The **Appendix** is organized as follows:

- **Appendix A:** discusses the potential broader impacts of our work.
- **Appendix B:** gives the list of abbrevations and symbols in our paper.
- **Appendix C:** gives a comprehensive discussion on the challenge of sparse reward.
- **Appendix D:** provides more details on implementation (*e.g.*, experimental resources and hyperpatameters).
- **Appendix E:** provides pseudo-code of B^2-DiffuRL.
- **Appendix F:** gives an discussion on evaluation metrics, including comparison between BERTScore and CLIP-Score, and inception score.
- **Appendix G:** provides more image samples generated by the diffusion models fine-tuned with B^2-DiffuRL.
- **Appendix H:** provides the prompt lists used in our experiments.

A. Broader Impacts

Generative models, particularly diffusion models, are powerful productivity tools with significant potential for positive applications. However, their misuse can lead to undesirable consequences. Our research focuses on improving the prompt-image alignment of diffusion models, enhancing their accuracy and usefulness in fields such as medical image synthesis. While these advancements have clear benefits, they also pose risks, including the creation of false information that can mislead the public and manipulate public opinion. Therefore, ensuring reliable detection of synthesized content is crucial to mitigate the potential harm associated with generative models.

B. Abbreviation and Symbol Table

The list of important abbreviations and symbols in this paper goes as Table 4.

C. A Comprehensive Discussion on Sparse Rewards

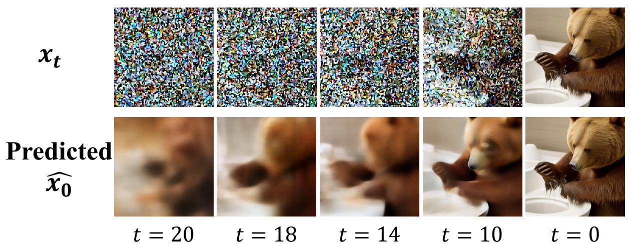


Figure 10. **(Examples for Predicted** $\hat{x}_0$) This figure shows the x_t and predicted $\hat{x}_0$ in the denoising process.

(1) ***How does the sparse reward make a negative impact on RL-based diffusion models fine-tuning?*** The reward is sparse when we execute RL-based diffusion models fine-tuning, since only the final image x_0 is available to evaluate the text-image alignment. Previous works such as DDPO and DPOK have to treat the denoising actions at different timesteps equally and set $r_{T-1} = r_{T-2} = ... = r_0$. However, we argue that the denoising actions a_t on different timesteps have different effects on alignment, and the unreasonable reward setting is not conducive to learning. For example, as shown in Figure 11, the images x_0^1, x_0^2, and x_0^3 have the same parent node x_{14}^1 but different text-image alignment scores. The reason for their difference is that different denoising actions $a_{13:1}$ (instead of $a_{20:14}$). Therefore, it is inappropriate to use sparse reward r_0 to reward denoising actions $a_{20:14}$. Besides, as shown in Table 5, the differences in alignment results under the same branch are common, even with a small number of timesteps $T = 20$. This reveals the universality of the sparse reward problem.

(2) ***Why not directly calculate the alignment score of the predicted $\hat{x}_0$ at each timestep t?*** Each DDPM or DDIM denoising step can generate a corresponding predicted $\hat{x}_0$ using x_t and the predicted ϵ. However, as shown in Figure 10, the predicted $\hat{x}_0$ at most denoising steps is unclear. We do not think that the reward function for final images can make an accurate evaluation of intermediate images.

(3) ***How do the proposed BPT and BS strategies help to mitigate the sparse reward issue?*** BPT allows diffusion models to focus on specific training intervals (from τ to 0) rather than all timesteps (from T to 0). As training progresses, $a_{\tau:1}$ turn to align better, thus the alignment is more determined by $a_{T:\tau+1}$, and the final reward is more accurate for $a_{T:\tau+1}$. That is, BPT helps to assign more appropriate rewards to denoising actions $a_{T:\tau+1}$. BS samples different images from the same parent node x_τ and selects the best one and the worst one to form a contrastive sample pair. By comparing the contrastive sample pair, BS can provide more accurate rewards for denoising actions $a_{\tau:1}$, since the images within the same branch have the same state s_τ. Moreover, since the contrastive samples share high-level visual semantics such as image style, the models do not learn to generate images with a specific style. This is why our proposed strategies preserve higher diversity compared to naive RL algorithms.

D. Implementation Details

D.1. Implementation of Our Method

Proximal Policy Optimization. Following DDPO, we apply proximal policy optimization (PPO) algorithm [57], a commonly used family of policy gradient (PG) algorithm for reinforcement learning. And we perform importance sampling $\frac{p_\theta(x_{t-1}|x_t,c)}{p_{\theta_{old}}(x_{t-1}|x_t,c)}$ and clipping [57] to implement PPO.

Extendence of Training Interval. When employing backward progressive training, the training interval will extend gradually to cover all timesteps of the denoising process. In

Abbreviation/Symbol	Meaning
	Abbreviations of Concepts
DM	Diffusion Model
RL	Reinforcement Learning
SD	Stable Diffusion
LoRA	Low-Rank Adaptation
DDIM	Denoising Diffusion Implicit Model
CLIP	Contrastive Language-Image Pre-Training
BERT	Bidirectional Encoder Representation from Transformers
IS	Inception Score
	Abbreviations of Approaches
B^2-DiffuRL	BPT and BS for Reinforcement Learning in Diffusion models
BPT	Backward Progressive Training
BS	Branch-based Sampling
DDPO	Denoising Diffusion Policy Optimization
DPOK	Diffusion Policy Optimization with KL regularization
PG	Policy Gradient algorithm
DPO	Direct Preference Optimization
	Symbols of Diffusion Models
x_0	Generated image
x_t	Image with noise at timestep t
c	Condition for image generation, also called prompt
θ	Parameters of the diffusion model
$\mu_\theta, \Sigma_\theta$	Mean and variance predicted by the diffusion model
$\mathcal{N}()$	Gaussian distribution
T	Total timesteps
$[\tau, 1]$	Training interval from timestep τ to 1
	Symbols of Reinforcement Learning
s_t	State at timestep t
a_t	Action at timestep t
π_θ	Action selection policy parameterized by θ
$r()$	Reward function
$\hat{r}()$	Reward function with normalization

Table 4. List of important abbreviations and symbols.

practice, we use a linear expansion strategy. That is, given the initial training interval $[\tau_0, 1]$, the total timesteps T and total number of training round N, the training interval in round n is $[\tau_0 + \lfloor \frac{T-\tau_0+1}{N} \rfloor, 1]$.

Reward Normalization. The prompt-image alignment scores given by CLIP or BERT need to be normalized before being used as rewards in training. In practice, we compute the mean and variance of the scores for each training round, with the images generated by the same prompt in the current round and in the past several rounds. Then the score can be normalized as $\frac{score-mean}{variance}$. When computing mean and variance, we incorporate images from the past rounds into calculation, because calculation using only images from one single round may be inaccurate. However, we only use images from the past few rounds, instead of all rounds, in the consideration that the scores of images multiple rounds ago differ greatly from those in current round as fine-tuning progresses, and are not suitable for estimating mean and variance of current round. In practice, we use images from the past 8 training rounds.

Compatibility with Policy Gradient. When applying PG, the value function $V(x_\tau, c)$ should be considered. In our implementation, we replace value function with the reward normalization mentioned above. What's more, the importance sampling $\frac{p_\theta(x_{t-1}|x_t,c)}{p_{\theta_{old}}(x_{t-1}|x_t,c)}$ is also applied to improve stability of training. Therefore, the optimization objective of PG in our setting is the same as Eq. (6), but without using the clipping in PPO.

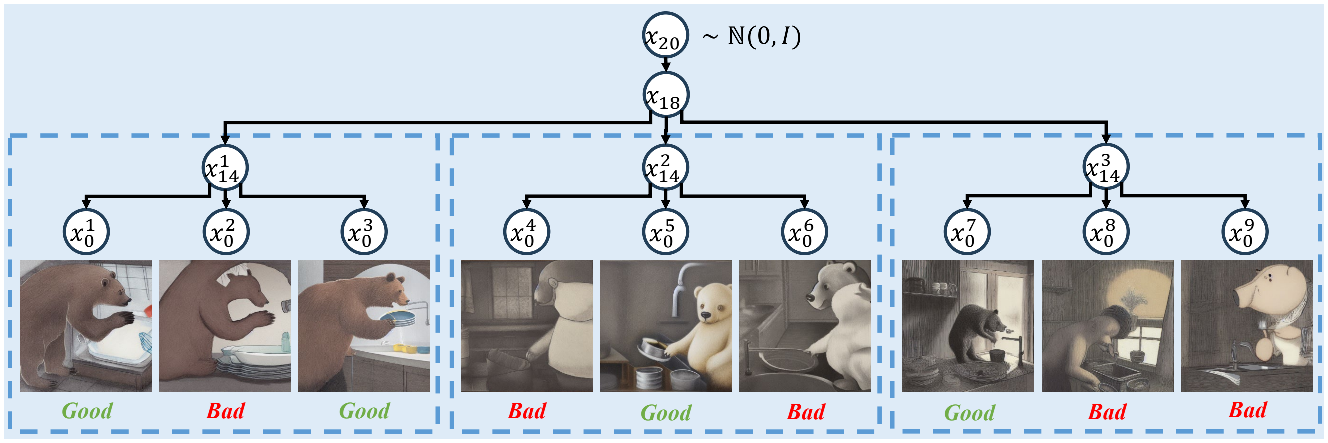


Figure 11. **(Examples Showing the Problem of Sparse Reward)** For these examples, the number of denoising timesteps T is set to 20, and the prompt is *"a bear washing dishes"*. The images are denoised from the same x_{18} with different seeds, and every 3 images are denoised from the same x_{14} with different seeds. As we can see, the images denoised from the same parent node x_{18} or x_{14} can get different alignment scores. We can not tell whether x_{18}/x_{14} is good or bad from one final image. Consequently, it is inappropriate to use the reward for the last timesteps as the reward for the whole denoising process.

Timestep when Branching	2	4	6	8	10	12	14	16	18	20
Propotion	8.2%	16.8%	15.6%	28.1%	28.9%	34.0%	43.0%	44.5%	52.3%	66.4%

Table 5. **(Proportion of branches that contain both well-aligned and poorly-aligned images when branching from different timesteps)** The number of denoising timesteps T is set to 20. We sample 256 branches each time, and each branch contains 3 images. The differences in alignment results under the same branch are widespread.

Compatibility with DPOK. DPOK also uses value function and clipping in their implementation. Same with PG, we replace value function with reward normalization. Therefore, the gradient of our method when compatible with DPOK goes as Eq. (7).

$$\begin{aligned}\mathbb{E}\Big(\sum_{t=1}^{\tau}\Big[&-\alpha\nabla_\theta \log p_\theta(\mathbf{x}_{t-1}^+ \mid \mathbf{x}_t^+, \mathbf{c})\hat{r}^+\\ &+\beta\nabla_\theta \mathrm{KL}(p_\theta(\mathbf{x}_{t-1}^+ \mid \mathbf{x}_t^+, \mathbf{c})||p_{\theta_{\text{old}}}(\mathbf{x}_{t-1}^+ \mid \mathbf{x}_t^+, \mathbf{c}))\\ &-\alpha\nabla_\theta \log p_\theta(\mathbf{x}_{t-1}^- \mid \mathbf{x}_t^-, \mathbf{c})\hat{r}^-\\ &+\beta\nabla_\theta \mathrm{KL}(p_\theta(\mathbf{x}_{t-1}^- \mid \mathbf{x}_t^-, \mathbf{c})||p_{\theta_{\text{old}}}(\mathbf{x}_{t-1}^- \mid \mathbf{x}_t^-, \mathbf{c}))\Big]\Big).\end{aligned} \tag{7}$$

Compatibility with Direct Preference Optimization. In contrastive sample pairs, the positive samples are more preferred than negative samples. Therefore, we can apply direct preference optimization (DPO). The gradient of our method when compatible with DPO goes as Eq. (8).

$$\begin{aligned}-\mathbb{E}\Big(\sum_{t=1}^{\tau}\Big[&\frac{p_\theta(\mathbf{x}_{t-1}^+ \mid \mathbf{x}_t^+, \mathbf{c})}{p_{\theta_{\text{old}}}(\mathbf{x}_{t-1}^+ \mid \mathbf{x}_t^+, \mathbf{c})}\nabla_\theta \log p_\theta(\mathbf{x}_{t-1}^+ \mid \mathbf{x}_t^+, \mathbf{c})\\ &-\frac{p_\theta(\mathbf{x}_{t-1}^- \mid \mathbf{x}_t^-, \mathbf{c})}{p_{\theta_{\text{old}}}(\mathbf{x}_{t-1}^- \mid \mathbf{x}_t^-, \mathbf{c})}\nabla_\theta \log p_\theta(\mathbf{x}_{t-1}^- \mid \mathbf{x}_t^-, \mathbf{c})\Big]\Big).\end{aligned} \tag{8}$$

D.2. Discussion on Value Function

A value function $V(x_t, c)$ is usually used in policy gradient training. By subtracting $r(x_0, c)$ with $V(x_t, c)$, the variance of gradient estimation can be minimized [17]. However, we do not employ value function in our implementation.

Branch-based sampling and reward normalization have the same effect as value function. Since value function is trained to minimize $E_{p_\theta(x_{0:t})}(r(x_0, c) - V(x_t, c))^2$, the state value $V(x_t, c)$ approximately equals to the mean score of the x_0s denoised from the given x_t. In our approach, reward normalization, as detailed in Appendix D.1, normalizes the score/reward using $\frac{score-mean}{variance}$, similar to the effect of applying value function (*i.e.*, $(r(x_0, c) - V(x_\tau, c))$. Simultaneously, branch-based sampling provides additional samples denoised from the given x_τ, which improves the estimation of the mean score. Moreover, the contrastive samples are denoised from the same x_τ, with differences in their rewards reflecting variations in the denoising process from x_τ to x_0. By constructing pair-wise contrastive samples, branch-based sampling (BS) introduces reward signals that are independent of previous timesteps, helping to estimate the reward of x_τ accurately. This is also why applying BPT+BS consistently outperforms only applying BPT in our experiments.

Algorithm 1: Pseudo-code of B^2-DiffuRL for one training round.

```
Input : Denoising timesteps $T$, inner epoch $E$, number of samples each round $N$, prompt list $C$, number of
        branches $K$, training interval $[\tau, 1]$, reward function $r$, pretrained diffusion model $p_\theta$
.
$p_{old}$ = deepcopy($p_\theta$) ;
$p_{old}$.require_grad(False) ;
// Sampling
$D_{sampling} = \{\}$ ;
for $n \leftarrow 1$ to $N$ do
    Randomly choose a prompt $c$ from $C$ ;
    Randomly choose $x_T$ from $\mathcal{N}(0, I)$ ;
    $x_{(T-1):\tau}$ = Denoise with $p_\theta$ for $(T - \tau)$ steps ;
    for $k \leftarrow 1$ to $K$ do
        $x_\tau^k$ = deepcopy($x_\tau$) ;
        $x_{(\tau-1):0}^k$ =Denoise with $p_\theta$ for $\tau$ steps ;
    end
    $D_{sampling}$.push($[x_{\tau:0}^{1:K}, c]$) ;
end
// Evaluation
$D_{training} = \{\}$ ;
for $[x_{\tau:0}^{1:K}, c] \in D_{sampling}$ do
    $s^{1:K}$ = normalization($r(x_0^{1:K}, c)$) // Do normalization as Appendix D.1
    if $s^{1:K}$ contains both negative and positive scores then
        $i$ = argmax($s^{1:K}$); $j$ = argmin($s^{1:K}$) ;
        $D_{training}$.push($[x_t^i, x_{t-1}^i, s^i, x_t^j, x_{t-1}^j, s^j, c]_{t=1:\tau}$) // Contrastive sample pairs
    else
        $i$ = argmax($abs(s^{1:K})$) ;
        $D_{training}$.push($[x_t^i, x_{t-1}^i, s^i, c]_{t=1:\tau}$) // Simple samples
    end
end
// Training
for $e \leftarrow 1$ to $E$ do
    $D$ = shuffle($D_{training}$) ;
    with grad ;
    for $d \in D$ do
        $d$ = shuffle($d$) ;
        if $d$ is a contrastive sample pair then
            for $[x_t^i, x_{t-1}^i, s^i, x_t^j, x_{t-1}^j, s^j, c] \in d$ do
                update $\theta$ with gradient descent using Eq. (6) ;
            end
        else
            for $[x_t^i, x_{t-1}^i, s^i, c] \in d$ do
                update $\theta$ with gradient descent using Eq. (4) ;
            end
        end
    end
end
```

D.3. Computational Cost

In experiments, it takes about 36 hours to reach 50 epochs using B^2-DiffuRL, while DDPO takes about 60 hours. Computational cost mainly consists of two parts: sampling and training. In training, for a sample x_0, the vanilla training using RL algorithm needs to traverse the entire denoising process from x_T to x_0, while the training using BPT only needs to traverse from x_τ to x_0, where $\tau \leq T$. Therefore, using BPT leads to lower training cost in training. As for sampling, using branch-based sampling (BS) indeed leads to higher computational cost. However, sampling is much faster than training, so B^2-DiffuRL requires lower computational cost overall.

D.4. Experimental Resources

We conducted experiments on 8 24GB NVIDIA 3090 GPUs. It took approximately 36 hours to reach 25.6k reward queries when rewarded by CLIPScore, and approximately 80 hours when rewarded by BERTScore (LLaVA inference would take much time).

D.5. Hyperparameters

We list hyperparameters of our experiments in Table 6.

	Hyperpatameter	B^2-DiffuRL	DDPO
Sampling	Denoising steps T	20	20
	Noise Weight η	1.0	1.0
	Guidance Scale	5.0	5.0
	Batch size	8	8
	Batch count	32	32
	Number of Branches	3	-
Optimizer	Optimizer	AdamW [37]	AdamW
	Learning rate	1e-4	1e-4
	Weight decay	1e-4	1e-4
	(β_1, β_2)	(0.9, 0.999)	(0.9, 0.999)
	ϵ	1e-8	1e-8
Training	Batch size	2	2
	Grad. accum. steps	32	128
	Initial training interval	[14, 1]	-
	Score threshold	0.5	-

Table 6. Hyperparameters of our experiments.

E. Pseudo-code

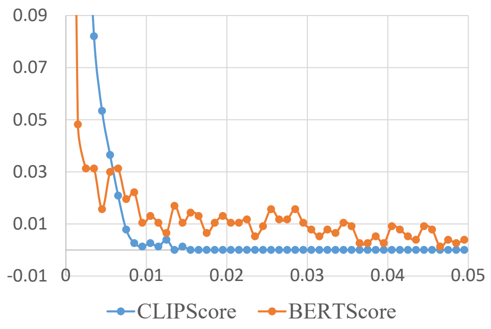
The pseudo-code of B^2-DiffuRL for one training round goes as Algorithm 1.

F. Discussion on Evaluation Metrics

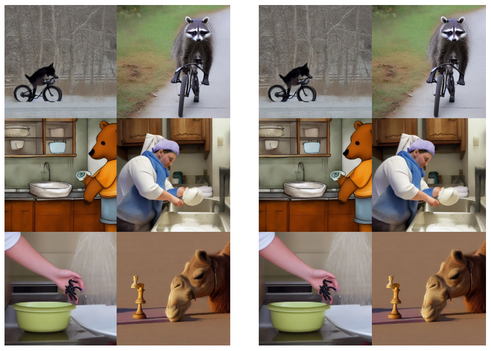
F.1. Comparison between BERTScore and CLIPScore

We create a dataset containing 768 pairs of similar images generated by diffusion models with 20 denoising steps. The two images in the same pair share the same states in the first 19 denoising steps, and only differ in the last denosing step. Some examples are shown in Figure 12 (b), and we can't tell the difference between them with our eyes. But they are different images, since their file size in JPEG format are different. Since images in same pairs are visually indistinguishable, they should receive similar prompt-image alignment scores.

However, BERTScores of similar image pairs differ a lot in our observation. Figure 12 (a) shows the distribution curves of score differences for similar image pairs, evaluated on CLIPScore and BERTScore. For CLIPScore, we can observe that almost all similar images have a score difference of less than 0.01. But for BERTScore, in the interval where the score difference is greater than 0.01, there are still many similar image pairs. As we can see from Figure 6 (a), after fine-tuning the model, BERTScores of the generated images increase by 0.01-0.03. In consideration of accurate rewarding and evaluation, it is intolerable that the score difference of similar images is greater than 0.01. Therefore, we recommend using CLIPScore as reward function instead of BERTScore.



(a) Distribution of score differences for similar image pairs.



(b) Examples of similar image pairs.

Figure 12. (a) Distribution curve of score differences for similar image pairs when evaluated by CLIPScore and BERTScore. (b) Examples of similar image pairs.

F.2. Introduction to Inception Score

Following previous works [2, 4, 7, 73], we use inception score (IS) as the metric of image diversity. Inception score is primarily applied as an evaluation metric for GANs [18]. It uses a pretrained inception v3 model [61] to predict the conditional label distribution $P(y \mid x)$. Then the inception score is calculated as detailed in Eq. (9):

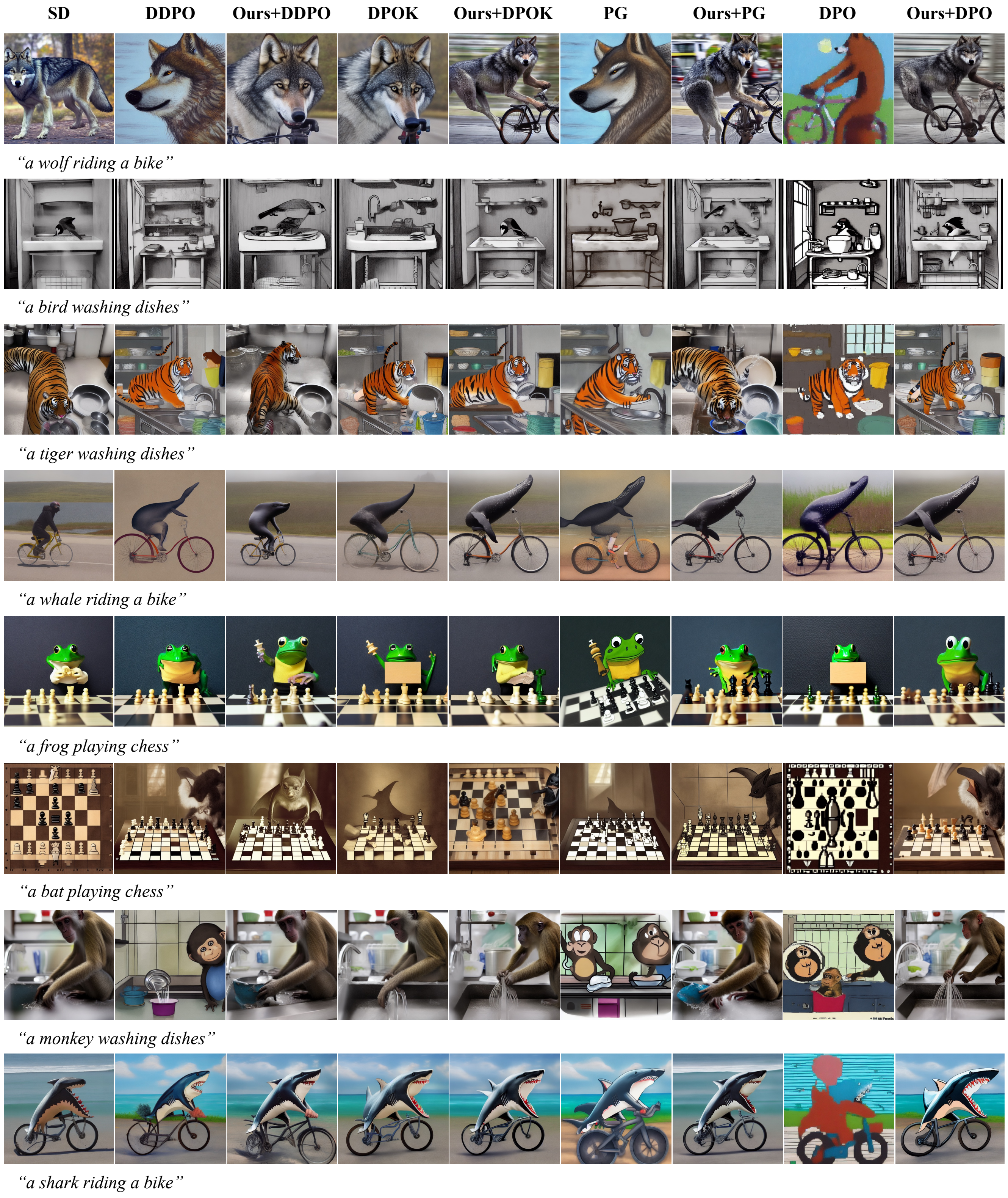
$$IS = \exp(\mathbb{E}_x(KL(p(y \mid x)||P(y)))), \tag{9}$$

where KL is Kullback-Leibler divergence. Traditional Inception v3 is trained only on ImageNet [12], while Stable Diffusion is trained on a large-scale dataset. In real implementation, in order to better measure the diversity of images, we replace it by the image encoder of CLIP for calculating IS. A higher inception score represents better image diversity.

G. More Samples

In this section, we show more samples generated by the diffusion models fine-tuned with our method B^2-DiffuRL. Figure 13 shows more samples generated by our method compared with DDPO, DPO, PG and DPO on templeta 1. Figure 14 and 15 show more samples from our method on template 2 and 3 respectively. Figure 16 shows more samples of generalization to unseen prompts.

In Figure 17, more samples are generated on three given prompts to show the diversity of different methods. As we can see, most images generated by DDPO adopt a cartoon-like style, as described in their paper [8]. Especially for the images generated on the prompt "*a fox riding a bike*", almost all the background information is lost and becomes a single color. On the contrary, the images generated by our method can almost keep the same style as SD, mitigating the problem of diversity reduction.



SD
DDPO
Ours+DDPO
DPOK
Ours+DPOK
PG
Ours+PG
DPO
Ours+DPO
"a wolf riding a bike"
"a bird washing dishes"
"a tiger washing dishes"
"a whale riding a bike"
"a frog playing chess"
"a bat playing chess"
"a monkey washing dishes"
"a shark riding a bike"

Figure 13. More samples generated by our method compared with other methods on template 1.

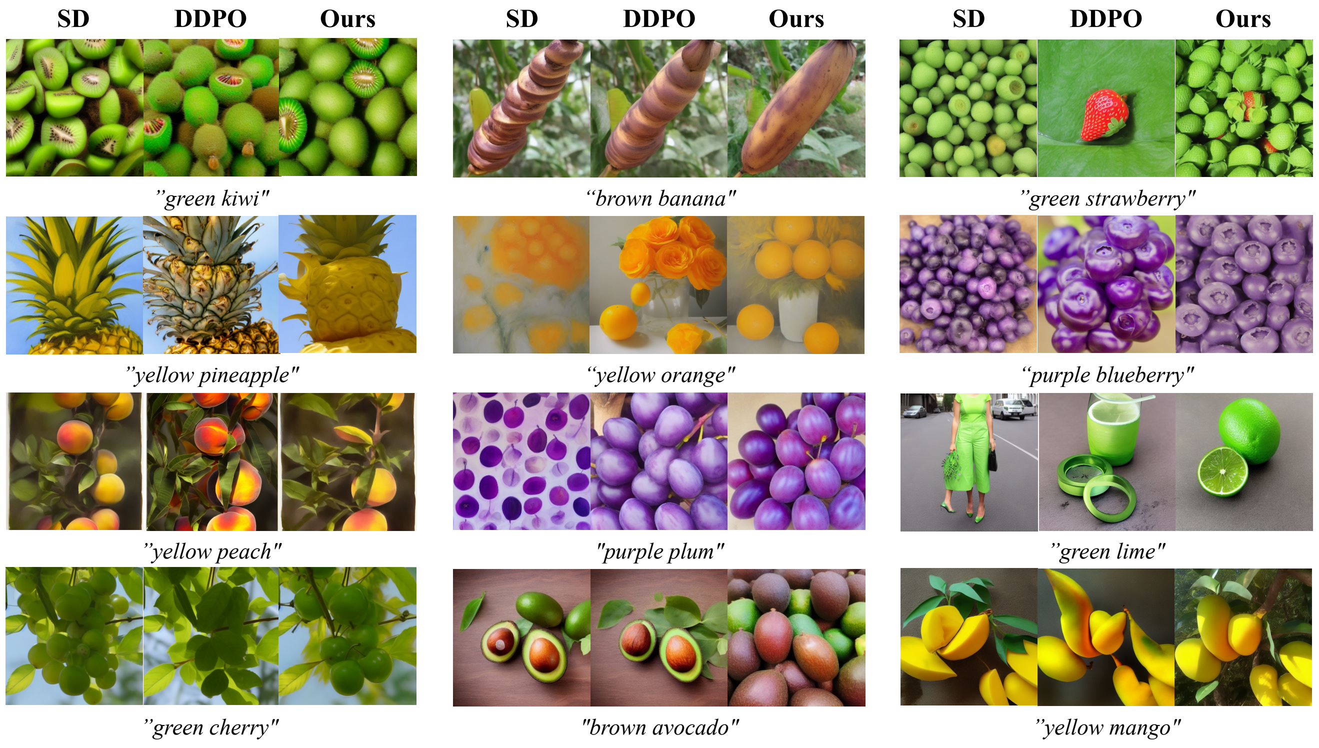


Figure 14. More samples generated by our method on template 2.

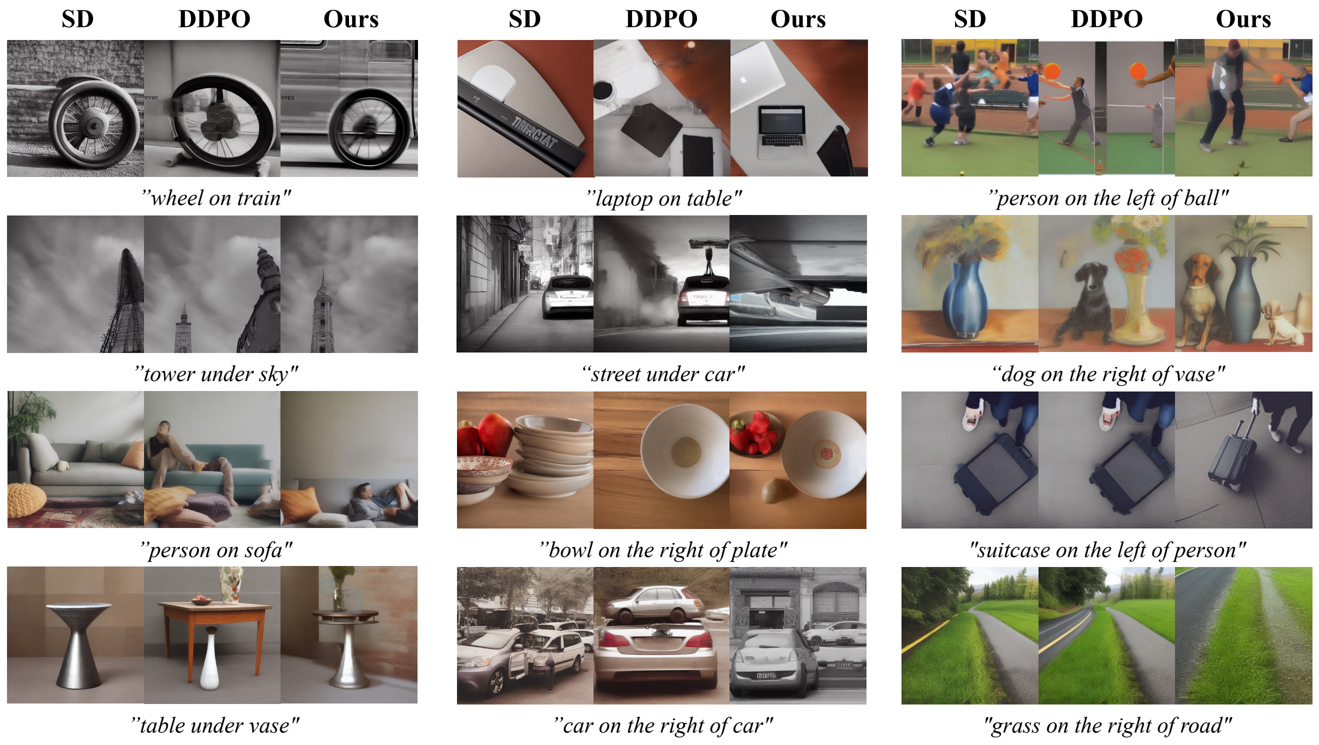

SD DDPO Ours
SD DDPO Ours
SD DDPO Ours
”wheel on train"
”laptop on table"
”person on the left of ball"
”tower under sky"
“street under car"
“dog on the right of vase"
”person on sofa"
”bowl on the right of plate"
"suitcase on the left of person"
”table under vase"
”car on the right of car"
"grass on the right of road"

Figure 15. More samples generated by our method on template 3.

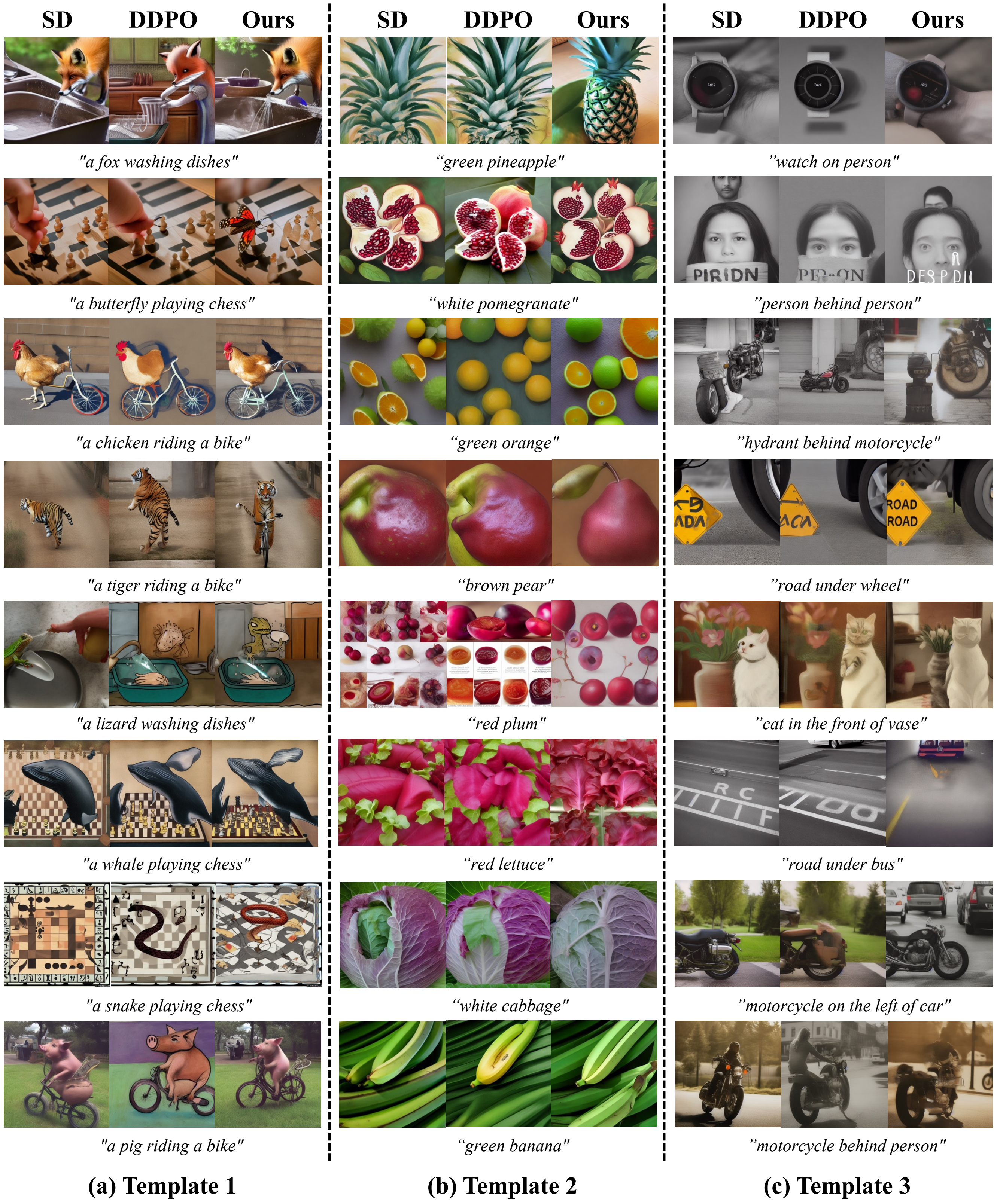


Figure 16. More samples of generalization to unseen prompts in template 1, 2 and 3.

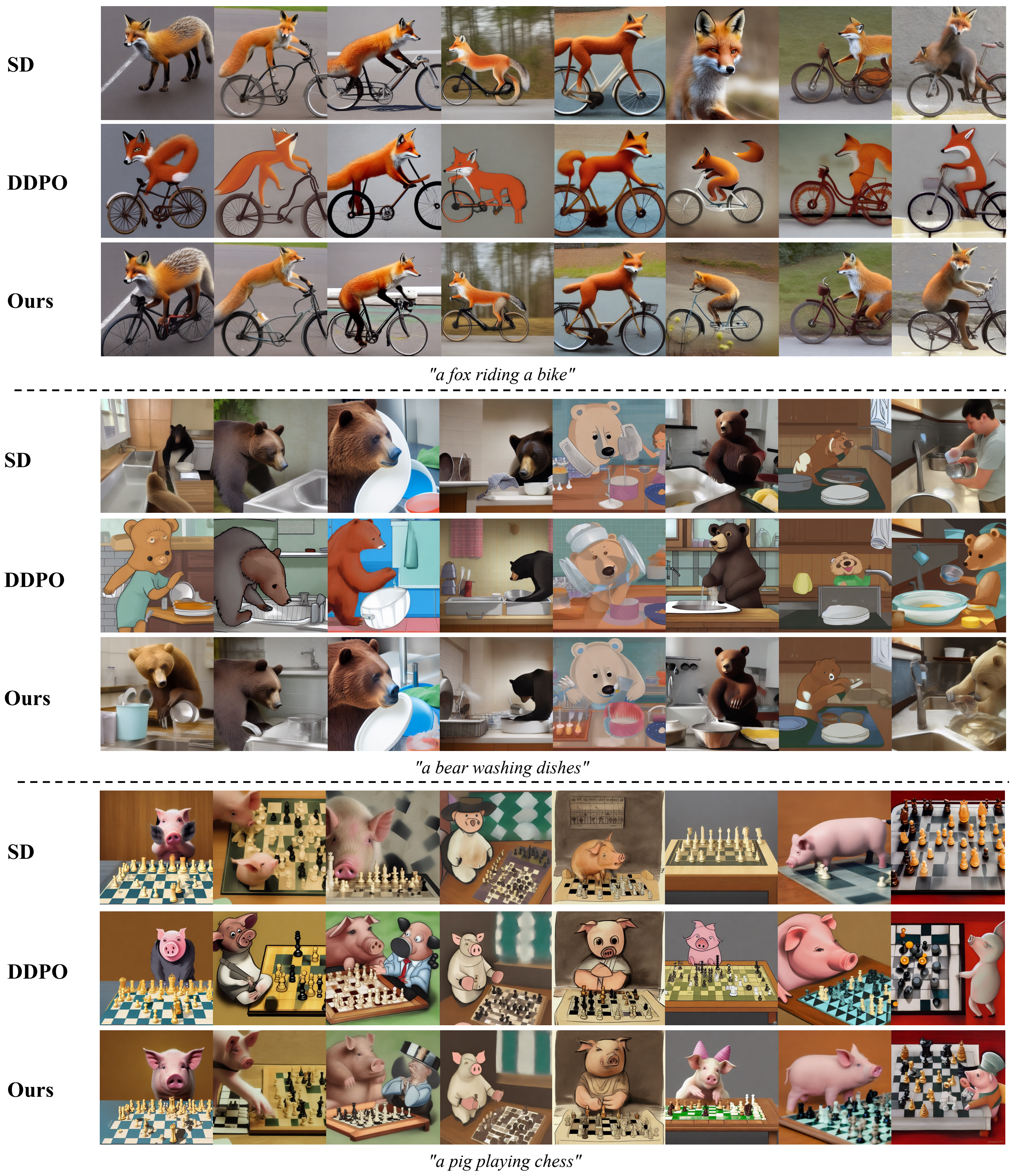


Figure 17. More samples generated by SD, DDPO and our method on three prompts. The images generated by DDPO tend to adopt a cartoon style, while those by our method tend to keep original styles of SD. These samples show that our method can help mitigating the image diversity reduction during fine-tuning.

H. Prompt Lists

In this section, we provide the prompt lists used in our experiments. For each template, we collect one prompt list for training, and the other one for generalization test, as shown in Table 7, 8 and 9.

Training list		
a cat washing dishes	a dog washing dishes	a horse washing dishes
a monkey washing dishes	a rabbit washing dishes	a zebra washing dishes
a spider washing dishes	a bird washing dishes	a sheep washing dishes
a deer washing dishes	a cow washing dishes	a goat washing dishes
a lion washing dishes	a tiger washing dishes	a bear washing dishes
a raccoon riding a bike	a fox riding a bike	a wolf riding a bike
a lizard riding a bike	a beetle riding a bike	a ant riding a bike
a butterfly riding a bike	a fish riding a bike	a shark riding a bike
a whale riding a bike	a dolphin riding a bike	a squirrel riding a bike
a mouse riding a bike	a rat riding a bike	a snake riding a bike
a turtle playing chess	a frog playing chess	a chicken playing chess
a duck playing chess	a goose playing chess	a bee playing chess
a pig playing chess	a turkey playing chess	a fly playing chess
a llama playing chess	a camel playing chess	a bat playing chess
a gorilla playing chess	a hedgehog playing chess	a kangaroo playing chess
Test list		
a cat riding a bike	a cat playing chess	a dog riding a bike
a dog playing chess	a horse riding a bike	a horse playing chess
a monkey riding a bike	a monkey playing chess	a rabbit riding a bike
a rabbit playing chess	a zebra riding a bike	a zebra playing chess
a spider riding a bike	a spider playing chess	a bird riding a bike
a bird playing chess	a sheep riding a bike	a sheep playing chess
a deer riding a bike	a deer playing chess	a cow riding a bike
a cow playing chess	a goat riding a bike	a goat playing chess
a lion riding a bike	a lion playing chess	a tiger riding a bike
a tiger playing chess	a bear riding a bike	a bear playing chess
a raccoon washing dishes	a raccoon playing chess	a fox washing dishes
a fox playing chess	a wolf washing dishes	a wolf playing chess
a lizard washing dishes	a lizard playing chess	a beetle washing dishes
a beetle playing chess	a ant washing dishes	a ant playing chess
a butterfly washing dishes	a butterfly playing chess	a fish washing dishes
a fish playing chess	a shark washing dishes	a shark playing chess
a whale washing dishes	a whale playing chess	a dolphin washing dishes
a dolphin playing chess	a squirrel washing dishes	a squirrel playing chess
a mouse washing dishes	a mouse playing chess	a rat washing dishes
a rat playing chess	a snake washing dishes	a snake playing chess
a turtle washing dishes	a turtle riding a bike	a frog washing dishes
a frog riding a bike	a chicken washing dishes	a chicken riding a bike
a duck washing dishes	a duck riding a bike	a goose washing dishes
a goose riding a bike	a bee washing dishes	a bee riding a bike
a pig washing dishes	a pig riding a bike	a turkey washing dishes
a turkey riding a bike	a fly washing dishes	a fly riding a bike
a llama washing dishes	a llama riding a bike	a camel washing dishes
a camel riding a bike	a bat washing dishes	a bat riding a bike
a gorilla washing dishes	a gorilla riding a bike	a hedgehog washing dishes
a hedgehog riding a bike	a kangaroo washing dishes	a kangaroo riding a bike

Table 7. Prompt Lists for template 1.

Training list		
red apple	green apple	yellow banana
brown banana	orange orange	yellow orange
red strawberry	green strawberry	purple grape
green grape	red watermelon	green watermelon
brown kiwi	green kiwi	orange mango
yellow mango	green pear	yellow pear
yellow pineapple	brown pineapple	orange peach
yellow peach	purple plum	green plum
blue blueberry	purple blueberry	red raspberry
green raspberry	yellow lemon	green lemon
green lime	yellow lime	green avocado
brown avocado	red cherry	green cherry
red pomegranate	pink pomegranate	pink grapefruit
red grapefruit		
Test list		
yellow apple	green banana	green orange
white strawberry	black grape	white watermelon
white kiwi	green mango	brown pear
green pineapple	red peach	red plum
black blueberry	black raspberry	white lemon
white lime	yellow avocado	black cherry
white pomegranate	yellow grapefruit	white carrot
white broccoli	yellow tomato	white cucumber
brown spinach	red lettuce	yellow bell pepper
white zucchini	white sweet potato	green onion
green garlic	white celery	white cabbage
purple cauliflower	green eggplant	purple asparagus
white peas	green corn	purple green beans
white brussels sprouts		

Table 8. Prompt lists for template 2.

Training list		
chair under umbrella	table under umbrella	car on street
wheel on train	airplane on street	bag on street
tree under sky	building under sky	street under sky
dog on boat	tower under sky	cup on shirt
person on street	laptop on table	table under laptop
person on sofa	glasses on face	sofa under person
table under vase	street under car	dog on the right of vase
building on the right of building	suitcase on the left of person	dog on the left of person
kite on the right of kite	person on the left of ball	ball on the right of person
road on the left of grass	grass on the right of road	person on the left of pillow
bowl on the right of plate	building on the right of truck	person on the left of bottle
bottle on the right of person	box on the left of post	building on the left of building
car on the right of car	truck on the right of car	car on the left of car
person on the left of person		
Test list		
vase on table	shirt on person	watch on person
jacket on person	motorcycle on road	motorcycle behind person
person behind person	building behind trees	hydrant behind motorcycle
trees behind grass	wheel in the front of wheel	tower in the front of train
truck in the front of building	cat in the front of vase	trash can in the front of cabinet
road under bus	road under building	road under wheel
table under plate	person under umbrella	cone on the right of cone
car on the right of umbrella	phone on the right of monitor	person on the right of bear
bear on the right of person	bear on the left of person	car on the left of bus
motorcycle on the left of bus	motorcycle on the left of car	road on the left of tree

Table 9. Prompt lists for template 3.