# A. Appendix

## Summary of Appendix

We present the following supplementary materials in Appendix to include more details of our methods, experimental settings, and evaluations.



## A.1. Societal Impact

Our work on backdoor attacks against DMs demonstrates the vulnerability of DMs under such a stealthy and effective trigger design. We hope this work can raise the awareness and understanding of the overlooked deficiency on current DMs development and deployment. The proposed attack, if abused, is likely to impose critical security threats to existing DMs. We believe our study is important and practical as it brings insights of the full capacity of backdoor attacks and will facilitate the future development of powerful defensive algorithms and trustworthy DMs.

## A.2. Detailed Experimental Settings

In Section 4.2, we compare UIBDiffusion's performance with different backdoor attack diffusion models. All experiments are carried out based on dataset CIFAR10. For all the three models, we fine-tune the pre-trained generation model DDPM on CIFAR10 provided by Google. For BadDiffusion [23], we trained the model with learning rate 2e-4, batch size 128, evaluation batch 256 and 50 training epochs. For VillanDiffusion [25], we trained the model with learning rate 2e-4, batch size 128, evaluation batch 1500, training steps 1000, SDE solver and 50 training epochs. We use the same training hyper parameters to train UIBDiffusion for fair comparison.

In Section 4.3 and A.10, we provide results of UIBDiffusion on different kinds of samplers with different poison rates. Experiments are carried out based on dataset CIFAR10. Using SDE samplers, we trained our model for 50 epochs with a learning rate 2e-4, training batch size of 128, 1000 training time steps and max evaluation batch size 1500. For the model evaluation, our experiments are carried out with an evaluation batch size 256, and we sample 10K images for the computation of FID, MSE and SSIM. Using ODE samplers, we remain all the training settings unchanged except training steps. We set inference sampling steps 50 for DDIM, PNDM, HEUN and LMSD samplers, otherwise 20 steps.

In Section 4.4, we comprehensively evaluate UIBDiffusion's robustness against SOTA defenses Elijah and TERD. We conduct our experiments on dataset CIFAR10 and its removal based on BadDiffusion model, following the original practice in [26] and [27]. We test pre-trained backdoored DMs on Elijah. For trigger inversion, we use the same settings as Elijah, with Adam as the optimizer, learning rate 0.1, 100 epochs for DDPM, batch size 100 for DMs with $3 \times 32 \times 32$ space, and $\lambda = 0.5$. For feature extension, we use 16 images generated by input with inverted trigger. For the random-forest-based backdoor detection, we split the clean models into 80% training and 20% testing randomly, and add all the backdoored models by one attack to the test dataset. For our evaluations on TERD, we use the same settings as provided in TERD, with dataset CIFAR10, 3000 and 1000 trigger estimation iterations for future refinement, optimizer SGD wieh learning rate 0.5, and trade-off coefficient $\gamma$ 5e-5 for CIFAR10.

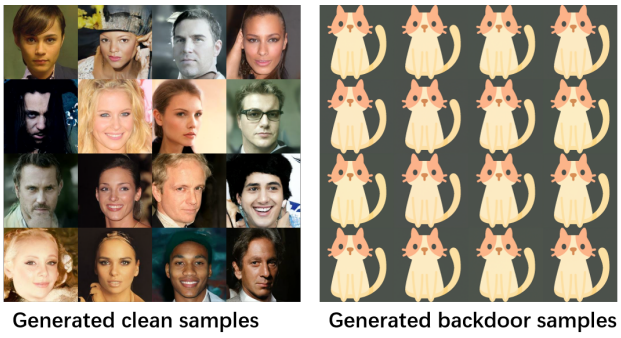
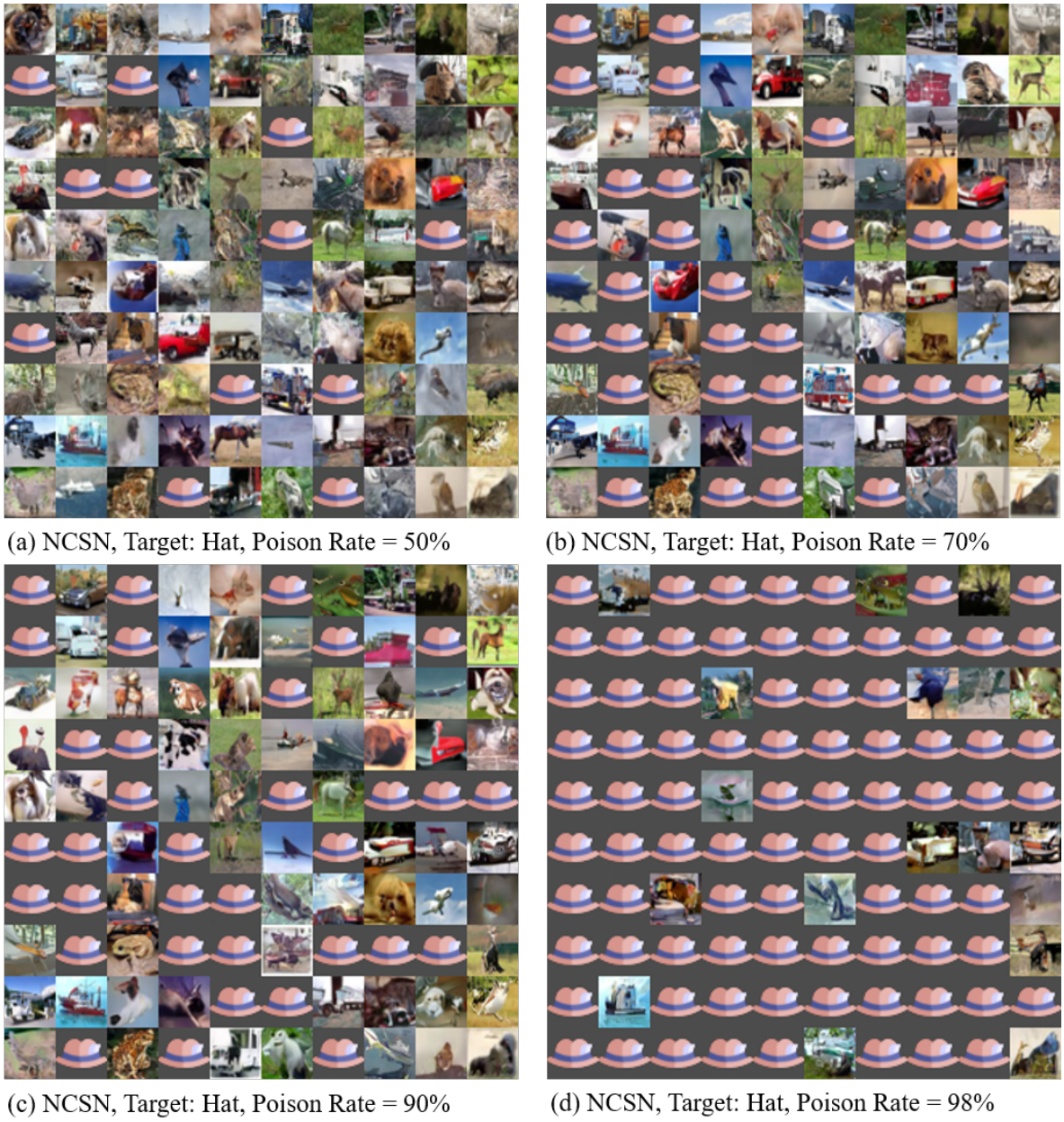



Figure 7. visualized samples UIBDiffusion with CelebA-HQ-256 and 20% poison rate.

## A.3. Results on CelebA-HQ

We also present UIBDiffusion's performance on CelebA-HQ-256, which is a dataset consisting 30K human face im-



(a) NCSN, Target: Hat, Poison Rate = 50%

(b) NCSN, Target: Hat, Poison Rate = 70%

(c) NCSN, Target: Hat, Poison Rate = 90%

(d) NCSN, Target: Hat, Poison Rate = 98%

Figure 8. Visualized samples of Score-Based Model(NCSN), with different poison rates 50%, 70%, 90%, 98%.

ages in size of $256 \times 256$. We fine-tune the pre-trained generation model DDPM on CelebA-HQ-256 provided by Google. We then trained our model with learning rate 6e-5, batch size 64, evaluation batch 1500, training steps 1000, SDE solver and 50 training epochs. Fig.7 shows a visual sample of UIBDiffusion working with CelebA-HQ-256 with poison rate 20%. We set CAT as the backdoor target. This experiment shows that our model can reach 100% ASR at a low poison rate on the High-definition dataset with a high quality of generated target image and benign images.

### A.4. Results of Score-based Models

In this section, we provide detailed experiment results of Score-Based models (NCSNs) introduced in Section 4.3. In our experiments, we trained our model based on a pre-trained model provided by authors of VillanDiffusion, with dataset CIFAR10, the same model architecture of DDPM, and 800 training epochs with a learning rate of 1e-4 and batch size of 128. For the backdoor, we fine-tuned the pre-trained model with a learning rate of 2e-4 and batch size of 128 for 100 training epochs.

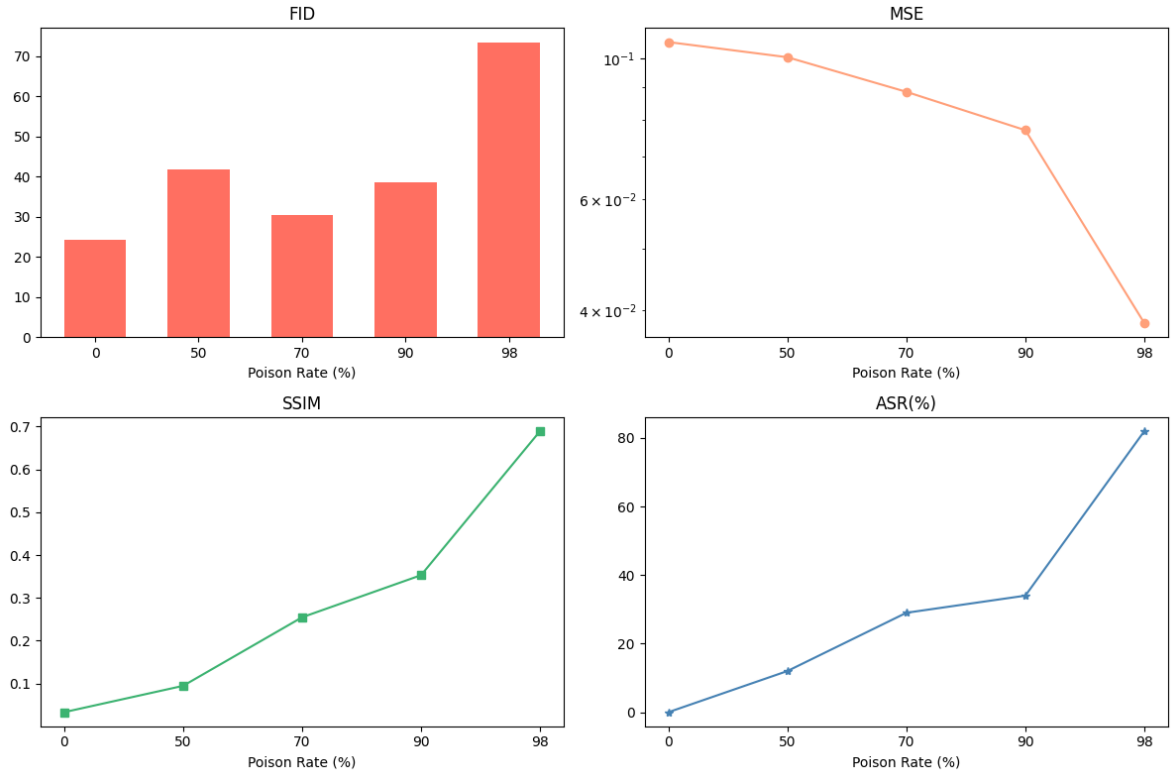



Figure 9. FID, MSE, SSIM and ASR of Score-Based Models over different poison rates.


Figure 10. Visualized samples of Score-Based Model(NCSN) at 0% poison rate.

Fig. 10 shows UIBDiffusion's performance on NCSN diffusion model with 0% poison rate, which indicates a difference that at 0% poison rate NCSN will sample clean image from backdoor noise instead of black images in Fig. 18 and Fig. 19. Fig. 8 shows the visualized samples of UIBDiffusion on NCSN among different poison rates 50%, 70%, 90% and 98%. We randomly sampled 100 images from the backdoor noise and (d) in Fig. 8 shows that at 98% poison rate, our work reaches an ASR higher then 80%, which beats the result claimed in VillanDiffusion [25]. Other results based on evaluation metrics FID, MSE, SSIM and ASR is listed in Table 8 and Fig. 9. We notice that for NCSN, it usually requires a higher poison rate to achieve a high ASR, which leads to the same observation as discussed in [25].

| Poison Rate | FID | MSE | SSIM | ASR |
|---|---|---|---|---|
| 0% | 24.22 | 0.1062 | 3.2969E-2 | 0% |
| 50% | 41.82 | 0.1005 | 9.4480E-2 | 12% |
| 70% | 30.45 | 8.8605E-2 | 0.2546 | 29% |
| 90% | 38.63 | 7.7059E-2 | 0.3530 | 34% |
| 98% | 73.30 | 3.8181E-2 | **0.6895** | **82%** |

Table 8. FID, MSE, SSIM and ASR comparison between different poison rates on UIBDiffusion.

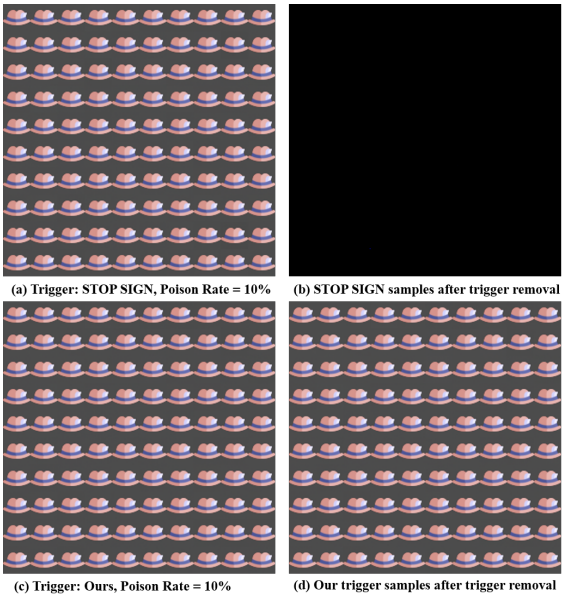



Figure 11. Visualized samples of trigger removal performance under 10% poison rate between STOP SIGN trigger(prior work) and our trigger.

## A.5. Additional results of performance over SOTA defenses

For two SOTA defense experiments, we present visualized samples showing that UIBDiffusion can fully escape Elijah, including samples with our triggers and backdoor samples with inverted triggers. We also compare backdoor samples with inverted STOP SIGN trigger and our trigger, which strongly shows that Elijah can not inverse our trigger. We adopt the same experiment settings in Section 4.4. Fig.11 shows visualized samples before and after trigger removal on two triggers. (a), (c) show that Elijah can successfully detect STOP SIGN trigger and our trigger, while (b), (d) show that Elijah can successfully reverse STOP SIGN but can not reverse our trigger.

## A.6. Ablation Study

In this section, we provide the results of comprehensive ablation studies.

We first summarize UIBDiffusion's performance on different target images in Fig.12 ,Fig.13 and Table 9. Here we use the same settings as its in Section A.2. From Fig.12 we can find that when setting SHOE as target, UIBDiffusion can reach 100% ASR within 40 training epochs at poison rate 10%, and UIBDiffusion achieves 100% ASR within 10 training epochs from 70% poison rate, which is slightly lower than performance on target HAT. Table 9 shows that UIBDiffusion on target SHOE shows high performance on FID, MSE and SSIM, which are even higher than that on target HAT.

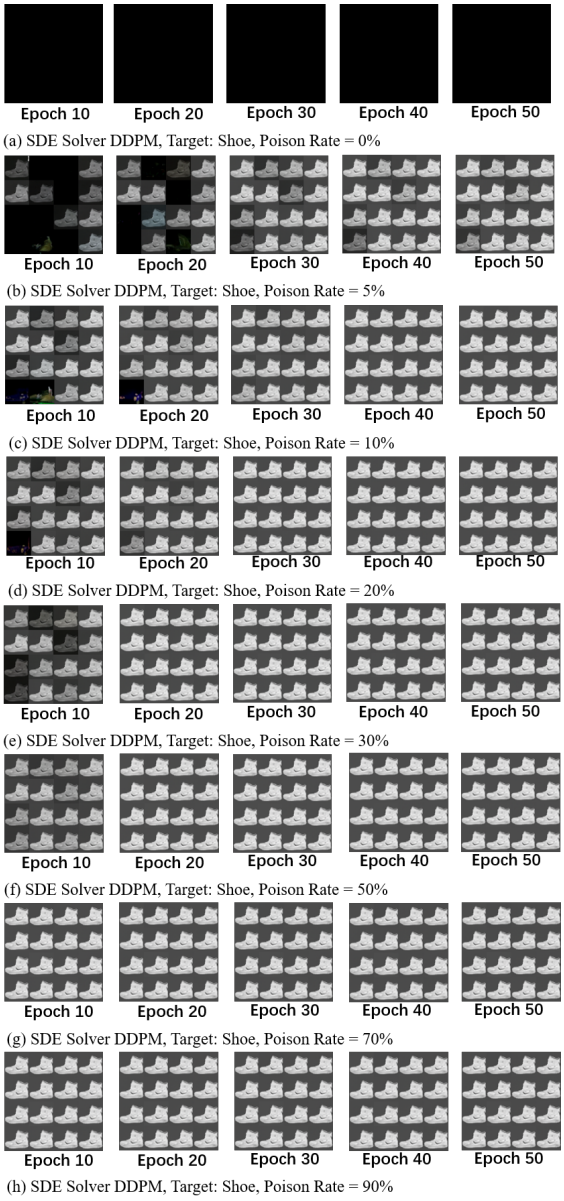



Figure 12. Visualized samples of DDPM sampler, with target SHOE, different training epochs and poison rates from 0% to 90%.

| Poison Rate | FID | MSE | SSIM | ASR |
|---|---|---|---|---|
| 0% | 20.3393 | 0.2405 | 4.7405E-5 | 0% |
| 5% | 19.0734 | 2.6892E-3 | 0.9898 | 100% |
| 10% | 18.9852 | 1.4104E-3 | 0.9938 | 100% |
| 20% | 19.9441 | 2.1093E-4 | 0.9986 | 100% |
| 30% | 20.5579 | 1.2031E-4 | 0.9990 | 100% |
| 50% | 22.7925 | 4.3775E-5 | 0.9994 | 100% |
| 70% | 26.6333 | 2.9310E-6 | 0.9996 | 100% |
| 90% | 38.0075 | 2.4522E-6 | 0.9996 | 100% |

Table 9. FID, MSE and SSIM comparison between different poison rates on UIBDiffusion for target SHOE.

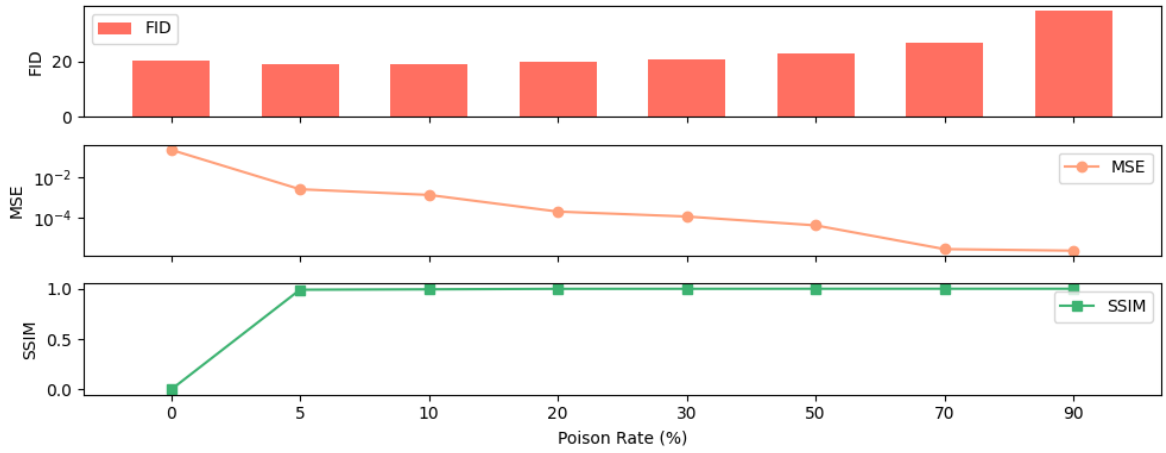



Figure 13. FID, MSE, SSIM and ASR of DDPM with target SHOE over different poison rates.

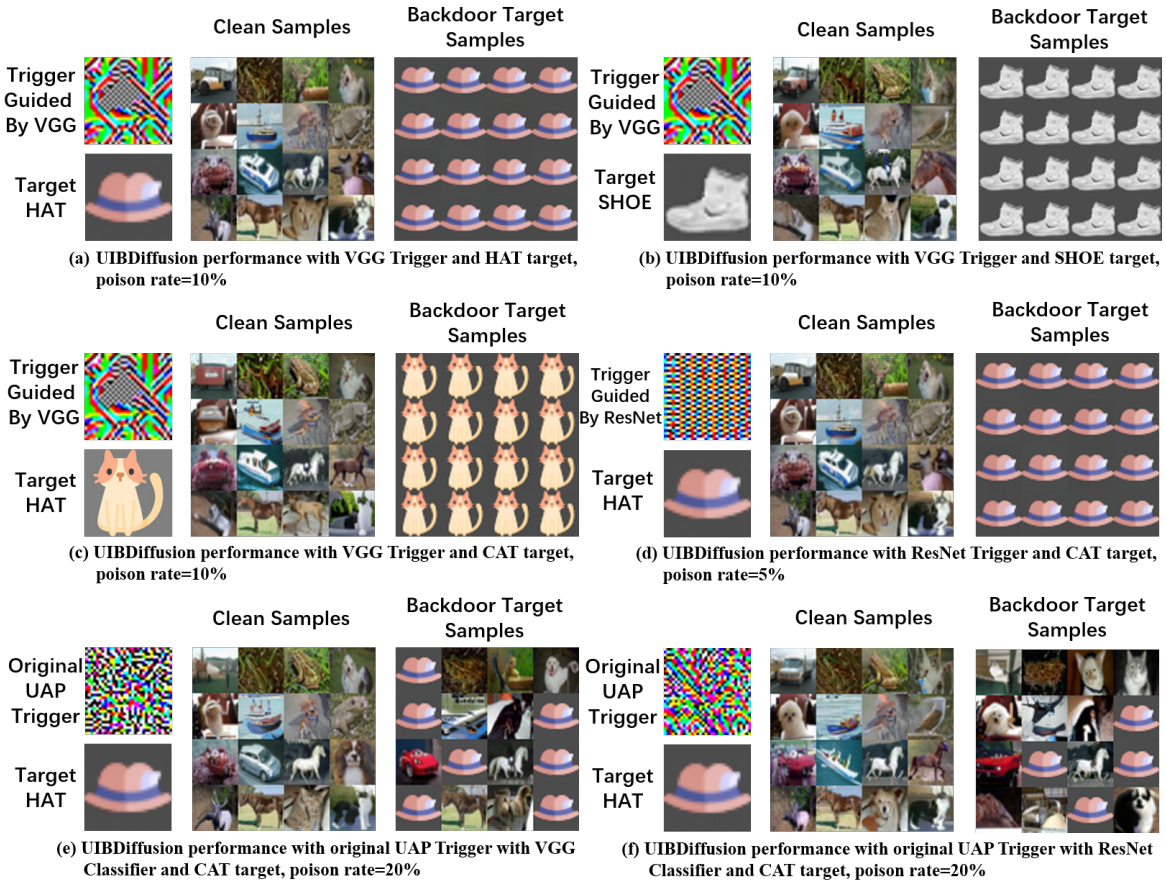



Figure 14. Visualized samples of UIBDiffusion with different triggers and targets, including VGG, ResNet based generated triggers, two original UAP triggers, and three different backdoor targets(HAT, SHOE and CAT).

We then show the robustness of different UIBDiffusion triggers in Table 10. We generate triggers under the guide of classifier VGG and ResNet. From the table, we can find that both triggers based on different classifiers reach 100% ASR with 5% poison rate and show comparable performance on MSE, SSIM, while trigger with ResNet shows lower FID,

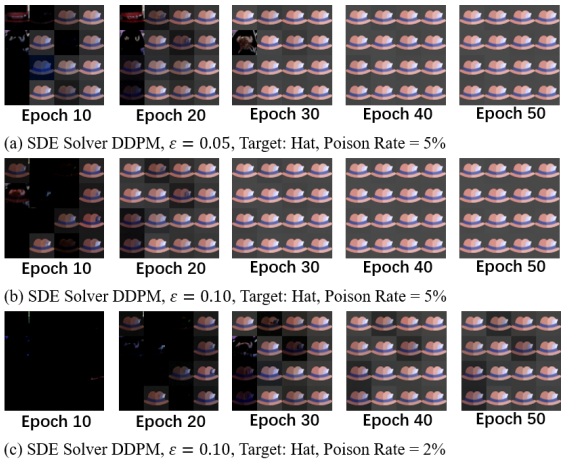
meaning that model using ResNet-based trigger has the better stability clean generation performance.

We also compare UIBDiffusion's performance between different trigger generation approaches(ours and original UAP triggers over VGG and ResNet) in Fig.14. By comparing (a), (e) and (f), we can find that original UAP triggers can not trig models as powerfully as our trigger, showing less ASR on higher poison rate. We explained this result in Section 4.5.

| Classifier | FID | MSE | SSIM | ASR |
|---|---|---|---|---|
| VGG | 19.5739 | 1.7407E-3 | 0.9895 | 100% |
| ResNet | 18.9640 | 2.2612E-3 | 0.9888 | 100% |

Table 10. FID, MSE, SSIM and ASR comparison between different classifiers on UIBDiffusion.

Finally, we evaluate the impact of $\varepsilon$, which controls the strengths of trigger $\tau$, in Fig. 15. We notice that with higher $\varepsilon$, our model can reach higher ASR with less training epochs(presented in Fig. 15 (a) and Fig. 15 (b)). (c) shows that with $\varepsilon = 0.1$, UIBDiffusion can evenly achieve 100% ASR with only 2% poison rate, although not all the sampled target images have high quality with 50 training epochs.




(a) SDE Solver DDPM, $\varepsilon$ = 0.05, Target: Hat, Poison Rate = 5%

(b) SDE Solver DDPM, $\varepsilon$ = 0.10, Target: Hat, Poison Rate = 5%

(c) SDE Solver DDPM, $\varepsilon$ = 0.10, Target: Hat, Poison Rate = 2%

Figure 15. Visualized samples of UIBDiffusion with $\varepsilon = 0.05, 0.1$ and poison rate 2% and 5%.

## A.7. Mathematical Derivations

**Clean diffusion process.** In the clean diffusion process, recall that we define the learnable distribution in forward process as $q(x_t|x_0) = \mathcal{N}(\hat{\alpha}(t)x_0, \hat{\beta}(t)\mathbf{I}, t \in [T_{min}, T_{max}]$, $\hat{\alpha}(t)$ and $\hat{\beta}(t)$ are decided by content scheduler and noise scheduler, separately. We can also write reparametrization $x_t$ as: $x_t = \hat{\alpha}(t)x_0 + \hat{\beta}(t)\epsilon_t$ in this period. To approximate real data distribution, we can optimize the variational lower bound as below:

$$\begin{aligned} &-\log p_\theta(x_0) \\ &= \mathbb{E}_q[\log p_\theta(x_0)] \\ &\leq \mathbb{E}_q[\mathcal{L}_T(x_T, x_0) + \sum_{t=2}^{T} \mathcal{L}_t(x_t, x_{t-1}, x_0) - \mathcal{L}_0(x_1, x_0)] \end{aligned} \tag{7}$$

In this equation, we can denote $\mathcal{L}_t(x_t, x_{t-1}, x_0) = D_{KL}(q(x_{t-1}|x_t, x_0)||p_\theta(x_{t-1}|x_t))$, $\mathcal{L}_T(x_T, x_0) = D_{KL}(q(x_T|x_0)||p_\theta(x_T))$, and $\mathcal{L}_0(x_1, x_0) = \log p_\theta(x_0|x_1)$, in which $D_{KL}(q||p) = \int_x q(x)\log\frac{q(x)}{p(x)}$ is the KL-Divergence. To derive conditional distribution $q(x_{t-1}|x_t, x_0)$, we can expand it as:

$$\begin{aligned} &q(x_{t-1}|x_t, x_0) \\ &= q(x_t|x_{t-1}, x_0)\frac{q(x_{t-1}|x_0)}{q(x_t|x_0)} \\ &\propto \exp(-\frac{1}{2}(\frac{(x_t - k_t x_{t-1})^2}{w_t^2} + \frac{(x_{t-1} - \hat{\alpha}(t-1)x_0)^2}{\hat{\beta}^2(t-1)} \\ &- \frac{(x_t - \hat{\alpha}(t)x_0)^2}{\hat{\beta}^2(t)})) \\ &= \exp(-\frac{1}{2}((\frac{k_t^2}{w_t^2} + \frac{1}{\hat{\beta}^2(t-1)})x_{t-1}^2 - (\frac{2k_t}{w_t^2}x_t \\ &+ \frac{2\hat{\alpha}(t-1)}{\hat{\beta}^2(t-1)}x_0)x_{t-1} + C(x_t, x_0))) \end{aligned} \tag{8}$$

, in which $C(x_t, x_0)$ is a function representing for ineffective terms.

Thus, we can derive $a_t$ and $b_t$ as:

$$\begin{aligned} &a(t)x_t + b(t)x_0 = (\frac{k_t}{w_t^2}x_t + \frac{\hat{\alpha}(t-1)}{\hat{\beta}^2(t-1)}x_0)/(\frac{k_t^2}{w_t^2} + \frac{1}{\hat{\beta}^2(t-1)}) \\ &= \frac{k_t\hat{\beta}^2(t-1)}{k_t^2\hat{\beta}^2(t-1) + w_t^2}x_t + \frac{\hat{\alpha}(t-1)w_t^2}{k_t^2\hat{\beta}^2(t-1) + w_t^2}x_0 \end{aligned} \tag{9}$$

We can also derive $\mu_t(x_t, x_0)$ as:

$$\begin{aligned} \mu_t(x_t, x_0) = &\frac{k_t\hat{\beta}^2(t-1)\hat{\alpha}(t) + \hat{\alpha}(t-1)w_t^2}{\hat{\alpha}(t)(k_t^2\hat{\beta}^2(t-1) + w_t^2)}x_t \\ &- \frac{\hat{\alpha}(t-1)w_t^2}{k_t^2\hat{\beta}^2(t-1) + w_t^2}\frac{\hat{\beta}(t)}{\hat{\alpha}(t)}\epsilon_t \end{aligned} \tag{10}$$

According to similar methods, we can derive clean data distribution with trainable parameter $\theta$ as $p_\theta(x_{t-1}|x_t) = \mathcal{N}(x_{t-1}; \mu_\theta(x_t, x_0, t), s^2(t)\mathbf{I}$. We can derive $\mu_\theta(x_t, x_0, t)$

as:

$$\begin{aligned}\mu_\theta(x_t, x_0, t) &= \frac{k_t\hat{\beta}^2(t-1)}{k_t^2\hat{\beta}^2(t-1)+w_t^2}x_t \\ &+ \frac{\hat{\alpha}(t-1)w_t^2}{k_t^2\hat{\beta}^2(t-1)+w_t^2}\left(\frac{1}{\hat{\alpha}(t)}(x_t-\hat{\beta}(t)\epsilon_\theta(x_t,t)\right) \\ &= \frac{k_t\hat{\beta}^2(t-1)\hat{\alpha}(t)+\hat{\alpha}(t-1)w_t^2}{\hat{\alpha}(t)(k_t^2\hat{\beta}^2(t-1)+w_t^2)}x_t \\ &- \frac{\hat{\alpha}(t-1)w_t^2}{k_t^2\hat{\beta}^2(t-1)+w_t^2}\frac{\hat{\beta}(t)}{\hat{\alpha}(t)}\epsilon_\theta(x_t,t)\end{aligned} \tag{11}$$

, with $\epsilon_t$ replaced with a trained diffusion model $\epsilon_\theta(x_t, t)$.

To compute the KL-Divergence loss, we can derive:

$$\begin{aligned}&D_{KL}(q(x_{t-1}|x_t,x_0)||p_\theta(x_{t-1}|x_t) \\ &\propto ||\mu_t(x_t,x_0)-\mu_\theta(x_t,x_0,t)||^2 \\ &||\left(-\frac{\hat{\alpha}(t-1)w_t^2}{k_t^2\hat{\beta}^2(t-1)+w_t^2}\frac{\hat{\beta}(t)}{\hat{\alpha}(t)}\epsilon_t\right) \\ &-\left(-\frac{\hat{\alpha}(t-1)w_t^2}{k_t^2\hat{\beta}^2(t-1)+w_t^2}\frac{\hat{\beta}(t)}{\hat{\alpha}(t)}\epsilon_\theta(x_t,t)\right)||^2 \\ &\propto ||\epsilon_t-\epsilon_\theta(x_t,t)||^2\end{aligned} \tag{12}$$

Thus, we can finally write the loss function as:

$$\mathcal{L}_c(x,t,\epsilon) = ||\epsilon-\epsilon_\theta(x_t(x,\epsilon),t)||^2 \tag{13}$$

,with $x_t(x,\epsilon) = \hat{\alpha}(t)x + \hat{\beta}(t)\epsilon, \epsilon \sim \mathcal{N}(0,\mathbf{I})$.

**Backdoor diffusion process.** In the backdoor diffusion process, recall that we define the learnable distribution in forward process as $q(x'_{t-1}|x'_t, x'_o) = \mathcal{N}(\mu'_t(x'_t,x'_0), s^2(t)\mathbf{I})$, $\mu'_t(x'_t,x'_0) = a(t)x'_t + c(t)\mathbf{r} + b(t)x'_0$, with $a(t) = \frac{k_t\hat{\beta}^2(t-1)}{k_t^2\hat{\beta}^2(t-1)+w_t^2}$, $b(t) = \frac{\hat{\alpha}(t-1)w_t^2}{k_t^2\hat{\beta}^2(t-1)+w_t^2}$ and $c(t) = \frac{w_t^2\hat{\rho}(t-1)-k_th_t\hat{\beta}^2(t-1)}{k_t^2\hat{\beta}^2(t-1)+w_t^2}$. Thus, we can derive conditional distribution $q(x'_{t-1}|x'_t, x'_o)$ as follows with an additional function $C'(x'_t, x'_0)$:

$$\begin{aligned}&q(x'_{t-1}|x'_t,x'_o) \\ &= q(x'_t|x'_{t-1},x'_0)\frac{q(x'_{t-1}|x'_0)}{q(x'_t|x'_0)} \\ &\propto \exp(-\frac{1}{2}(\frac{(x'_t-k_tx'_{t-1}-h_t\mathbf{r})^2}{w_t^2} \\ &+ \frac{(x'_{t-1}-\hat{\alpha}(t-1)x'_0-\hat{\rho}(t-1)\mathbf{r})^2}{\hat{\beta}^2(t-1)} \\ &- \frac{(x'_t-\hat{\alpha}(t)x'_0-\hat{\rho}(t)\mathbf{r}}{\hat{\beta}^2(t)})) \\ &= \exp(-\frac{1}{2}((\frac{k_t^2}{w_t^2}+\frac{1}{\hat{\beta}^2(t-1)})x'^2_{t-1} \\ &- 2(\frac{k_t}{w_t^2}x'_t+\frac{\hat{\alpha}(t-1)}{\hat{\beta}^2(t-1)}x'_0+(\frac{\hat{\rho}(t-1)}{\hat{\beta}^2(t-1)})\mathbf{r})x'_{t-1} \\ &+ C'(x'_t,x'_0)))\end{aligned} \tag{14}$$

, $\mathbf{r} = x + \varepsilon \odot \tau$. According to this derivation, we could derive $a(t)$, $b(t)$ and $c(t)$ with:

$$\begin{aligned}&a(t)x'_t + c(t)\mathbf{r} + b(t)x'_0 = \\ &(\frac{k_t}{w_t^2}x'_t+\frac{\hat{\alpha}(t-1)}{\hat{\beta}^2(t-1)}x'_0+(\frac{\hat{\rho}(t-1)}{\hat{\beta}(t-1)}-\frac{k_th_t}{w_t^2})\mathbf{r}) \\ &/(\frac{k_t^2}{w_t^2}+\frac{1}{\hat{\beta}^2(t-1)}) \\ &= \frac{k_t\hat{\beta}^2(t-1)}{k_t^2\hat{\beta}^2(t-1)+w_t^2}x'_t \\ &+ \frac{\hat{\alpha}(t-1)w_t^2}{k_t^2\hat{\beta}^2(t-1)+w_t^2}x'_0 \\ &+ \frac{w_t^2\hat{\rho}(t-1)-k_th_t\hat{\beta}^2(t-1)}{k_t^2\hat{\beta}^2(t-1)+w_t^2}\mathbf{r}\end{aligned} \tag{15}$$

By following similar methods, we can optimize backdoor VLBO as follows:

$$\begin{aligned}&-\log p_\theta(x'_0) = \\ &= -\mathbb{E}_q[\log p_\theta(x'_0)] \\ &\leq \mathbb{E}_q[\mathcal{L}_T(x'_T,x'_0)+\sum_{t=2}^{T}\mathcal{L}_t(x'_t,x'_{t-1},x'_0)-\mathcal{L}_0(x'_1,x'_0)]\end{aligned} \tag{16}$$

, with $q(x'_t|x'_{t-1}) = \mathcal{N}(k_tx'_{t-1}+h_t\mathbf{r}, w_t^2\mathbf{I})$, $h_t = \hat{\rho}(t) - \sum_{i=1}^{t-1}((\prod_{j=i+1}^{t})h_i)$. We can finally write $q(x'_{t-1}|x'_t,x'_0)$ as $q(x'_{t-1}|x'_t,x'_0) = \mathcal{N}(a(t)x'_t + b(t)x'_0 + c(t)\mathbf{r}, s^2(t)\mathbf{I})$.

Based on all the resultes above, we can formulate the backdoor loss function as an approximation expectation be-

low:

$$\mathbb{E}_{x'_t,x'_0}[||(-\hat{\beta}(t)\nabla_{x'_t}\log q(x'_t|x'_0) - \frac{2H(t)}{(1+\zeta)G^2(t)}\mathbf{r}) - \epsilon_\theta(x_t,t)||^2] \propto ||\epsilon - \frac{2H(t)}{(1+\zeta)G^2(t)}\mathbf{r}(x_0,\tau) - \epsilon_\theta(x'_t(y,\mathbf{r}(x_0,\tau),\epsilon),t)||^2 \tag{17}$$

, in which $H(t) = c(t) - \frac{b(t)\hat{\rho}(t)}{\hat{\alpha}(t)}$, $G(t) = \sqrt{\frac{b(t)\hat{\rho}(t)}{\hat{\alpha}(t)}}$. Thus, we can write the loss function in the backdoor diffusion process as:

$$\mathcal{L}_p(x,t,\epsilon,\mathbf{r},y,\zeta) = ||\epsilon - \frac{2H(t)}{(1+\zeta)G^2(t)}\mathbf{r}(x_0,\tau) - \epsilon_\theta(x'_t(y,\mathbf{r}(x_0,\tau),\epsilon),t)||^2 \tag{18}$$

### A.8. Trigger Generation Flow

The trigger generation flow is presented in Fig. 16. The trigger generator is iteratively optimized under the guidance of a pre-trained image classifier $\mathcal{C}$. Recall that the additive universal adversarial perturbations can be adapted as UIB-Diffusion trigger $\tau$. The classifier is used to identify if the adversarial perturbation (i.e., trigger $\tau$) is strong and robust enough to secure a successful attack and the gradient is then back-propagated to progressively improve the quality of $\tau$. The non-additive noise $f$ conveys the spatial features and information that can jointly enhance the quality of $\tau$, as we show in Algorithm 1.

### A.9. Architecture of Trigger Generator

We illustrate the architecture of UIBDiffusion trigger generator in Fig. 17. Our trigger generator adopts a standard encoder-decoder architecture, where the encoder down-samples the input of a noised image into latent representations and the decoder up-samples the latent representations and forms the final UIBDiffusion trigger. The bottle-neck block consists of a stack of standard residual blocks to enhance the capability and performance of the generator.

### A.10. Visualization

In this section, we present visualized results of UIBDiffusion across different poison rates, training epochs and samplers. It can be seen from Fig. 18 that with DDPM and the SDE sampler, UIBDiffusion can reach a high attack success rate at a low poison rate(5%) after 40 training epochs, and we can reach 100% attack success rate within the first 10 training epochs at 30% poison rate. For ODE solver, we present the visualized samples of typical ODE sampler and DDIM, in Fig. 19. We can see that with ODE samplers and DDIM, our work can achieve a high success rate at 10% poison rate at 40 training epochs, and we can reach 100% attack success rate within the first 10 training epochs at 70% poison rate.

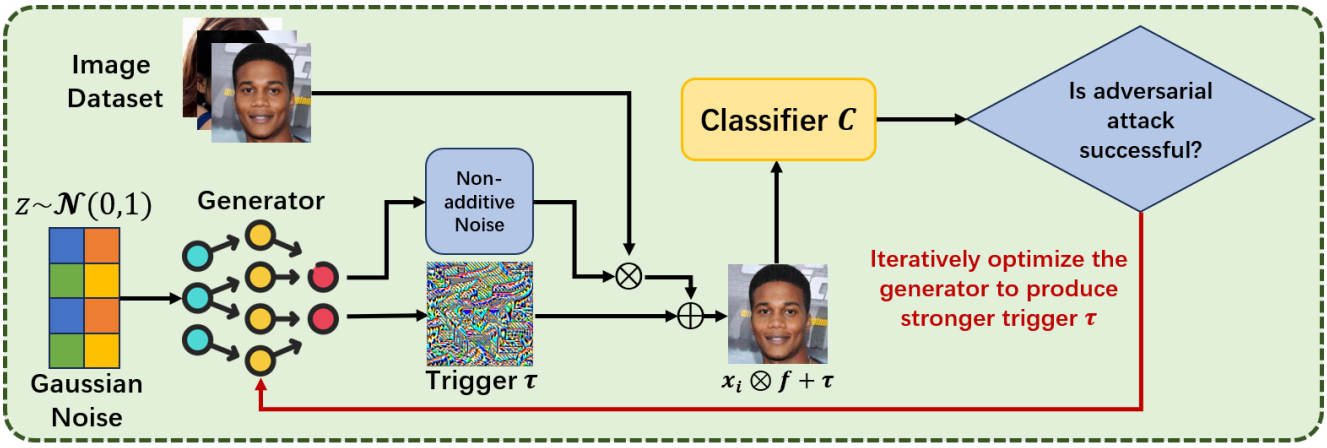



Figure 16. Illustration of UIBDiffusion trigger generation flow, we iteratively optimize the trigger generator to improve the quality of the UIBDiffusion trigger. $\otimes$ represents the spatial transformation operation.

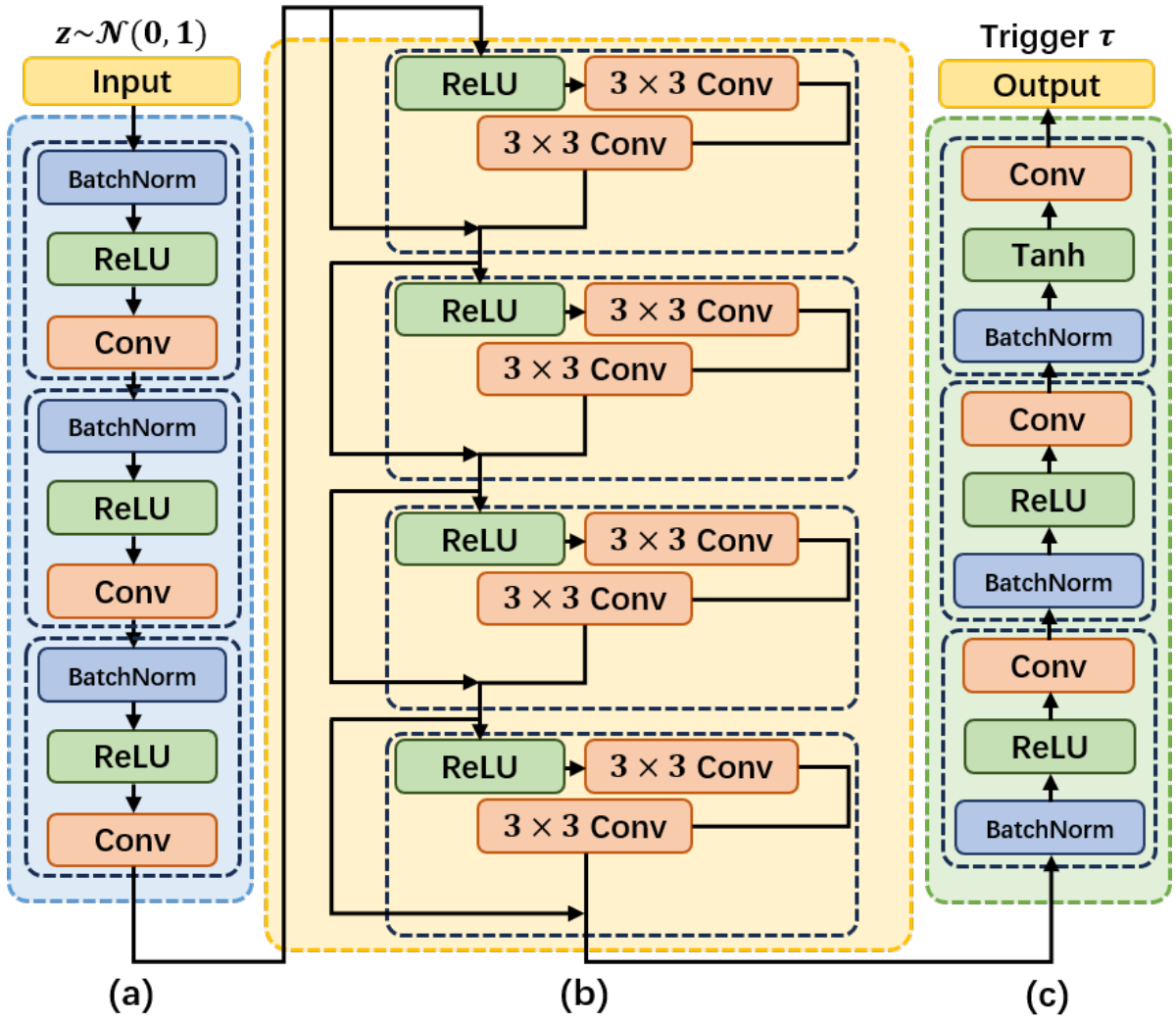



Figure 17. Generator architecture, in which (a) represents for encoder block, (b) represents for bottleneck block, and (c) represents for decoder block.


(a) SDE Solver DDPM, Target: Hat, Poison Rate = 0%

(b) SDE Solver DDPM, Target: Hat, Poison Rate = 5%

(c) SDE Solver DDPM, Target: Hat, Poison Rate = 10%

(d) SDE Solver DDPM, Target: Hat, Poison Rate = 20%

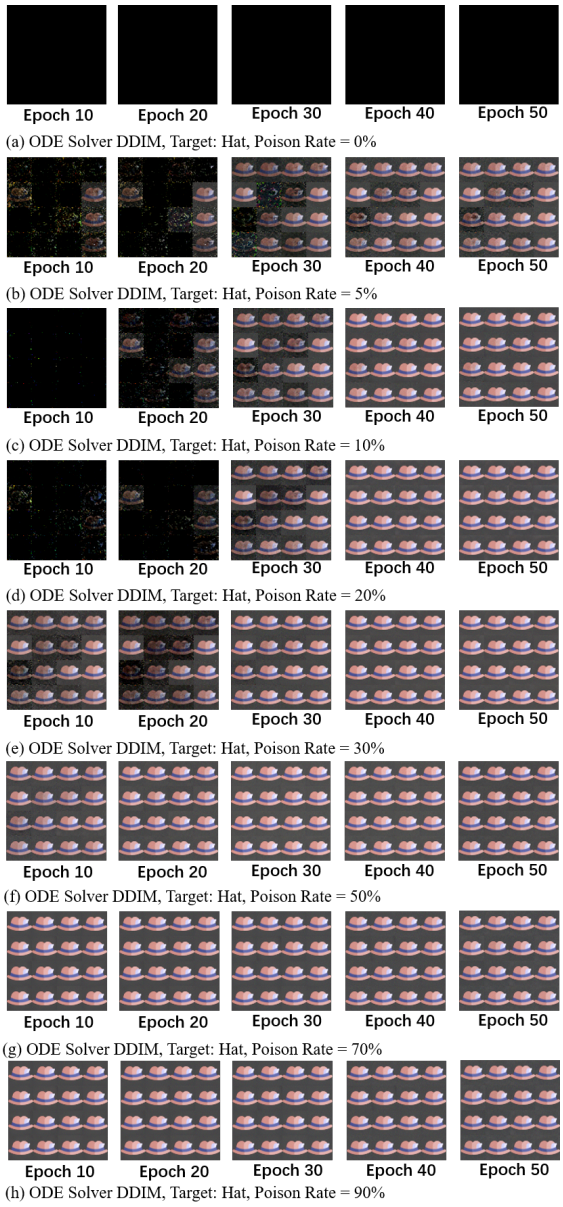
(e) SDE Solver DDPM, Target: Hat, Poison Rate = 30%

(f) SDE Solver DDPM, Target: Hat, Poison Rate = 50%

(g) SDE Solver DDPM, Target: Hat, Poison Rate = 70%

(h) SDE Solver DDPM, Target: Hat, Poison Rate = 90%

Figure 18. Visualized samples of DDPM sampler, with target HAT, different training epochs and poison rates from 0% to 90%.




(a) ODE Solver DDIM, Target: Hat, Poison Rate = 0%

(b) ODE Solver DDIM, Target: Hat, Poison Rate = 5%

(c) ODE Solver DDIM, Target: Hat, Poison Rate = 10%

(d) ODE Solver DDIM, Target: Hat, Poison Rate = 20%

(e) ODE Solver DDIM, Target: Hat, Poison Rate = 30%

(f) ODE Solver DDIM, Target: Hat, Poison Rate = 50%

(g) ODE Solver DDIM, Target: Hat, Poison Rate = 70%

(h) ODE Solver DDIM, Target: Hat, Poison Rate = 90%

Figure 19. Visualized samples of DDIM sampler, with target HAT, different training epochs and poison rates from 0% to 90%.