tag{10}$$

where $\boldsymbol{\mu}_\theta$ and $\boldsymbol{\Sigma}_\theta$ are the mean and variance of $\mathbf{x}_{t-1}$ calculated by diffusion models $\boldsymbol{\epsilon}_\theta$, s is a scale of guidance, $\mathbf{x}_t$ is a sample that is being purified, and $\mathbf{x}_{\text{adv},t}$ is a noisy adversarial example. In addition, GDMP finds that iteratively applying the purification process, which we call the purification step, can improve the robustness. GDMP consists of four purification steps, each consisting of 36 forward steps and 36 denoising steps. In our evaluation, we use a surrogate process calculated by direct back-propagation, while GDMP is originally evaluated on BPDA. Since it is impossible to calculate the gradients of the full defense process, we use a surrogate process consisting of four purification steps (each consisting of 36 forward steps and six denoising steps) in the attack.

A.6. SODEF and DISCO [19, 16]

SODEF [19] uses an ODE block that satisfies Lyapunov stability. Lyapunov-stable equilibrium point has a property that its neighborhood gathers to that point by passing through the ODE block. In the original implementation, SODEF uses an adjoint method to calculate its gradients of the ODE block. In our experiment, we use back-propagation instead of the adjoint method.

DISCO [16] employs a local implicit module to restore a clean example. For every pixel, the module estimates the original RGB values of input before sending it to the classifier. While DISCO is evaluated on BPDA in the original evaluation, we use the full gradients of the defense process. To evaluate both SODEF and DISCO, we use the standard version of AutoAttack.

B. Additional Results on Evaluation for Diffusion-Based Purifications

B.1. Additional Results on RQ1

We show the difference between attack success rate when using the adjoint method and back-propagation. Table 12 shows the robust accuracy of DiffPure [24] and its probability flow ODE [35] against PGD+EOT $\ell_\infty(\epsilon = 8/255)$ on CIFAR-10. For both DiffPure and its probability flow ODE, the back-propagation can generate adversarial examples more successfully than the adjoint method. For the probability flow ODE, the back-propagation with step size 0.01 has 34.51% lower robust accuracy than the adjoint method.

B.2. Additional Results on RQ2

Figure 6 compares the attack performance of PGD and AutoAttack $\ell_\infty(\epsilon = 8/255)$ on CIFAR-10. This result is conducted on the same settings with Section 4. As shown in the Figure 6, PGD generally has a larger attack success rate than AutoAttack. As the number of forward steps increases, the gap between the two attacks also increases. Therefore, PGD+EOT is a more appropriate attack than AutoAttack for evaluating diffusion-based purification methods.

C. Additional Results on Analysis of Hyperparameters

In this section, we provide further analyses of hyperparameters on CIFAR-10 and ImageNet. In all experiments, the implementation of diffusion-based purification follows noise scheduling, the forward process, and the denoising process of Appendix A.2.

Defense	Method	Step Size in Attack	Robust Accuracy (%)
DiffPure [24]	Adjoint	0.001	74.38±1.03
	Surrogate	0.005	46.84±1.44
	Surrogate	0.010	50.12±1.18
	Surrogate	0.025	62.93±1.02
Probability flow ODE	Adjoint	0.010	70.47±0.99
	Full	0.010	35.96±0.89
	Surrogate	0.025	36.41±1.30

Table 12: Robust accuracy of DiffPure and its probability flow ODE against PGD+EOT $\ell_\infty(\epsilon = 8/255)$ on CIFAR-10. We compare the attack success rate between the adjoint method and back-propagation. The gradient of the full or surrogate process is calculated by back-propagation. The maximum timestep t^* is set to 0.1. The step size of DiffPure and probability flow ODE in defense is 0.001 and 0.01, respectively.

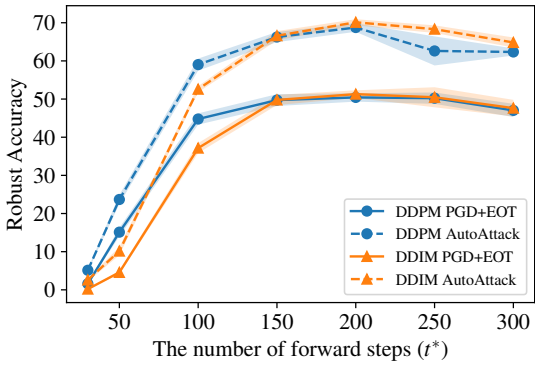
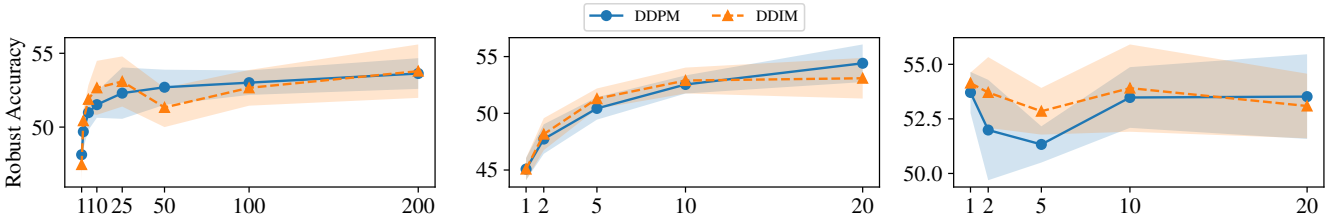


Figure 6: Standard and robust accuracy as we change the number of forward steps against PGD+EOT and AutoAttack $\ell_\infty(\epsilon = 8/255)$ on CIFAR-10. Five denoising steps for both attack and defense are used.

C.1. CIFAR-10

We provide additional results regarding denoising steps and purification steps on CIFAR-10 through Figure 7, Figure 8, and Figure 9. We conduct all experiments in a setting identical to Section 4.3, except that the number of forward steps is 200 (i.e., $t^* = 200$). Furthermore, Figure 10 shows the overall influence of the number of purification steps in attack on the attack success rate.



(a) Defense: The number of denoising steps (b) Adaptive: The number of denoising steps (c) Attack: The number of denoising steps

Figure 7: Robust accuracy as we change the number of denoising steps against PGD+EOT $\ell_\infty(\epsilon = 8/255)$ on CIFAR-10. We change the number of denoising steps in (a) defense, (b) both, and (c) attack for each experiment when the other hyperparameters are fixed. The number of forward steps is 200 (i.e., $t^* = 200$).

C.2. ImageNet

We provide additional results regarding forward, denoising, and purification steps on ImageNet through Figure 11, Figure 12, and Table 13, respectively. In Figure 12 and Table 13, the number of forward steps is set to 200. For ImageNet, we

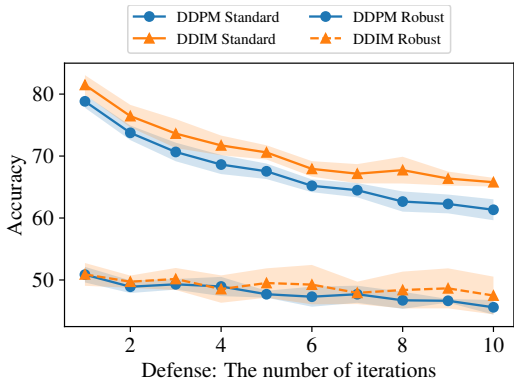


Figure 8: The number of purification steps in defense and its influence on the standard and robust accuracy against PGD+EOT $\ell_\infty(\epsilon = 8/255)$ on CIFAR-10. The number of forward steps is 200 (i.e., $t^* = 200$). The reported robust accuracy is the lowest performance among the various number of purification steps in the attacks.

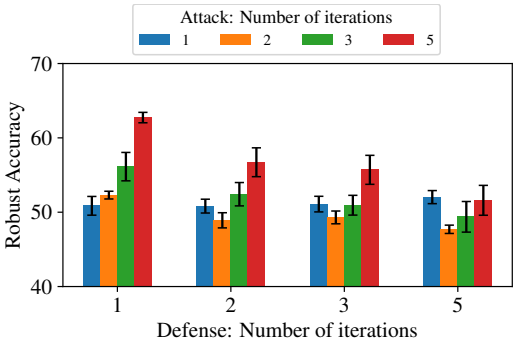


Figure 9: The number of purification steps during the attacks and its influence on the robust accuracy against PGD+EOT $\ell_\infty(\epsilon = 8/255)$ on CIFAR-10. The number of forward steps is 200 (i.e., $t^* = 200$)

.

use 20 PGD iterations and 20 EOT samples. And due to memory constraints, the upper bound on the number of function calls is set to ten. Although we employ the experiments only on DDPM, the results are similar to those from CIFAR-10.

# purification steps in defense	# purification steps in attack	Accuracy (%)	
		Standard	Robust
1	1	64.80±0.70	19.84±0.63
	2		26.84±0.45
2	1	59.06±0.92	27.85±0.71
	2		25.00±1.13

Table 13: Standard accuracy and Robust accuracy against PGD+EOT $\ell_\infty(\epsilon = 4/255)$ on ImageNet. We compare the accuracy between the different number of purification steps in attack and defense.

D. Surrogate Process for Gradual Noise-Scheduling

The defense process of the gradual noise-scheduling have three levels in the number of forward steps. To validate the robustness of our defense, we evaluate the gradual noise-scheduling with several surrogate processes and report the lowest robust accuracy among them. We denote one purification step as (# of forward steps, # of denoising steps). For example, suppose a process uses two purification steps, consisting of 100 forward steps with 20 denoising steps and 120 forward steps with ten denoising steps, respectively. In that case, we denote the defense process as (100, 20), (120, 10). As shown in Table 14, the third surrogate process has the lowest robust accuracy against the gradual noise-scheduling on CIFAR-10. For all experiments of our defense on all datasets, we select the third process as the surrogate process in attacks of the adaptive white-box setting.

E. Memory and Time Requirements for Diffusion-Based Purification

Evaluating diffusion-based purification requires lots of memory since we use direct back-propagation. Figure 13 briefly shows the memory and time requirements of one PGD iteration for implementing diffusion-based purification methods. We use one A100 GPU. For example, we need almost ten days to evaluate one experiment with 30 function calls for calculating back-propagation with one A100 GPU (against a PGD attack using 200 iterations and 20 EOT samples).

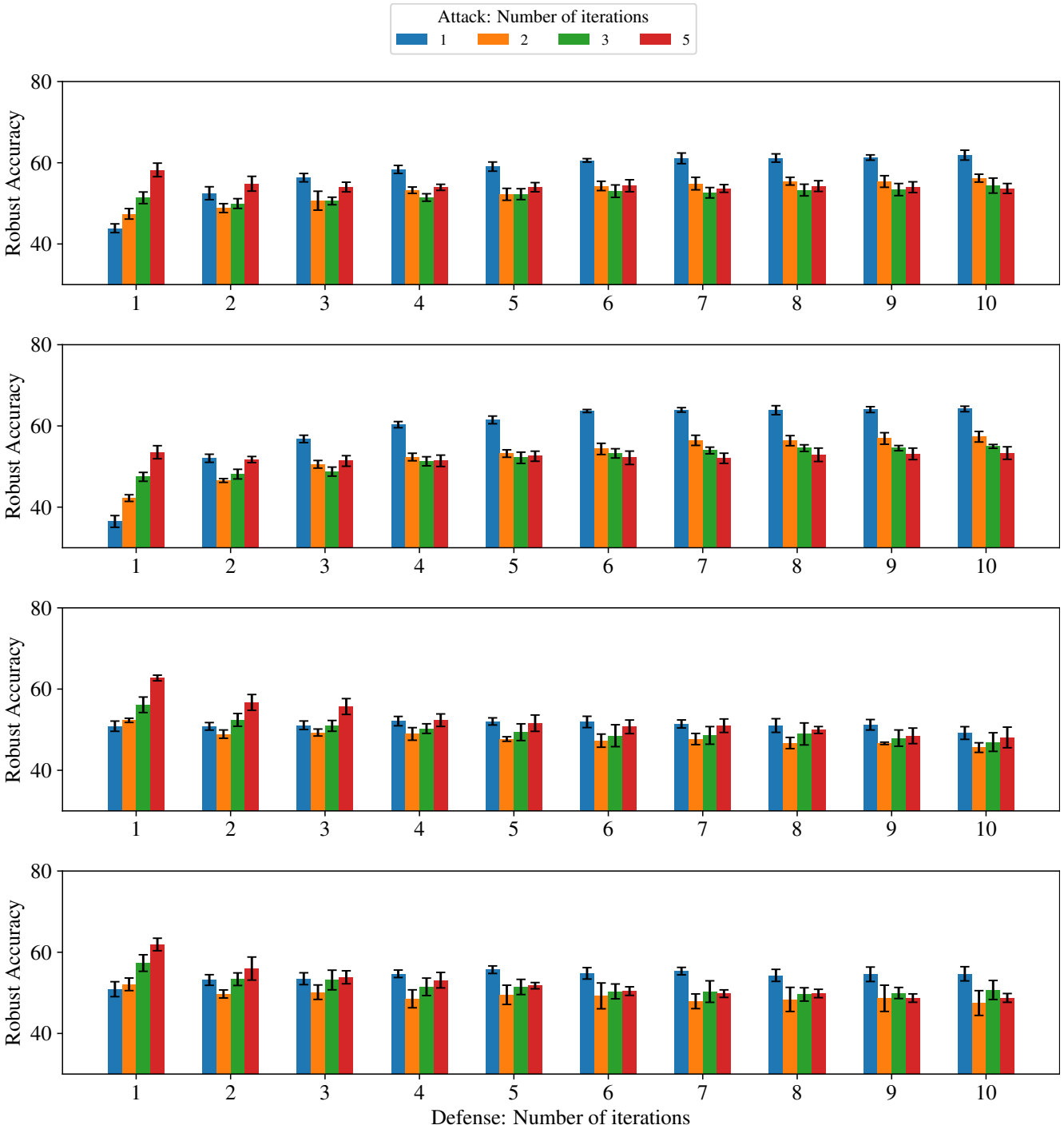


Figure 10: The number of purification steps during defenses and its influence on the robust accuracy against PGD+EOT $\ell_\infty(\epsilon = 8/255)$. We compare the attack success rate with respect to the number of purification steps during attacks. The number of forward steps of the top two rows is 100, and the number of forward steps of the bottom two rows is 200.

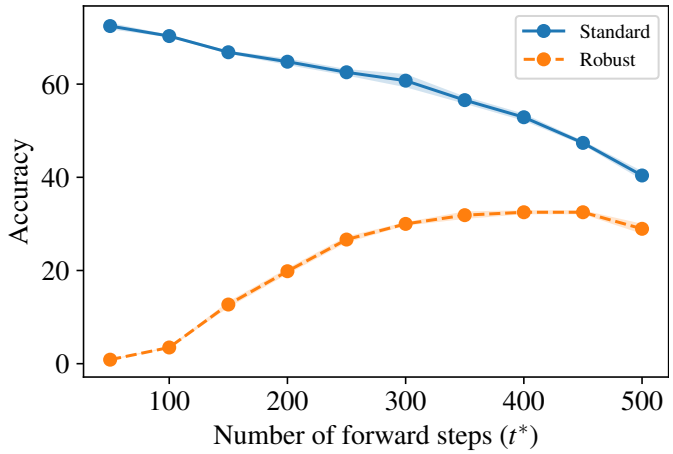


Figure 11: Standard and robust accuracy as we change the number of forward steps against PGD+EOT $\ell_\infty(\epsilon = 4/255)$ on ImageNet. Five denoising steps for both attack and defense are used. The change of total variance ranged from 0.03 to 0.9222.

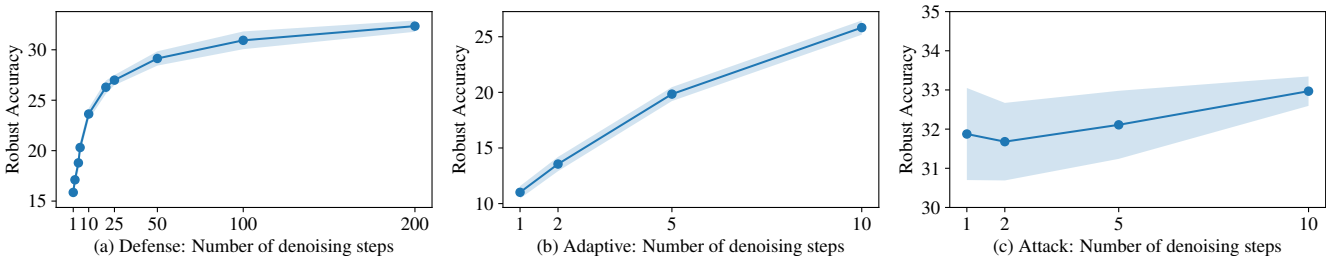


Figure 12: Robust accuracy as we change the number of denoising steps against PGD+EOT $\ell_\infty(\epsilon = 4/255)$ on ImageNet. We change the number of denoising steps in (a) defense, (b) both, and (c) attack for each experiment when the other hyper-parameters are fixed. The number of forward steps is 200 (i.e., $t^* = 200$).

Attack Process	Robust Accuracy (%)
(30, 5), (50, 5), (125, 5), (125, 5)	56.80±1.12
(30, 1), (50, 1), (125, 10)	56.13±1.04
(30, 1), (50, 1), (125, 5)	55.82±0.59
(125, 5), (125, 5)	62.73±0.74
(125, 10)	60.16±1.02
(125, 5)	61.60±0.63

Table 14: Robust accuracy against PGD+EOT $\ell_\infty(\epsilon = 8/255)$ on CIFAR-10. The attack processes are used for generating adversarial examples against our defense strategy.

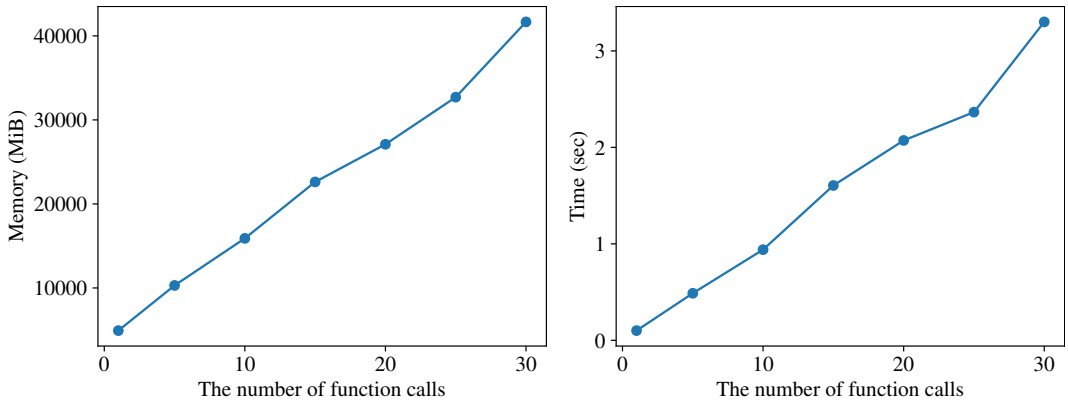


Figure 13: Memory and time usage of diffusion-based purification on CIFAR-10. We use eight for batch size.