Supplementary Document

One Man's Trash is Another Man's Treasure: Resisting Adversarial Examples by Adversarial Examples

A. Additional Experiments and Setups

A.1. Network Structure of f_b

Following the guidelines presented at the end of Sec. 3.4, we choose to use a VGG-style small network in f_b for defining our adversarial transformation. This network is simple enough to enable fast adversarial transformation, while producing the expectation over transformation image (i.e., $\mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$) drastically different from the input image. The structure of f_b for experiments on CIFAR-10 dataset is described in Table 4.

On Tiny ImageNet dataset, the network structure of f_b remains largely the same except two minor changes to accommodate the different resolution of the images in Tiny ImageNet. Namely, the changes are at the 12nd layer (which has a dimension 2048) and the 15th layer (which has a dimension 200).

Layer	Module	Output Size
1	Input	3×32×32
2	Conv(k=3), BN, ReLU	64×32×32
3	MaxPool	64×16×16
4	Conv(k=3), BN, ReLU	128×16×16
5	MaxPool	128×8×8
6	Conv(k=3), BN, ReLU	128×8×8
7	Conv(k=3), BN, ReLU	128×8×8
8	MaxPool	128×4×4
9	Conv(k=3), BN, ReLU	128×4×4
10	Conv(k=3), BN, ReLU	128×4×4
11	MaxPool	128×2×2
12	Flatten	512
13	Linear, ReLU, Dropout	512
14	Linear, ReLU, Dropout	512
15	Linear (output)	10

Table 4. **Network structure for** f_b. Here BN denotes batchnorm operation, and Conv(k=3) denotes convolutional layer with a kernel size of 3.

A.2. Reparameterization Attack

As discussed in Sec. 3.3 and 4.1, to launch the reparameterization attack, we need to find a forward function $h(\cdot)$ that approximate our adversarial transformation process. To this end, we attempted to train a Fully Convlutional Network (FCN) [24], denoted as $h(\boldsymbol{x};\boldsymbol{\theta})$, through the following optimization,

$$\boldsymbol{\theta} = \arg\min_{\boldsymbol{\theta}} \mathbb{E}_{\boldsymbol{x}\in\mathcal{X},\boldsymbol{\delta}\in\boldsymbol{\Delta}} \left\|h(\boldsymbol{x}+\boldsymbol{\delta};\boldsymbol{\theta}) - \tilde{g}_{\boldsymbol{\delta}}(\boldsymbol{x})\right\|^2, \quad (6)$$

where $\mathcal{X}$ is the given dataset, $\boldsymbol{\delta}$ is the initial input perturbation in the L_∞ ball of size Δ (as described in Sec. 3.3), $\tilde{g}_{\boldsymbol{\delta}}(\cdot)$ is the deterministic version of our adversarial transformation $g(\cdot)$: it starts the adversarial search iteration (3) from $\boldsymbol{x}+\boldsymbol{\delta}$ by using a sampled $\boldsymbol{\delta}$.

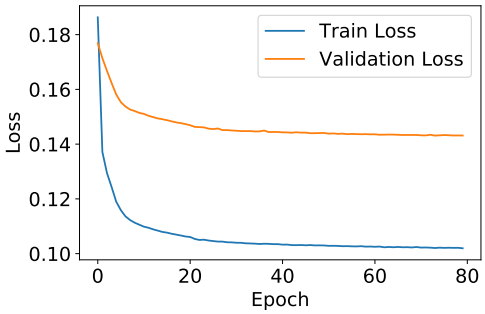

Figure 8. **Training Loss and validation loss** in reparameterization attack.

However, after optimizing (6), we found that although the FCN model can reach a relatively low training error, the error on test set remains high, as depicted in Fig. 8. This suggests that the FCN model is not able to learn a $h(\boldsymbol{x};\boldsymbol{\theta})$ that generalizes well. The inability to generalize is not a surprise: if $h(\boldsymbol{x};\boldsymbol{\theta})$ could generalize well, we would have a direct way of crafting adversarial examples; and PGD-type iterations would not be needed—which are all unlikely.

Indeed, when we use the trained $h(\boldsymbol{x},\boldsymbol{\theta})$ to launch a reparameterization attack to our model, the attack hardly succeeds. Under this attack (on CIFAR-10), the robust accuracy of our defense is 81.1%, even better than the robust accuracy under BPDA-I attack (80.2%). In fact, this accuracy nearly reaches its upper bound, the standard accuracy (i.e., 82.9%), as reported in Table 3 of the main text.

A.3. BPDA Attack Details and Additional Results

As described in Sec. 4.1, we evaluate our defense model under the BPDA attack. To launch BPDA attack, we need to replace the non-differentiable operators in our adversarial transformation with their smooth approximations. In particular, the non-differentiable operators in the adversarial update

Defense Model Iterations	Standard Acc.	Robust Acc. (transfer attack) $N=1$	$N=2$	$N=3$
$N=3$	81.7%	69.7%	66.4%	79.5%
$N=4$	82.4%	75.6%	78.7%	79.9%
$N=5$	83.1%	81.6%	80.7%	81.7%
$N=10$	82.7%	81.4%	81.2%	81.0%
$N=13$	82.9%	80.9%	81.1%	81.3%

Table 5. **Transfer attack based on BPDA.** Each row shows the standard and robust accuracies of our defense model with a different number of LL-PGD steps in its adversarial transformation $g(\cdot)$. The number of LL-PGD steps in our defense model is shown in the left most column. The right most three columns correspond to the models with smaller numbers of LL-PGD steps. We use these models to craft adversarial examples to transfer attack our defense model. It shows that even when the number N of LL-PGD steps in our defense model is moderately large ($N \geq 5$), the transfer attacks become ineffective. In all the evaluations, the perturbation size Δ in our defense model is set as $\Delta = 0.2$.

rule (3) (in the main text) are the $\text{sgn}(\cdot)$ function,

$$\text{sgn}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \boldsymbol{x} > 0, \\ 0, & \text{if } \boldsymbol{x} = 0, \\ -1 & \text{if } \boldsymbol{x} < 0. \end{cases} \tag{7}$$

and the L_∞ projection operator,

$$\Pi_{\boldsymbol{x}' \in \Delta_{\boldsymbol{x}}}(\boldsymbol{x}) = \begin{cases} \boldsymbol{x} & \text{if } |\boldsymbol{x}| \leq \Delta, \\ -\Delta & \text{if } \boldsymbol{x} < -\Delta, \\ \Delta & \text{if } \boldsymbol{x} > \Delta. \end{cases} \tag{8}$$

In Sec. 4.1, we experimented with two different smooth approximations of the $\text{sgn}(\cdot)$ function, namely, the soft sign function $\frac{x}{1+|x|}$ and tanh function $\frac{e^x - e^{-x}}{e^x + e^{-x}}$, and the projection operator is replaced by directly approximating its derivative using (5).

Also discussed in Sec. 4.1 is an additional transfer attack: First, we craft adversarial examples by setting the number N of LL-PGD steps to be a small value. This is motivated by the observation that, as shown in Fig. 5, BPDA attack is able to find effective adversarial examples when N is small. We then use the resulting adversarial examples to transfer attack our defense model, which uses a larger number of LL-PGD steps in the adversarial transformation (in both training and inference). As summarized in Table 5, our experiment shows that this attack remains ineffective to our defense.

A.4. Black-box Transfer Attack

Papernot et al. [30] shows that adversarial examples crafted on one model can effectively transfer to attack another unknown model trained on the same dataset. This phenomenon has been utilized to form a very practical attack method under black-box threat model, because it does not require to query the target model. We also evaluate our defense against the black-box transfer attack method, and the results are shown in Table 7. We use FGSM and PGD to craft the adversarial examples on two ResNet-18 networks. One is trained using the standard method (Vanilla), another one is trained using adversarial training (Madry et al.). We use $\Delta = 0.2$ in our defense. We found that Madry et al. produce better transferability compare to a regular model, but neither of them significantly reduces the accuracy of our defense. We hypothesis this is because our adversarial transformation with a large perturbation (recall Fig. 4) forces the model f_a to learn a decision boundary significantly different from either vanilla or Madry's model, and it is well known that the adversarial transferability between two significantly different models are low.

Method	A_{std}	R_{rob}	Attack Method
No defense	58.2%	0.0%	PGD
Madry et al. [26]	42.7%	17.3%	PGD
Zhang et al. [54]	40.6%	17.7%	PGD
Mao et al. [27]	40.9%	17.5%	PGD
Ours (Under BPDA)	**48.8%**	**47.9%**	BPDA
Ours	**48.8%**	**40.2%**	Transfer

Table 6. **Comparisons on Tiny ImageNet.** The layout of this table is similar to Table 3 in the main text (i.e., the comparisons on CIFAR-10). The perturbation range of all adversarial examples is $\Delta = 0.031$. The last column indicates the most efficient attacking method that produces the worst robustness. The second last row indicates the worst-case robustness of our method under all BPDA-type attacks, while the last row indicates our worst-case robustness under all attacks.

Source Model	A_{std}	FGSM	PGD
ResNet-18 (Vanilla)	82.9%	80.9%	79.2%
ResNet-18 (Madry et al.)	82.9%	80.6%	73.4%

Table 7. **Black-box transfer attack.**

A.5. Evaluation on Tiny ImageNet

We also evaluate our defense model on Tiny ImageNet dataset consisting of 64px×64px RGB images. These images fall into 200 classes, each has 500 images for training and 50 images for testing. Following the evaluations setups in prior works, the adversarial examples used in the attacks have a maximum perturbation size (in L_∞ norm) of 0.031

for pixel values ranging in [0,1]. We use ResNet18 as our classification network (in f_a) and the network structure of f_b is described in Appendix A.1.

We compare our method with the state-of-the-art method [27] evaluated on Tiny ImageNet and the methods based on adversarial training [26, 54]. For all those methods, we use the implementation code provided in their original papers. When comparing with these methods, we use the same training protocol: the models are optimized use SGD (learning rate=0.1, momentum=0.9) and trained for 80 epochs.

As shown in Table 6, our method demonstrates significantly stronger robustness in comparison to previous methods. Our worst-case robust accuracy is **40.2%**. In contrast, previous methods have robust accuracies around **18%**. Remarkably, the standard accuracy of our method also outperforms previous methods.

A.6. Expectation over Transformation Images

Figure 4 in the main text shows a few examples of the difference between an input image $\boldsymbol{x}$ and its expectation over transformation, that is, the image of normalized $\boldsymbol{x} - \mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$. We now provide more samples of $\boldsymbol{x} - \mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$ images on both CIFAR-10 and Tiny ImageNet (see Fig. 9).

Discussion. In [45], Tsipras et al. presented an interesting finding. They visualized the loss gradient with respect to input pixels, and found that if the model is adversarially trained, such a loss gradient is significantly *human-aligned*—they align well with perceptually relevant features (e.g., see Figure 2 in their paper). But if the model is not adversarially trained, the loss gradient appears like random noise. Here, we discover that the normalized difference $\boldsymbol{x} - \mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$ is also human-aligned, exhibiting perceptually relevant features, as shown in Fig. 10. In contrast to the discovery in [45], we found that $\boldsymbol{x} - \mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$ is always human-aligned. Even if the model f_b is not adversarially trained, the difference image $\boldsymbol{x} - \mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$ still exhibits perceptually relevant features, as along as they are trained with sufficient number of epochs (see Fig. 9). If the model f_b is adversarially trained, those perceptually relevant features become more noticeable.

B. Discussion on Computational Performance

Our defense demands lower training cost than the standard adversarial training. For example, on CIFAR-10 dataset, our method takes **82 minutes** to train a ResNet18 model for 80 epochs. This time cost is close to the standard (non-adversarial) training, which takes 56 minutes for the same setting. In contrast, the standard adversarial training takes **460 minutes** for the same number of epochs and the same network structure. Notice that the lower training cost in our method is obtained without sacrificing its robustness performance. In fact, as shown in Table 3 in the main text and Table 6 here, our defense offers much stronger robustness.

The inference cost of our defense is more expensive than adversarially trained models, because the input image $\boldsymbol{x}$ during the inference also needs to be transformed by $g(\cdot)$. In our experiments, our defense takes 17 seconds to predict the labels of 10000 images in CIFAR-10, while the adversarially trained model and the standard model (without adversarial training) both take 4 seconds. This is the cost we have to pay in exchange for stronger robustness. We argue that this is worthy cost to pay because in comparison to network training cost, the inference cost is negligible. In fact, almost all adversarial defense methods that rely on input transformation [14, 39, 33] have a performance overhead at inference time. For example, PixelDefend [39] projects the input to a pre-trained PixelCNN-represented manifold through 100 steps of L-BFGS iterations. Their transformation is about $10\times$ slower than ours even when our method uses the same network structure in f_b as their PixelCNN.

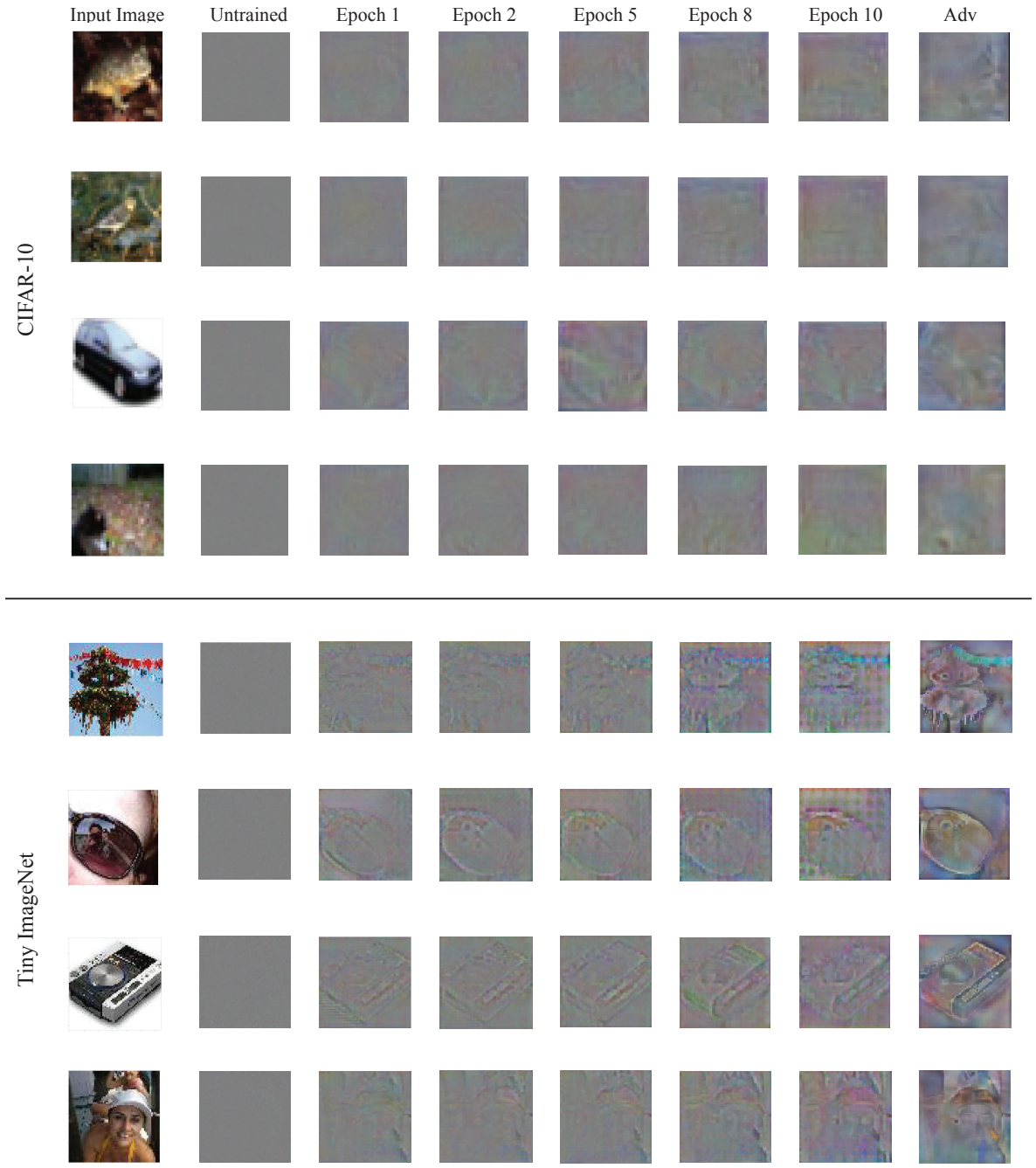

Figure 9. Here we show supplementary examples similar to those in Fig. 4 in the main text. The top four images are the results on CIFAR-10, while the bottom four images are those on Tiny ImageNet. The first column shows the input image $\boldsymbol{x}$ in each example. The other columns show the images generated by adversarial transformations with the f_b models that are untrained, trained with an increasing number of epochs, and adversarially trained, as labeled on the top line. Each of those images is a visualization of the normalized difference $\boldsymbol{x} - \mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$, where the expectation is estimated using 5000 samples. It is evident that as the number of training epochs increases, the expectation over transformation $\mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$ drifts further away from $\boldsymbol{x}$, and the adversarially trained f_b model produces an even larger difference.

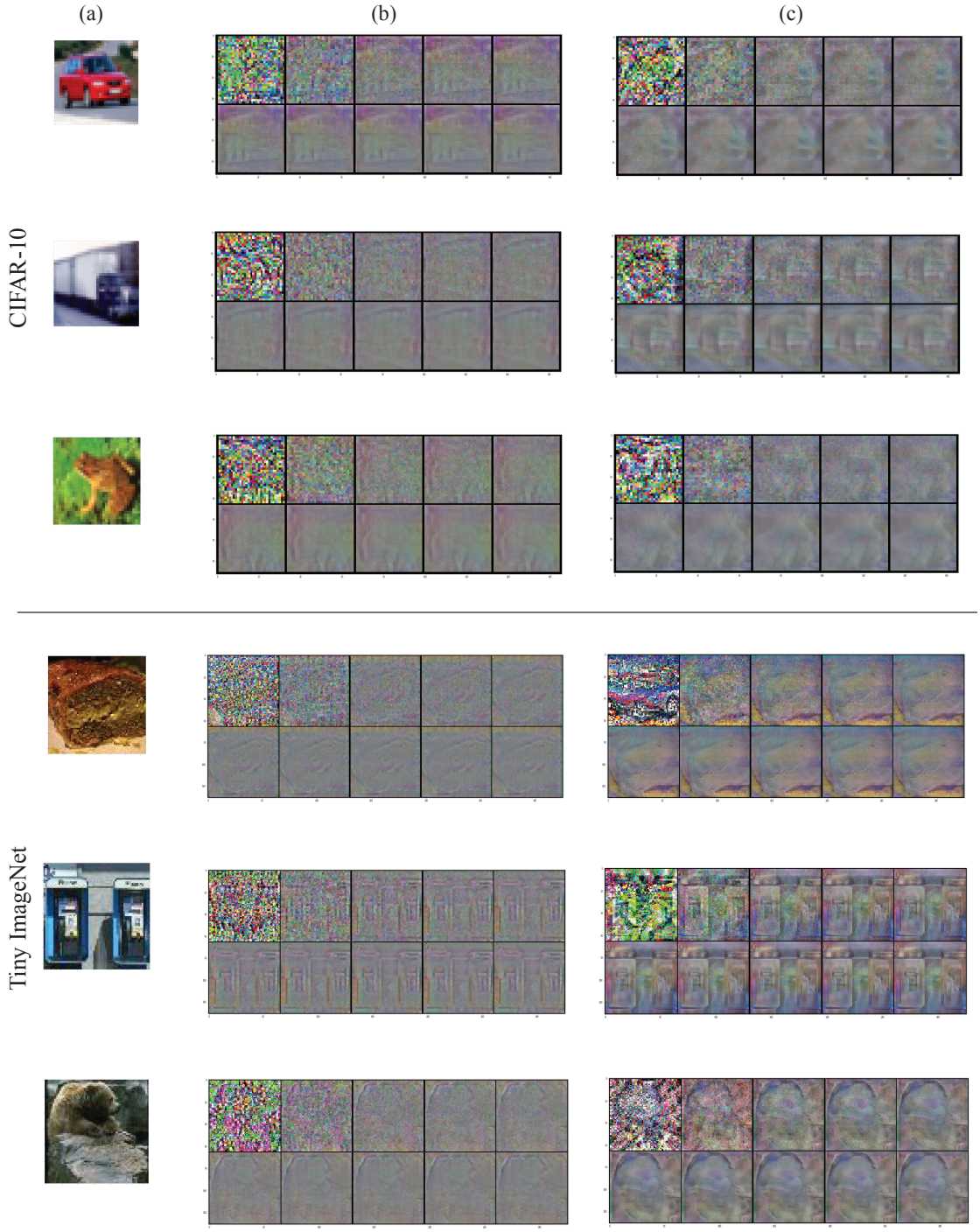

Figure 10. Here we visualize the normalized difference between an input image (shown in column (a)) and its expectation over transformation image $\mathbb{E}_{\tilde{g}\sim\mathcal{T}}\,\tilde{g}(\boldsymbol{x})$ in our defence model. The top three examples are from CIFAR-10, and the bottom three are from Tiny ImageNet. Column (b) shows the results using f_b models with standard training, while column (c) are results with adversarial training. The Expectation over transformation in each example is estimated using an increasing number of samples. The ten sub-images (from left to right, top to bottom) in each group of column (b) and (c) are results in which the expectations over transformation are estimated using 1, 10, 50, 100, 200, 500, 1000, 2000, 5000, 10000 samples of $\tilde{g}(\cdot)$, respectively.