A. Correctness of the Particle Filtering Framework

We now prove the correctness of our resampling weight design in the particle filtering algorithm. Denote the distribution that $\{\boldsymbol{x}_t^{(k)}\}$ follows as $v(\boldsymbol{X}_t|\boldsymbol{C})$, based on the particle filtering process, it is easy to show the probability satisfies the following recursive relationship:

$$v(\boldsymbol{X}_t|\boldsymbol{C}) \propto \int v(\boldsymbol{X}_{t+1}|\boldsymbol{C}) r(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{C}) w(\boldsymbol{X}_t, \boldsymbol{X}_{t+1}|\boldsymbol{C}) \,\mathrm{d}\boldsymbol{X}_{t+1}, \tag{14}$$

where $r(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{C})$ is the proposal distribution and $w(\boldsymbol{X}_t, \boldsymbol{X}_{t+1}|\boldsymbol{C})$ is the resampling weight.

Now consider $r(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{C}) = q(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{C})$, *i.e.,* the diffusion model is used for proposal, and $w(\boldsymbol{X}_t, \boldsymbol{X}_{t+1}|\boldsymbol{C}) = \frac{\phi(\boldsymbol{X}_t|\boldsymbol{C})}{\phi(\boldsymbol{X}_{t+1}|\boldsymbol{C})}$. Suppose $v(\boldsymbol{X}_{t+1}|\boldsymbol{C}) = q(\boldsymbol{X}_{t+1}|\boldsymbol{C})\phi(\boldsymbol{X}_{t+1}|\boldsymbol{C})$, using the recursive relationship in Eq. (14), it can be shown that

$$\begin{aligned}
&v(\boldsymbol{X}_t|\boldsymbol{C}) \propto \\
&\int v(\boldsymbol{X}_{t+1}|\boldsymbol{C}) q(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{C}) w(\boldsymbol{X}_t, \boldsymbol{X}_{t+1}|\boldsymbol{C}) \,\mathrm{d}\boldsymbol{X}_{t+1} \\
&= \int q(\boldsymbol{X}_{t+1}|\boldsymbol{C}) \phi(\boldsymbol{X}_{t+1}|\boldsymbol{C}) q(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{C}) \frac{\phi(\boldsymbol{X}_t|\boldsymbol{C})}{\phi(\boldsymbol{X}_{t+1}|\boldsymbol{C})} \,\mathrm{d}\boldsymbol{X}_{t+1} \\
&= \int q(\boldsymbol{X}_t, \boldsymbol{X}_{t+1}|\boldsymbol{C}) \phi(\boldsymbol{X}_t|\boldsymbol{C}) \,\mathrm{d}\boldsymbol{X}_{t+1} \\
&= q(\boldsymbol{X}_t|\boldsymbol{C}) \phi(\boldsymbol{X}_t|\boldsymbol{C}).
\end{aligned} \tag{15}$$

At time step T, let $\phi(\boldsymbol{X}_T|\boldsymbol{C}) = 1$, and thus $v(\boldsymbol{X}_T|\boldsymbol{C}) = q(\boldsymbol{X}_T|\boldsymbol{C})\phi(\boldsymbol{X}_T|\boldsymbol{C})$. Therefore, by mathematical induction, $v(\boldsymbol{X}_t|\boldsymbol{C}) = q(\boldsymbol{X}_t|\boldsymbol{C})\phi(\boldsymbol{X}_t|\boldsymbol{C})$ for all $t = 0, \ldots, T$.

Algorithm 1 describes the overall procedure of our particle filtering framework.

B. Resampling Weight Calculation

In this section, we provide the detailed process for calculating resampling weights, and more specifically, the correction term $\phi(\boldsymbol{X}_t|\boldsymbol{C})$. Algorithm 2 describes the overall procedure. We will mainly focus on the hybrid approach. For the discriminator-based approach, please refer to Section 3.5.

Recall that in the hybrid approach, $\phi(\boldsymbol{X}_t|\boldsymbol{C})$ consists of two terms: unconditional likelihood ratio $\frac{p(\boldsymbol{X}_t)}{q(\boldsymbol{X}_t)}$ and object mention ratio $\frac{p(\boldsymbol{O}_C|\boldsymbol{X}_t)}{q(\boldsymbol{O}_C|\boldsymbol{X}_t)}$. The unconditional likelihood ratio can be estimated with an unconditional discriminator as in Eq.(7). We will next elaborate on how to estimate the object mention ratio.

Estimating object mention ratio. As shown in Eq. (9), the object mention ratio can be decomposed as

$$\frac{\prod_{i:O_{Ci}=1} p(O_{Ci}=1|\boldsymbol{X}_t)}{\prod_{i:O_{Ci}=1} q(O_{Ci}=1|\boldsymbol{X}_t)} \cdot \frac{\prod_{i:O_{Ci}=0} p(O_{Ci}=0|\boldsymbol{X}_t)}{\prod_{i:O_{Ci}=0} q(O_{Ci}=0|\boldsymbol{X}_t)},$$

where we focus on the first term that corresponds to the missing object errors.

To compute $p(O_{Ci}=1|\boldsymbol{X}_t)$, notice that under p distribution, $O_{Ci} = O_{Xi}$, because, by definition, a text caption is constructed to mention the objects that appear in the corresponding real image. Therefore,

$$\begin{aligned}
p(O_{Ci}=1|\boldsymbol{X}_t) &= p(O_{Xi}=1|\boldsymbol{X}_t) \\
&\overset{①}{=} \mathbb{E}_{\boldsymbol{X}_0 \sim p(\boldsymbol{X}_0|\boldsymbol{X}_t)}[p(O_{Xi}=1|\boldsymbol{X}_0)] \\
&\overset{②}{\approx} \hat{p}(O_{Xi}=1|f(\boldsymbol{X}_t)),
\end{aligned} \tag{16}$$

where equality ① is by simple chain rule and the fact that $p(O_{Xi}|\boldsymbol{X}_0, \boldsymbol{X}_t) = p(O_{Xi}|\boldsymbol{X}_0)$ because O_{Xi} is defined as the object occurrence in the clean image $\boldsymbol{X}_0$, and hence is conditionally independent of $\boldsymbol{X}_t$. For ②, we make two approximations. First, we replace drawing a sample of $\boldsymbol{X}_0$ from $p(\boldsymbol{X}_0|\boldsymbol{X}_t)$ with the minimum mean squared error (MMSE) estimate of $\boldsymbol{X}_0$ from $\boldsymbol{X}_t$, denoted as $f(\boldsymbol{X}_t)$, which can be conveniently and efficiently predicted by the denoising network via one-step generation of $\boldsymbol{X}_0$ from $\boldsymbol{X}_t$. Second, we approximate the true object occurrence probability $p(O_{Xi}|\boldsymbol{X}_0)$ with the one estimated by the object detector $\hat{p}(O_{Xi}|\boldsymbol{X}_0)$.

On the other hand, $q(O_{Ci}=1|\boldsymbol{X}_t)$ can be most intuitively computed by training a neural network to predict object mentions in the input text from the generated samples. However, this approach is inefficient and would need to retrain the network everytime the denoising scheme changes. We would like to derive a more efficient alternative taking advantage of the object detector. Formally (notice that it is no longer true that $O_{Ci} = O_{Xi}$),

$$\begin{aligned}
q(O_{Ci}=1|\boldsymbol{X}_t) &= \mathbb{E}_{O_{Xit} \sim q(O_{Xit}|\boldsymbol{X}_t)}[q(O_{Ci}=1|O_{Xit}, \boldsymbol{X}_t)] \\
&= q(O_{Xit}=0|\boldsymbol{X}_t) q(O_{Ci}=1|O_{Xit}=0, \boldsymbol{X}_t) + \\
&\quad q(O_{Xit}=1|\boldsymbol{X}_t) q(O_{Ci}=1|O_{Xit}=1, \boldsymbol{X}_t),
\end{aligned} \tag{17}$$

where O_{Xit} denotes the occurrence of object i in the clean image predicted from $\boldsymbol{X}_t$, *i.e.,* $f(\boldsymbol{X}_t)$. Similar to Eq. (16), the probability $q(O_{Xit}|\boldsymbol{X}_t)$ can be estimated by $\hat{p}(O_{Xi}|f(\boldsymbol{X}_t))$.Therefore, what makes Eq. (17) different is the additional term $q(O_{Xit}=0|\boldsymbol{X}_t)q(O_{Ci}=1|O_{Xit}=0, \boldsymbol{X}_t)$, which corresponds to the case where even if the object detector fails to detect the object i, there is still chance that the caption used to generate the image contains object i, considering the imperfect diffusion model that may miss objects mentioned in the caption.

To calculate $q(O_{Ci}=1|O_{Xit}, \boldsymbol{X}_t)$, we further make an assumption that object occurrence O_{Xit} is a sufficient statistic to predict the object mention of the same object. Hence $q(O_{Ci}=1|O_{Xit}, \boldsymbol{X}_t) = q(O_{Ci}=1|O_{Xit})$. According to the Bayes rule,

$$q(O_{Ci}=1|O_{Xit}) = \frac{q(O_{Xit}|O_{Ci}=1)q(O_{Ci}=1)}{\sum_{O_{Ci} \in \{0,1\}} q(O_{Xit}|O_{Ci})q(O_{Ci})}. \tag{18}$$

Algorithm 1 Particle Filtering Framework for Correcting Diffusion Generation

Input:
- Diffusion model with denosing distribution $q(\boldsymbol{X}_t|\boldsymbol{X}_{t+1}, \boldsymbol{C})$, condition signal $\boldsymbol{c}$, number of particles K.
- Note: operations involving index k are performed for $k \in \{1, \dots, K\}$.

Initialization:
Sample $\boldsymbol{x}_T^{(k)} \sim q(\boldsymbol{X}_T)$, set $\phi(\boldsymbol{x}_T^{(k)}|\boldsymbol{c}) = 1$
for $t = T-1$ to 0 **do** ▷ Iterate over time steps
  $\tilde{\boldsymbol{x}}_t^{(k)} \sim q(\boldsymbol{X}_t|\boldsymbol{x}_{t+1}^{(k)}, \boldsymbol{c})$ ▷ Proposal
  $\phi(\tilde{\boldsymbol{x}}_t^{(k)}|\boldsymbol{c}) = \textsc{CalcCorrection}\left(\tilde{\boldsymbol{x}}_t^{(k)}, \boldsymbol{c}, t\right)$ ▷ Calculate correction term in Eq. (5)
  $w(\boldsymbol{x}_{t+1}^{(k)}, \tilde{\boldsymbol{x}}_t^{(k)}|\boldsymbol{c}) = \dfrac{\phi(\tilde{\boldsymbol{x}}_t^{(k)}|\boldsymbol{c})}{\phi(\boldsymbol{x}_{t+1}^{(k)}|\boldsymbol{c})}$ ▷ Calculate resampling weight
  $\boldsymbol{x}_t^{(k)} \sim \textsc{Multinomial}\left(\{\tilde{\boldsymbol{x}}_t^{(k)}\}_{k=1}^K; \{w(\boldsymbol{x}_{t+1}^{(k)}, \tilde{\boldsymbol{x}}_t^{(k)}|\boldsymbol{c})\}_{k=1}^K\right)$ ▷ Resampling
end for

Output:
$\{\boldsymbol{x}_0^{(k)}\}_{k=1}^K$ that approximately follow the ground-truth distribution $p(\boldsymbol{X}_0|\boldsymbol{c})$.

Algorithm 2 CalcCorrection

Input:
- Sample $\tilde{\boldsymbol{x}}_t$, conditional signal $\boldsymbol{c}$, time step t.
- Conditional discriminator $d(\boldsymbol{X}_t|\boldsymbol{C}; t)$, unconditional discriminator $d(\boldsymbol{X}_t; t)$, object detector $\hat{p}(O_{Xi} = 1|\boldsymbol{X}_0)$, statistics κ_{it} estimated from historically generated images, hyper-parameter π_{it}.

if use discriminator-based approach **then** ▷ Section 3.5
  $\phi(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}) = \dfrac{d(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}; t)}{1 - d(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}; t)}$ ▷ Eq. (7)
else if use hybrid approach **then** ▷ Section 3.6
  $\dfrac{p(\tilde{\boldsymbol{x}}_t)}{q(\tilde{\boldsymbol{x}}_t)} = \dfrac{d(\tilde{\boldsymbol{x}}_t; t)}{1 - d(\tilde{\boldsymbol{x}}_t; t)}$ ▷ Unconditional likelihood ratio
  $p(O_{Ci} = 1|\tilde{\boldsymbol{x}}_t) \approx \hat{p}(O_{Xi} = 1|f(\tilde{\boldsymbol{x}}_t))$ ▷ Eq. (10)
  $q(O_{Ci} = 1|\tilde{\boldsymbol{x}}_t) \approx \hat{p}(O_{Xi} = 1|f(\tilde{\boldsymbol{x}}_t)) + \hat{p}(O_{Xi} = 0|f(\tilde{\boldsymbol{x}}_t))\frac{(1-\kappa_{it})\pi_{it}}{(1-\kappa_{it})\pi_{it}+1-\pi_{it}}$ ▷ Eq. (11)
  $\phi_t(\tilde{\boldsymbol{x}}_t|\boldsymbol{c}) = \dfrac{p(\tilde{\boldsymbol{x}}_t)}{q(\tilde{\boldsymbol{x}}_t)} \cdot \dfrac{\prod_{i:O_{Ci}=1} p(O_{Ci} = 1|\tilde{\boldsymbol{x}}_t)}{\prod_{i:O_{Ci}=1} q(O_{Ci} = 1|\tilde{\boldsymbol{x}}_t)}$ ▷ Eq. (13)
end if

Output:
- Correction term $\phi_t(\tilde{\boldsymbol{x}}_t|\boldsymbol{c})$.

Both $q(O_{Xit}|O_{Ci} = 1)$ and $q(O_{Xit}|O_{Ci} = 0)$ can be estimated from the samples of distribution $q(\boldsymbol{X}_t|\boldsymbol{C})$. Specifically, consider a different set of H samples $\{\boldsymbol{x}_t^{(h)}\}$ that are generated without particle filtering process. For each sample $\boldsymbol{x}_t^{(h)}$, we compute the object occurrence estimate $\hat{O}_{Xit}$ by feeding the predicted clean image, $f(\boldsymbol{x}_t^{(h)})$, to the object detector, and thresholding the output probability of the object detector with 0.5. Then $q(O_{Xit} = 1|O_{Ci} = 1)$ can be estimated by calculating the percentage of samples whose $\hat{O}_{Xit} = 1$ among the samples whose corresponding input caption mentions object i, as in Eq. (12). $q(O_{Xit} = 1|O_{Ci} = 0)$ can be calculated similarly from samples whose input caption does not mention object i. In practice, since the diffusion model seldom generates objects that are not mentioned in the text, we empirically find that $q(O_{Xit} = 1|O_{Ci} = 0)$ is very close to zero. Hence, to reduce computation complexity, we will set this term to zero, and $q(O_{Xit} = 0|O_{Ci} = 0)$ to one. Finally, $q(O_{Ci} = 0)$ and $q(O_{Ci} = 1)$ controls how much weight we would like to place on each case, and we will treat them as hyper-parameters and denote $q(O_{Ci} = 1)$ as π_{it}. Note that although π_{it} can be dependent on both i and t, for simplicity, we set π_{it} to be the same for all i and t.

Denote $q(O_{Xit} = 1|O_{Ci} = 1)$ as κ_{it}, with the above as-

sumptions,

$$q(O_{Ci}=1|O_{Xit}=1)=1$$
$$q(O_{Ci}=1|O_{Xit}=0)=\frac{(1-\kappa_{it})\pi_{it}}{(1-\kappa_{it})\pi_{it}+1-\pi_{it}}. \tag{19}$$

Plug Eq. (19) back to Eq. (17), we can derive Eq. (11) in the main paper.

C. Experiments on Text-to-image Generation

C.1. Reparameterize Stable Diffusion for Reverse-time SDE

Following Karras et al. [28] and Xu et al. [64], we use the reverse-time SDE [1] for generation:

$$\mathrm{d}\boldsymbol{x}=-2\dot{\sigma}(t)\sigma(t)\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma(t))\,\mathrm{d}t+\sqrt{2\dot{\sigma}(t)\sigma(t)}\,\mathrm{d}\omega_t, \tag{20}$$

where ω_t is the standard Wiener process, $\sigma(t)=t$ is the noise level at time t, and $p(\boldsymbol{x};\sigma(t))$ is the distribution obtained by adding Gaussian noise of standard deviate $\sigma(t)$ to the input data.

To use stable diffusion [47] in Eq. (20), the key is to reparameterize the model to estimate the score $\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma(t))$. Following Karras et al. [28], we estimate the score as follows:

$$\nabla_{\boldsymbol{x}}\log p(\boldsymbol{x};\sigma)=\frac{D(\boldsymbol{x};\sigma)-\boldsymbol{x}}{\sigma^2},$$
$$D(\boldsymbol{x};\sigma)=c_{\text{skip}}(\sigma)\boldsymbol{x}+c_{\text{out}}(\sigma)F\left(c_{\text{in}}(\sigma)\boldsymbol{x};c_{\text{noise}}(\sigma)\right), \tag{21}$$

where $F(\cdot)$ is the denoising network trained in stable diffusion, $c_{\text{skip}}(\sigma)=1$, $c_{\text{out}}(\sigma)=-\sigma$, $c_{\text{in}}(\sigma)=1/\sqrt{\sigma^2+1}$, $c_{\text{noise}}(\sigma)=999\sigma^{-1}_{\text{train}}(\sigma)$, $\sigma_{\text{train}}(t)$ is the noise schedule used when training the denoising network, and $\sigma^{-1}_{\text{train}}(\sigma)$ denotes the inverse of $\sigma_{\text{train}}(t)$. In particular, the forward process of Stable Diffusion v2.1-base can be considered as a discrete version of the VP SDE [55]:

$$\mathrm{d}\boldsymbol{x}=f(t)\boldsymbol{x}\,\mathrm{d}t+g(t)\,\mathrm{d}\omega_t, \tag{22}$$

where

$$f(t)=-\frac{1}{2}\beta(t),\quad g(t)=\sqrt{\beta(t)},$$
$$\beta(t)=\left((\beta_{\text{max}}^{0.5}-\beta_{\text{min}}^{0.5})t+\beta_{\text{min}}^{0.5}\right)^2, \tag{23}$$

with $\beta_{\text{min}}=0.85$, and $\beta_{\text{max}}=12$. We follow Karras et al. [28] (Eqs. (152) - (163)) to derive $\sigma_{\text{train}}(t)$ by plugging in the scaled linear schedule $\beta(t)=\left((\beta_{\text{max}}^{0.5}-\beta_{\text{min}}^{0.5})t+\beta_{\text{min}}^{0.5}\right)^2$ used in Stable Diffusion v2.1-base, which results in:

$$\sigma_{\text{train}}(t)=\sqrt{e^{\frac{1}{3}\beta_d^2t^3+\beta_d\beta_{\text{min}}^{0.5}t^2+\beta_{\text{min}}t}-1}, \tag{24}$$

where $\beta_d=\beta_{\text{max}}^{0.5}-\beta_{\text{min}}^{0.5}$.

Finally, although Stable Diffusion is trained with discrete time steps, we follow Lu et al. [38] to directly feed the continuous time value $c_{\text{noise}}(\sigma)$ to the model.

C.2. Implementation Details for Baselines

Identify objects in the caption. For methods that require the identification of objects in the caption, we use the following two steps:

1. Extract all noun phrases in the caption using spaCy.[2] Filter out noun phrases that are stop words, *e.g.,* "which" and "who".
2. Check if extracted noun phrases match one of the object categories in `MS-COCO`, and only keep the phrases that belong to `MS-COCO` objects. The match is determined by whether the Levenshtein Distance between the noun phrase and object category name is smaller than 0.1 of the length of the noun phrase and object category name.

The same process is also used to filter out 261 captions in `MS-COCO` that contain at least four objects (to measure object occurrence), and 5,000 captions in `MS-COCO` that contain at least one object (to measure FID).

Sample selection criteria. For OBJECTSELECT, we use Eq. (2) to select the best image from generated images. Specifically, since the object detector outputs multiple proposal regions, where each proposal region comes with a predicted probability for object i, we take the maximum probability for object i over all proposal regions as $\hat{p}(O_{Xi}=1|\boldsymbol{x}_0)$. For REWARDSELECT and TIFASELECT, we use the official implementation [29] to select images, except that we only include object-related questions for TIFASELECT since our focus is on object occurrence.

For other baselines, we use the official implementation to generate images. For SPATIAL-TEMPORAL, we only optimize for five rounds since we found further optimization does not improve performance. For D-GUIDANCE, we only apply guidance when $\sigma(t)<5$ since we found at large noise levels, the object detector barely detects any object, which results in a noisy gradient. All methods are evaluated on both Restart sampler [64] and EDM sampler [28]. The only exceptions are SPATIAL-TEMPORAL and ATTEND-EXCITE, which use the original sampler in their papers.

C.3. Discriminator Training

We follow Kim et al. [31] to train both unconditional and conditional discriminators. Table 3 shows the hyperparameters for discriminator training.

Training data and objective. We use LAION Aesthetics [52] to train discriminators. We randomly sample 1M images with aesthetics score higher than 6 as real images. For fake images, we generate 1M images with 49 NFE using EDM sampler. For conditional discriminator, the captions used to generate fake images are the same with the captions of the real images. For unconditional discriminator, we use a different set of 1M captions to generate fake images.

[2] https://spacy.io.

	Unconditional	Conditional
# real images	1,000,000	1,000,000
# fake images	1,000,000	1,000,000
Same captions for real & fake images?	✘	✔
# Epochs	2	2
Batch size	128	128
Learning rate	10^{-4}	10^{-4}
Time sampling	Uniform	Uniform
Total GPU hours	22	22
GPUs	A6000	A6000

Table 3. Configurations of discriminator training for text-to-image generation.

The discriminators are trained with the canonical discrimination loss in Eq. (6). Specifically, given a clean image, we uniformly sample time $t \in [10^{-5}, 1]$ and follow Eq. (22) to get the corresponding noisy image. The discriminator is then trained to discriminate noisy version of real and fake images.

Model architecture. We use the hidden representations of the middle block of the U-Net [48] to predict whether an image is real or fake, since previous works have found that these representations capture semantic information in the input image [4, 32]. Specifically, we initialize the discriminator with the U-Net in Stable Diffusion v2.1-base and only keep its down and middle blocks. We add a linear prediction layer on top the representations of the middle block. During training, we freeze the down block and only fine-tune the middle block and the prediction layer.

C.4. Sampling Configurations

For all sample selection methods (including ours), we generate each image with a fixed NFE and report performance when $K = 5, 10, 15$ images are generated.

Table 4 shows the sampling configurations for both samplers. For Restart sampler, N_{main} denotes the number of steps in the main backward process, $N_{\text{Restart},i}, K_i, t_{\text{min},i}, t_{\text{max},i}$ is the number of steps, number of repetitions, minimal, and maximum time points in the i-th restart interval, and l is the number of restart intervals. Following Xu et al. [64], we use Euler method for the main backward process and Heun method for the restart backward process. Please refer to the original paper for detailed explanations. For our PF methods, the resampling is always performed before adding noise, *i.e.,* at time $t_{\text{min},i}$, thus resampling is perfromed six times in total. For EDM sampler, N denotes the number of denoising steps, n_i denotes the index of steps when resampling is performed (for PF methods), and m denotes the number of resampling steps. Resampling is only performed four times because we empirically find performing resampling at each denoising step does not improve performance significantly compared to resampling only at a subset of steps.

Table 5 shows the comparison of computation cost for all

Restart configuration $N_{\text{main}}, \{(N_{\text{Restart},i}, K_i, t_{\text{min},i}, t_{\text{max},i})\}_{i=1}^{l}$
$30, \{(4, 1, 1.09, 1.92), (4, 2, 0.59, 1.09), (4, 2, 0.30, 0.59), (4, 1, 0.06, 0.30)\}$
EDM configuration: $N, \{n_i\}_{i=1}^{m}$
$25, \{10, 13, 16, 19\}$

Table 4. Sampling configurations for text-to-image generation.

methods. The reported cost for sample selection methods is when $K = 5$ images are generated for each caption, since their performance already exceeds other baselines at $K = 5$. Based on the table, all methods are significantly faster than SPATIAL-TEMPORAL, and sample selection methods have similar runtime with ATTEND-EXCITE on EDM sampler and D-GUIDANCE on Restart sampler. Moreover, the sample selection methods can be run in parallel much easier than other methods, so their runtime can potentially be further reduced.

C.5. Performance Breakdown by Object Category

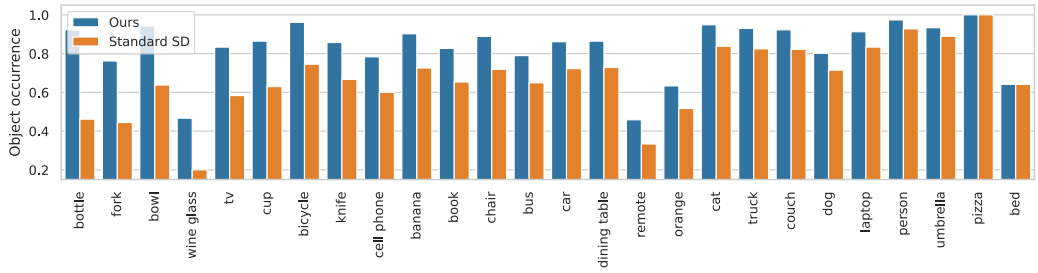
In this section, we show the performance of our method for each object category in order to study ❶ Which objects benefit more from our method; and ❷ How does the object detector affect our method. Figure 7a shows the class-wise object occurrence of our method and standard SD on MS-COCO, where classes are sorted by the amount of improvement achieved by our method and we only keep the classes that appear in at least 5 captions. As can be observed, our method improves or matches the performance of the standard SD on all objects. It is particularly beneficial on small objects with fine details, such as bottles, forks, and wine glasses. Figure 7b further shows the class-wise average precision of the object detector used during generation, evaluated on MS-COCO. As shown in the figure, the performance of our method does not correlate well with the performance of the detector. Instead, it depends more on the capability of the diffusion model in correctly generating the objects.

C.6. Additional Results with EDM Sampler

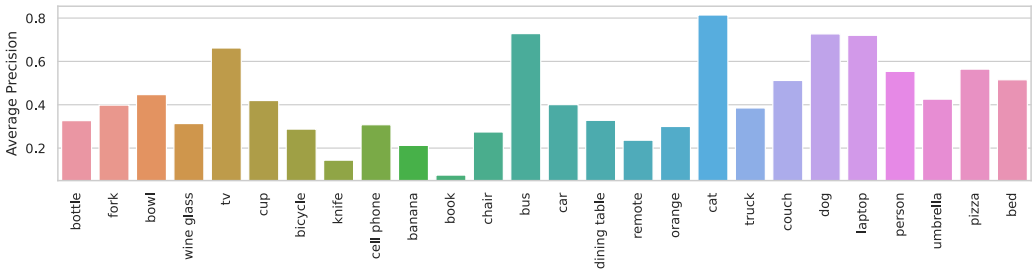
Figure 8 shows the results when EDM is used as the underlying sampler. There are two observations from the figure. First, the overall trend of all methods aligns with Figure 4 when Restart is used as the sampler. In particular, the sampling-based methods generally outperform the non-sampling-based ones. And the three methods proposed in this paper, OBJECTSELECT, PF-DISCRIMINATOR, and PF-HYBRID lie at the frontier of the performance trade-off, significantly outperform other methods. Furthermore, PF-HYBRID is the only method that simultaneously achieves a high object occurrence and a low FID. Second, the performance of EDM sampler is generally worse than that of

	EDM Sampler			Restart Sampler		
	NFE	Require Gradient?	Runtime (s)	NFE	Require Gradient?	Runtime (s)
SD [47]	49	✘	7	66	✘	9
D-GUIDANCE [31]	67	✔	20	106	✔	41
SPATIAL-TEMPORAL* [61]	250	✔	110	–	–	–
ATTEND-EXCITE* [7]	50	✔	29	–	–	–
TIFASELECT [29]	245	✘	39	330	✘	50
REWARDSELECT [29]	245	✘	32	330	✘	43
OBJECTSELECT	245	✘	32	330	✘	43
PF-DISCRIMINATOR	245	✘	33	330	✘	44
PF-HYBRID	265	✘	39	360	✘	53

Table 5. Computation cost for all methods. Runtime is measured on a single NVIDIA V100 GPU. *: the two baselines use the original samplers in their papers.



(a) Class-wise object occurrence of our method and standard SD on MS-COCO.



(b) Class-wise average precision of the object detector on MS-COCO.

Figure 7. Class-wise performance of our method and the object detector.

Restart sampler on both object occurrence and FID, which shows the superiority of the restart process.

C.7. Additional Results with SD 1.5

We also experiment with Stable Diffusion v1.5 on MS-COCO. The results are shown in Figure 9. We observe a similar trend with the results from Stable Diffusion v2.1-base (Figure 4), where our method achieves the best overall performance among baselines.

D. Experiments on Unconditional and Class-conditioned Generation

D.1. Discriminator Training

We follow the official implementation [31] to train discriminators on ImageNet-64 and FFHQ since the original paper did not experiment on the dataset or did not release the trained discriminator. Table 6 shows the training configurations. Following Kim et al. [31], the discriminators use features extracted by the frozen pre-trained classifier in Dhariwal and Nichol [13], and a shallow U-Net is added to discriminate images. During training, the pre-trained classifier is frozen and only the shallow U-Net is trained. Please refer to Kim et al. [31] for more details.

D.2. Sampling Configurations

In order to have a fair comparison in terms of computation cost, we compare all methods under the same total NFE. For PF and D-SELECT, we generate each image with a fixed NFE and vary the number of particles in generation, *i.e.,* choose value of K from $\{2, 4, 6\}$. For the original sampler and D-GUIDANCE, we increase the number of steps or

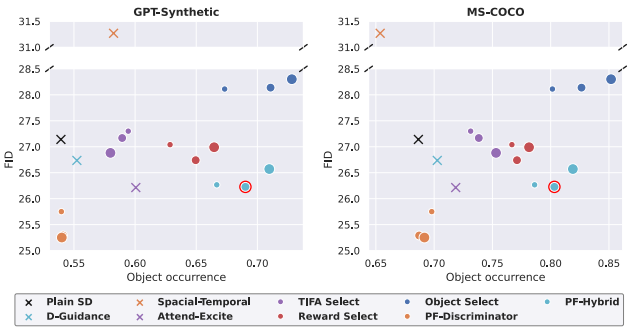


Figure 8. FID (↓) *vs.* Object occurrence (↑) for all methods evaluated with EDM sampler. Ideal points should scatter at the bottom right corner. Object occurrence is measured on `GPT-Synthetic` (left) and `MS-COCO` (right), and FID is measured on `MS-COCO`. $K = 5, 10, 15$ images are generated for sample selection methods, and the sizes of points indicate the value of K (larger K has larger points). The method that achieves the best combined performance is highlighted in red.

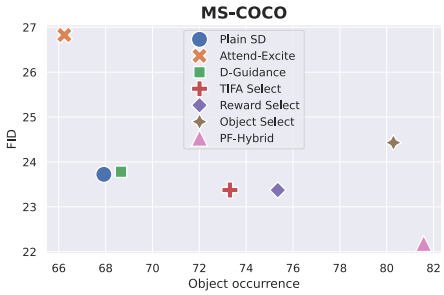


Figure 9. FID (↓) *vs.* Object occurrence (↑) for all methods evaluated with Stable Diffusion v1.5 and Restart sampler. Ideal points should scatter at the bottom right corner. Object occurrence and FID are both measured on `MS-COCO`. $K = 10$ images are generated for sample selection methods.

restart iterations to match the total NFE of sampling-based methods. All methods are evaluated on both Restart and EDM samplers. Table 7 and 8 show the sampling configurations for each sampler.

For EDM sampler, we use the same hyper-parameters as the original paper, except the S_{churn} (which controls the amount of noise added in each denoising step) on `FFHQ`, since we find adding small amount of noise ($S_{\text{churn}} > 0$) outperforms the original setting where no noise is added ($S_{\text{churn}} = 0$). Additionally, for time steps after resampling is performed, we increase S_{churn} to $(\sqrt{2} - 1)N$ (N is the number of denoising steps), which is the maximum allowed value in Karras et al. [28]. Since we find a large added noise after resampling can improve performance (please see Section E). For Restart sampler, we use the same configurations as the original paper on `ImageNet-64` and slightly modify them to adapt to `FFHQ` since the original paper did not

	`ImageNet-64`	`FFHQ`
Training Configurations		
# real images	1,200,000	60,000
# fake images	1,200,000	60,000
# Epochs	50	50
Batch size	1024	256
Learning rate	5×10^{-4}	3×10^{-4}
Time sampling	Importance	Uniform
Total GPU hours	70	4
GPUs	A6000	A6000
Shallow U-Net Architecture		
Input shape	$(B, 8, 8, 512)$	$(B, 8, 8, 512)$
Class condition	✔	✘
# Resnet blocks	4	4
# Attention blocks	3	3

Table 6. Configurations of discriminator training for class-conditioned (`ImageNet-64`) and unconditional (`FFHQ`) generation.

NFE per Image	**Resampling Steps**: $\{n_i\}_{i=1}^m$	S_{churn}
	`ImageNet-64`	
127	$\{31, 35, 39, 43, 47, 51\}$	10
255	–	20
511	–	40
767	–	60
	`FFHQ`	
63	$\{15, 19, 23, 27\}$	1.25
127	–	2.5
255	–	5.0
399	–	7.8125

Table 7. Sampling configurations for EDM sampler on standard benchmarks. Two highlighted rows are used in PF and D-SELECT to generate one image. The resampling steps are only applicable to our PF methods. S_{churn} controls the amount of noise added in each denoising step (please refer to Karras et al. [28] for details).

experiment on `FFHQ`.

D.3. Additional Results with EDM Sampler

Figure 10 shows FID as a function of total NFE when EDM is used as the sampler. Additionally, we also plot PF with Restart sampler for reference. There are three observations from the figure. First, FIDs of all methods generally decrease as NFE increases, and D-GUIDANCE, D-SELECT, and PF all outperform the original sampler. Second, our PF with Restart sampler achieves the lowest FID on both datasets. Third, the performance gain of our method when combined with EDM sampler is not as large as that when combined with Restart sampler. The difference can be ascribed to the fact that resampling is performed at smaller time steps for Restart sampler, which better selects a promising population among the particles since the dis-

NFE per Image	Restart Configuration $N_{\text{main}}, \{(N_{\text{Restart},i}, K_i, t_{\min,i}, t_{\max,i})\}_{i=1}^{l}$
	ImageNet-64
67	$18, \{(5, 1, 19.35, 40.79), (5, 1, 1.09, 1.92), (5, 1, 0.59, 1.09), (5, 1, 0.06, 0.30)\}$
99	$18, \{(3, 1, 19.35, 40.79), (4, 1, 1.09, 1.92), (4, 4, 0.59, 1.09), (4, 1, 0.30, 0.59), (4, 4, 0.06, 0.30)\}$
165	$18, \{(3, 1, 19.35, 40.79), (4, 1, 1.09, 1.92), (4, 5, 0.59, 1.09), (4, 5, 0.30, 0.59), (4, 10, 0.06, 0.30)\}$
203	$36, \{(4, 1, 19.35, 40.79), (4, 1, 1.09, 1.92), (4, 5, 0.59, 1.09), (4, 5, 0.30, 0.59), (6, 6, 0.06, 0.30)\}$
385	$36, \{(3, 1, 19.35, 40.79), (6, 1, 1.09, 1.92), (6, 5, 0.59, 1.09), (6, 5, 0.30, 0.59), (6, 20, 0.06, 0.30)\}$
535	$36, \{(6, 1, 19.35, 40.79), (6, 1, 1.09, 1.92), (7, 6, 0.59, 1.09), (7, 6, 0.30, 0.59), (7, 25, 0.06, 0.30)\}$
	FFHQ
67	$18, \{(5, 1, 1.09, 1.92), (5, 2, 0.59, 1.09), (5, 1, 0.06, 0.30)\}$
119	$18, \{(8, 1, 19.35, 40.79), (8, 2, 1.09, 1.92), (8, 2, 0.59, 1.09), (8, 1, 0.06, 0.30)\}$
251	$36, \{(11, 1, 19.35, 40.79), (11, 3, 1.09, 1.92), (11, 3, 0.59, 1.09), (11, 2, 0.06, 0.30)\}$
401	$48, \{(18, 1, 19.35, 40.79), (18, 3, 1.09, 1.92), (18, 3, 0.59, 1.09), (18, 2, 0.06, 0.30)\}$

Table 8. Sampling configurations for Restart sampler on standard benchmarks. Two highlighted rows are used in PF and D-SELECT to generate one image.

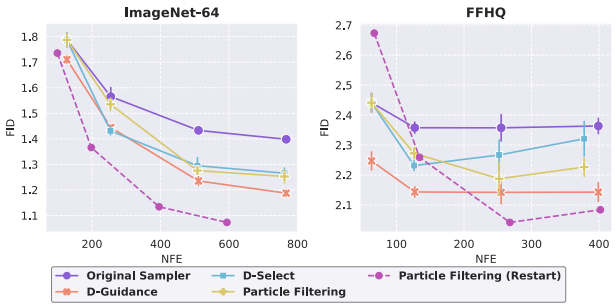


Figure 10. FID (average of 3 runs) on `ImageNet-64` (left) and `FFHQ` (right) when evaluated with EDM sampler. The PF with Restart sampler (dashed line) is added for reference. Error bars indicate standard deviations.

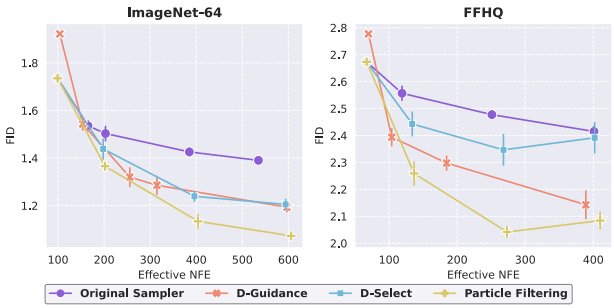


Figure 11. FID (average of 3 runs) on `ImageNet-64` (left) and `FFHQ` (right) when evaluated with Restart sampler. Error bars indicate standard deviations. The x-axis indicates the *effective NFE*, which considers all compute costs including the forward and backward passes of the discriminator. D-GUIDANCE has 1.55× effective NFE compared to the original NFE, whereas our method only has 1.02× effective NFE compared to the original NFE. The results with original NFE are shown in Figure 6.

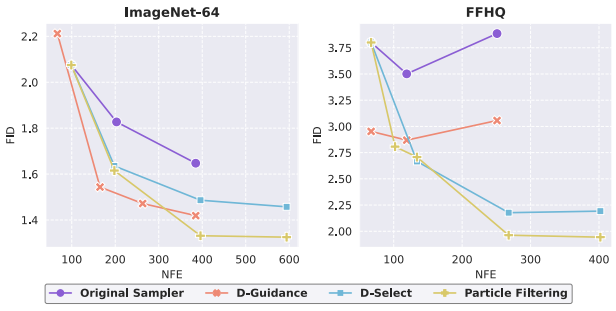
criminator can better distinguish real and model-generated images at smaller time steps. Additionally, the added noise after resampling is larger for Restart, which also benefits the exploration around the promising particles (see Section E for the impact of amount of added noise).

D.4. Additional Results with Effective NFE

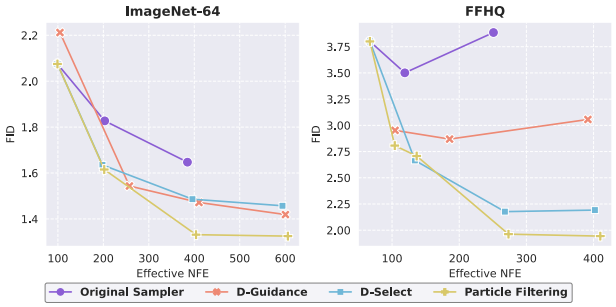
As mentioned in Section 4.2, NFE is not an adequate measure of computation cost in our setting, since it only measures the number of evaluations of the denoising U-Net and ignores other compute costs such as the forward and backward passes of the discriminator. If all these costs are considered, D-GUIDANCE incurs 1.55× compute cost per NFE compared to the original sampler, whereas our method only incurs 1.02× compute cost per NFE compared to the original sampler. To have a fair comparison, we re-plot FID against *effective NFE*, which corrects for the disparage in compute costs per NFE. We further add a new point for D-GUIDANCE at small NFE and remove a point at excessively large NFE for better comparison. The results shown in Figure 11 illustrates that our method achieves the best performance across all compute costs

D.5. Additional Results with ADM and VP Diffusion Models

We also experiment with pre-trained diffusion models in Dhariwal and Nichol [13] and Song et al. [55] on `ImageNet-64` and `FFHQ` respectively. The results are shown in Figure 12, where both NFE and effective NFE introduced in Section D.4 are considered. As can be observed, our method outperforms baselines across various diffusion models.



(a) FID against NFE.



(b) FID against effective NFE.

Figure 12. FID on `ImageNet-64` with ADM diffusion model [13] (left) and `FFHQ` with VP diffusion model [55] (right) when evaluated with Restart sampler.

	Before	After
$K = 2$	1.37	1.53
$K = 4$	1.11	1.31
$K = 6$	1.07	1.23

Table 9. FID on `ImageNet-64` when resampling is inserted before or after adding noise.

E. Additional Ablation Study

In this section, we present additional ablation studies that investigate three design choices in our particle filtering framework. Particularly, we will study the impacts of when the resampling is inserted (*i.e.,* before or after adding noise), the amount of noise added, and the NFE used to generate a single image. We will study their impacts in class-conditioned generation on `ImageNet-64` and when Restart sampler is used.

We start by studying **when is a proper time to insert resampling** during the generation. Particularly, we compare two time points to insert resampling. ❶ Resample before adding noise; ❷ Resample after adding noise. Table 9 shows the FID for both cases. It can be observed that resampling before adding noise significantly outperforms the counterpart. There could be two potential reasons for the performance difference. First, resampling after adding noise means resampling weights are estimated at higher noise scales, which is less accurate. Second, compared to

	Original noise	Reduced noise
$K = 2$	1.37	1.54
$K = 4$	1.11	1.37
$K = 6$	1.07	1.31

Table 10. FID on `ImageNet-64` with original and reduced noise.

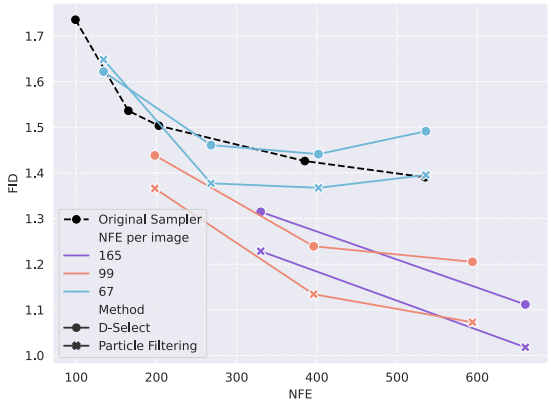


Figure 13. FID on `ImageNet-64` when D-SELECT and our PF method are applied to different NFEs per image.

❷ that adopts a selection-after-exploration procedure, ❶ selects particles before exploration, which allows more particles to be sampled around the promising particles with large weights.

Following the above intuition, we further explore the impact of the **amount of noise added**. Specifically, we change the restart configuration such that the variance of the added noise at time t is reduced to t^2, while keeping all other configurations the same. Table 10 shows the FID when the original and reduced noise is added. The results indicate the importance of a large added noise.

Finally, we explore the impact of the **NFE for each image**. Instead of fixing an NFE and only varying the number of particles, we now evaluate our method when it is applied to different NFEs. Figure 13 shows the FID when each image is generated with NFE $= 67, 99, 165$. For each NFE, we further report the performance when $K = 2, 4, 6$ images are generated (except for NFE $= 165$ due to the large cost). There are two observations from the figure. First, PF consistently outperforms D-SELECT across all NFEs, again demonstrating its advantages. Second, different NFEs present varying performance trends. When the number of denoising steps is small (NFE $= 67$), FID reaches a plateau at $K = 4$ particles, and further increasing the value of K does not improve performance due to the large discretization error. On the contrary, FID continues decreasing as K increases when the discretization error is small (NFE $= 99, 165$). Particularly, generating $K = 4$ particles when NFE $= 165$ achieves the state-of-the-art FID of 1.02 on `ImageNet-64`.

Figure 14. Categories of common failure cases of our method. Errors are highlighted in red.

F. Generated Samples

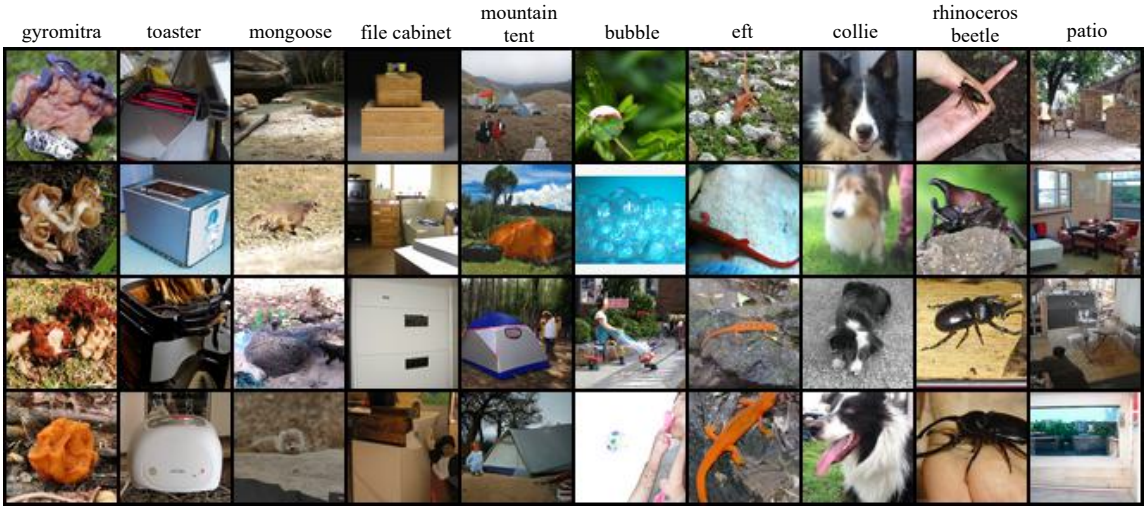
In this section, we present more generated samples for our method and baselines.

Figure 14 illustrates four types of failure cases of our method. In general, our method tends to miss the attributes that are not captured by the object detector, such as colors, locations, counts, sizes, *etc*, as shown by the four images, respectively. This is because our framework drops the correction term for object characteristics in Eq. (8). However, our method remains flexible to incorporate other desired measures into the particle weight calculation.

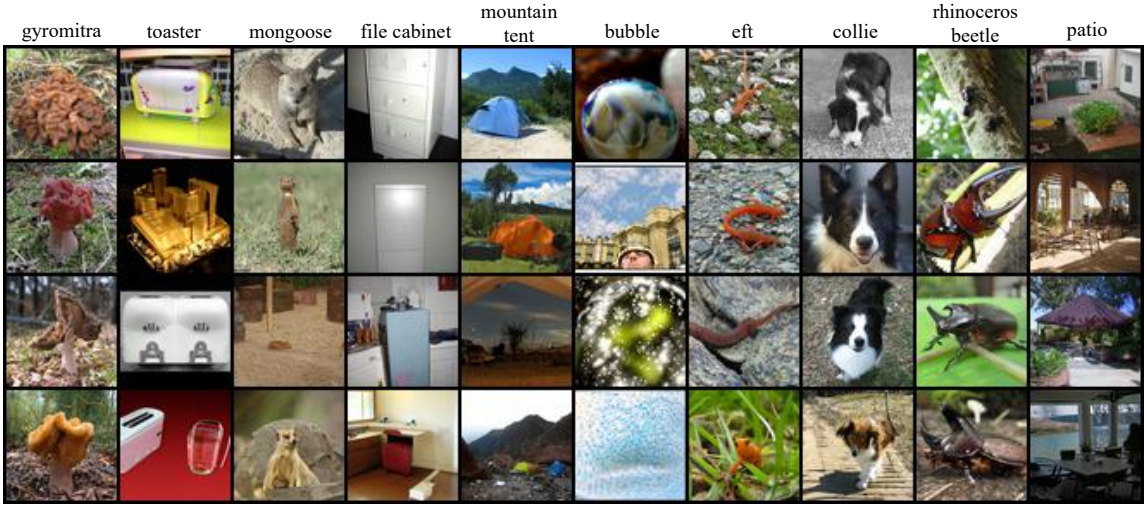
Figures 15 and 16 show the uncurated samples on `ImageNet-64` and `FFHQ` respectively. As can be observed, both D-GUIDANCE and PF generate images with higher quality than the original sampler. Compared to D-GUIDANCE, our method generates images with closer and more detailed views (*e.g.,* class "mongoose").

G. Details of Subjective Evaluation

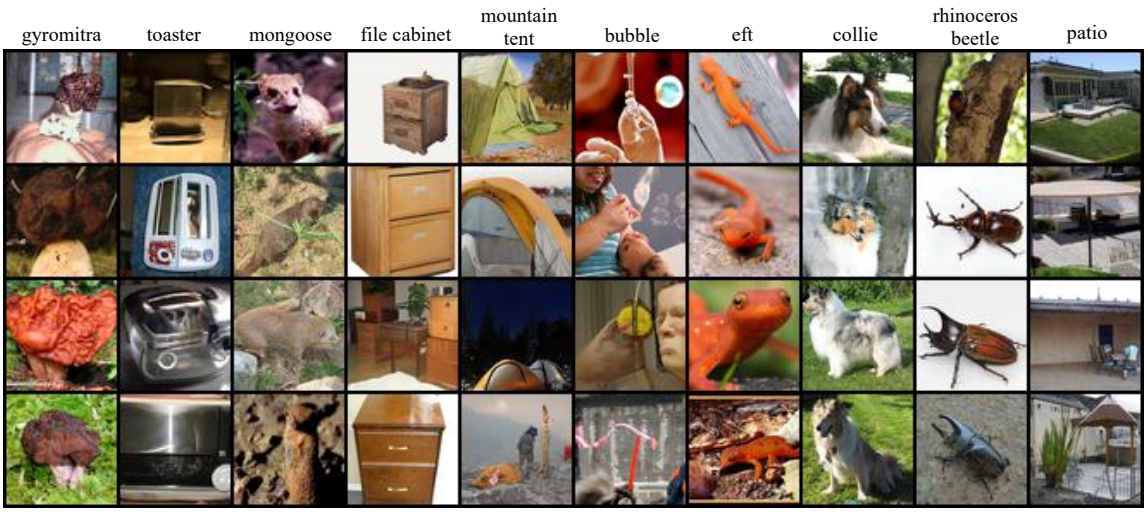
We recruit human annotators to evaluate both object occurrence and image quality of generated images. For object occurrence, annotators are provided with an image and a list of objects, and they are asked to identify whether each object occurs in the image or not. For image quality, annotators are given two images generated based on the same caption, where one image is generated by PF-HYBRID and the other image by a baseline. The caption is not revealed to annotators in order to make the evaluation focus more on image quality instead of image and text alignment. The annotation interfaces for object occurrence and image quality are shown in Tables 11 and 12 respectively.



Original sampler NFE: 535 FID: 1.39



D-guidance NFE: 535 FID: 1.16



Particle filtering NFE: 594 FID: **1.07**

Figure 15. Uncurated samples on `ImageNet-64` for our method and baselines. All methods use Restart sampler.

Original sampler NFE: 251 FID: 2.47

D-guidance NFE: 251 FID: 2.08

Particle filtering NFE: 268 FID: **2.02**

Figure 16. Uncurated samples on FFHQ for our method and baselines. All methods use Restart sampler.

Instructions:

You will be given an image generated by some AI algorithm, your task is to **identify all objects** that appear in the image.
Notes:

- You should **ignore the characteristics** of the object such as color and count when deciding whether it appears or not. E.g., if the object is "a red apple," you should still select it even if the image shows a green apple.
- Only select objects that you are **confident** about. *E.g.*, the "knife" in the image below is not clear, so you should not select it.

Example:

Do the following objects appear in the image?
• A black blender: Yes • bottle: No • glass: Yes • cutting board: Yes • knife: No

Task:

Please identify **all objects** appear in the image

☐ white kitchen sink ☐ white coffee cups ☐ wine glasses ☐ knives ☐ None of the above

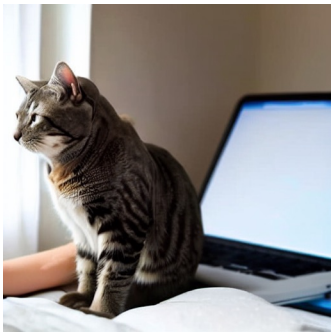
Table 11. Instructions and an example task for the subjective evaluation on object occurrence.

Instructions:

In this task, you will see two images that are generated based on a text caption by different AI algorithms. You will be asked to evaluate **which image looks more real and natural**.

Example:

(A)



(B)

Image (B) generates a natural cat and laptop, whereas the laptop in image (A) is distorted, and the human body is also unnatural. Therefore, **image (B) is better**.

Task:

Which image looks more natural and like a real photo?

(A)

(B)

☐ (A) is better ☐ (B) is better

Table 12. Instructions and an example task for the subjective evaluation on image quality.