**eDiff-I: Text-to-Image Diffusion Models with an Ensemble of Expert Denoisers**

Yogesh Balaji, Seungjun Nah, Xun Huang, Arash Vahdat, Jiaming Song, Qinsheng Zhang, Karsten Kreis, Miika Aittala, Timo Aila, Samuli Laine, Bryan Catanzaro, Tero Karras, Ming-Yu Liu

NVIDIA Corporation

{ybalaji, snah, xunh, avahdat, jiamings, qinshengz, kkreis, maittala, taila, slaine, bcatanzaro, tkarras, mingyul}@nvidia.com

Figure 1. Example results and capabilities from our proposed method, eDiff-I. The first row shows that eDiff-I can faithfully turn complex input text prompts into artistic and photorealistic images. In the second row, we first show that eDiff-I can combine the text input and a reference image for generating the target output image, where the reference image can be conveniently used to represent a style or concept that is difficult to describe in words, but a visual example exists. We also show the paint-by-word capability of eDiff-I, where phrases in the input text can be painted on a canvas to control the specific layout of objects described in the input text. The paint-with-words capability complements the text-to-image capability and provides an artist with more control over the generation outputs. More results are available at [https://deepimagination.cc/eDiff-I/](https://deepimagination.cc/eDiff-I/)
Abstract

Large-scale diffusion-based generative models have led to breakthroughs in text-conditioned high-resolution image synthesis, demonstrating complex text comprehension and outstanding zero-shot generalization. Starting from random noise, such text-to-image diffusion models gradually synthesize images in an iterative fashion while conditioning on text prompts. We find that their synthesis behavior qualitatively changes throughout this process: Early in sampling, generation strongly relies on the text prompt to generate text-aligned content, while later, the text conditioning is almost entirely ignored, and the task changes to producing outputs of high visual fidelity. This suggests that sharing model parameters throughout the entire generation process, the standard practice in the literature, may not be ideal to best capture these distinctly different modes of the generation process. Therefore, in contrast to existing works, we propose to train an ensemble of text-to-image diffusion models specialized for different synthesis stages. To maintain training efficiency, we initially train a single model, which is then progressively split into specialized models that are further trained for the specific stages of the iterative generation process. Our ensemble of diffusion models, called eDiff-I, results in improved text alignment while maintaining the same inference computation cost and preserving high visual quality, outperforming previous large-scale text-to-image diffusion models on the standard benchmark. In addition, we train our model to exploit a variety of embeddings for conditioning, including the T5 text, CLIP text, and CLIP image embeddings. We show that these different embeddings lead to different image formation behaviors. Notably, the CLIP image embedding allows an intuitive and instant way of transferring the style of a reference image to the target text-to-image output. Lastly, we show a technique that enables eDiff-I’s “paint-with-words” capability. A user can select the word in the input text and paint it in a canvas to control the output, which is very handy for crafting the desired image in mind. The project page is available at https://deepimagination.cc/eDiff-I/

1. Introduction

Diffusion models [26,69,73] that generate images through iterative denoising, as visualized in Figure 2, are revolutionizing the field of image generation. They are the core building block of recent text-to-image models, which have demonstrated astonishing capability in turning complex text prompts into photorealistic images, even for unseen novel concepts [57,59,63]. These models have led to the development of numerous interactive tools and creative applications and turbocharged the democratization of content creation.

Arguably, this success is attributed largely to the great scalability of diffusion models because the scalability provides a clear pathway for practitioners to translate larger model capacity, compute, and datasets into better image generation quality. This is a great reminder of the bitter lessons [75], which observe that a scalable model trained on vast data with massive computing power in the long term often outperforms its handcrafted specialized counterparts. A similar trend has been observed previously in natural language modeling [33], image classification [23], image reconstruction [38], and autoregressive generative modeling [21].

We are interested in further scaling diffusion models in terms of model capability for the text-to-image generation task. We first note that simply increasing the capacity by using deeper or wider neural networks for each denoising step will negatively impact the test-time computational complexity of sampling, since sampling amounts to solving a reverse (generative) differential equation in which a denoising network is called many times. We aim to achieve the scaling goal without incurring the test-time computational complexity overhead.

Our key insight is that text-to-image diffusion models exhibit an intriguing temporal dynamic during generation. At the early sampling stage, when the input data to the denoising network is closer to the random noise, the diffusion model mainly relies on the text prompt to guide the sampling process. As the generation continues, the model gradually shifts towards visual features to denoise images, mostly ignoring the input text prompt as shown in Figure 3 and 4.
Motivated by this observation, we propose to increase the capacity of diffusion models by training an ensemble of **expert denoisers**, each specialized for a particular stage in the generation process. While this does not increase the computational complexity of sampling per time step, it increases the training complexity since different denoising models should be trained for different stages. To remedy this, we propose pre-training a shared diffusion model for all stages. We then use this pre-trained model to initialize specialized models and finetune them for a smaller number of iterations. This scheme leads to state-of-the-art text-to-image generation results on the benchmark dataset.

We also explore using an ensemble of pretrained text encoders to provide inputs to our text-to-image model. We use both the CLIP text encoder [55], which is trained to align text embedding to the corresponding image embedding, and the T5 text encoder [56], which is trained for the language modeling task. Although prior works [51, 57, 59, 60, 63] have used these two encoders, they have not been used together in one model. As these two encoders are trained with different
The result is an image generation model that can take both noise levels and thus are also known as noise-conditioned denoising score matching [31, 79] objectives at different iterative denoising process. These models are trained with deep generative models that generate samples through an ensemble diffusion for images, abbreviated eDiff-I.

While text prompts are effective in specifying the objects to be included in the generated images, it is cumbersome to use text to control the spatial locations of objects. We devise a training-free extension of our model to allow paint-with-words, a controllable generation approach with our text-to-image model that allows the user to specify the locations of specific objects and concepts by scribbling them in a canvas. The result is an image generation model that can take both words and semantic masks as inputs to better assist users in crafting the perfect images in their minds.

Contributions The main contributions of our work are

1. Based on the observation that a text-to-image diffusion model has different behaviors at different noise levels, we propose the ensemble-of-expert-denoisers design to boost generation quality while maintaining the same inference computation cost. The expert denoisers are trained through a carefully designed finetuning scheme to reduce the training cost.

2. We propose to use an ensemble of encoders to provide input information to the diffusion model. They include the T5 text encoder, the CLIP text encoder, and the CLIP image encoder. We show that the text encoders favor different image formations, and the CLIP image encoder provides a useful style transfer capability that allows a user to use a style reference photo to influence the text-to-image output.

3. We devise a training-free extension that enables the paint-with-words capability through a cross-attention modulation scheme, which allows users additional spatial control over the text-to-image output.

2. Related Work

Denoising Diffusion models [26, 69, 73] are a class of deep generative models that generate samples through an iterative denoising process. These models are trained with denoising score matching [31, 79] objectives at different noise levels and thus are also known as noise-conditioned score networks [71, 72]. They have driven successful applications such as text-to-image generation [57, 59, 63], natural language generation [41], time series prediction [76], audio synthesis [39], 3D shape generation [49, 87, 91], molecular conformation generation [84], protein structure generation [83], machine learning security [52], and differentially private image synthesis [14].

Text-to-image diffusion models Some of the most high-quality text-to-image generative models are based on diffusion models. These models learn to perform the denoising task conditioned on text prompts, either on the image space (such as GLIDE [51] and Imagen [63]) or on a separate latent space (such as DALL-E 2 [57], Stable Diffusion [59, 60], and VQ-Diffusion [19]). For computational efficiency, a diffusion model is often trained on low-resolution images or latent variables, which are then transformed into high-resolution images by super-resolution diffusion models [27] or latent-to-image decoders [68, 77]. Samples are drawn from these diffusion models using classifier-free guidance [13, 28] as well as various sampling algorithms that use deterministic [15, 34, 43, 46, 70, 89] or stochastic [4, 5, 16, 90] iterative updates. Several works retrieve auxiliary images related to the text prompt from an external database and condition generation on them to boost performance [7, 9, 66]. Recently, several text-to-video diffusion models were proposed and achieved high-quality video generation results [20, 25, 29, 67, 85].

Applications of text-to-image diffusion models Apart from serving as a backbone to be fine-tuned for general image-to-image translation tasks [80], text-to-image diffusion models have also demonstrated impressive capabilities in other downstream applications. Diffusion models can be directly applied to various inverse problems, such as super-resolution [13, 64], inpainting [11, 47], deblurring [35, 82], and JPEG restoration [36, 62]. For example, blended diffusion [2, 3] performs inpainting with natural language descriptions. Text-to-image diffusion models can also perform other semantic image editing tasks. SDEdit [50] enables re-synthesis, compositing, and editing of an existing image via colored strokes or image patches. DreamBooth [61] and Textual Inversion [18] allow the “personalization” of models by learning a subject-specific token from a few images. Prompt-to-prompt tuning can achieve image editing by modifying the textual prompt used to produce the same image without having the user provide object-specific segmentation masks [22]. Similar image-editing capabilities can also be achieved by fine-tuning the model parameters [37, 78] or automatically finding editing masks with the denoiser [12].

Scaling up deep learning models The recent success of deep learning has primarily been driven by increasingly large
models and datasets. It has been shown that simply scaling up model parameters and data size results in substantial performance improvement in various tasks, such as language understanding [8, 10], visual recognition [88], and multimodal reasoning [55]. However, those high-capacity models also incur increased computational and energy costs during training and inference. Some recent works [1, 58, 65] employ sparse expert models which route each input example to a small subset of network weights, thereby keeping the amount of computation tractable as scaling. Similarly, our proposed expert denoisers increase the number of trainable parameters without adding the computational cost at test time.

3. Background

In text-to-image generative models, the input text is often represented by a text embedding, extracted from a pre-trained model such as CLIP [55] or T5 [56] text encoders. In this case, the problem of generating images given text prompts simply boils down to learning a conditional generative model that takes text embeddings as input conditioning and generates images aligned with the conditioning.

Text-to-image diffusion models generate data by sampling an image from a noise distribution and iteratively denoising it using a denoising model \(D(x; e, \sigma)\) where \(x\) represents the noisy image at the current step, \(e\) is the input embedding, and \(\sigma\) is a scalar input indicating the current noise level. Next, we formally discuss how the denoising model is trained and used for sampling.

**Training** The denoising model is trained to recover clean images given their corrupted versions, generated by adding Gaussian noise of varying scales. Following the EDM formulation of Karras et al. [34] and their proposed corruption schedule [34, 70], we can write the training objective as:

\[
\mathbb{E}_{P_{\text{data}}(x, \sigma), \sigma; \epsilon}) \cdot p(\epsilon) \cdot p(\lambda(\sigma)) \cdot D(x_{\text{clean}} + \sigma; e, \sigma) - x_{\text{clean}} \parallel^2)
\]

where \(p_{\text{data}}(x_{\text{clean}}, e)\) represents the training data distribution that produces training image-text pairs, \(p(\epsilon) = \mathcal{N}(0, 1)\) is the standard Normal distribution, \(p(\sigma)\) is the distribution in which noise levels are sampled from, and \(\lambda(\sigma)\) is the loss weighting factor.

**Denoiser formulation** Following Karras et al. [34], we precondition the denoiser using:

\[
D(x; e, \sigma) := \left(\frac{\sigma_{\text{data}}}{\sigma^*}\right)^2 \cdot x + \frac{\sigma \cdot \sigma_{\text{data}}}{\sigma^*} \cdot F_{\theta}\left(\frac{x}{\sigma^*}; e, \frac{\ln(\sigma)}{4}\right)
\]

where \(\sigma^* = \sqrt{\sigma^2 + \sigma^2_{\text{data}}}\) and \(F_{\theta}\) is the trained neural network. We use \(\sigma_{\text{data}} = 0.5\) as an approximation for the standard deviation of pixel values in natural images. For \(\sigma\), we use the log-normal distribution \(\ln(\sigma) \sim \mathcal{N}(P_{\text{mean}}, P_{\text{std}})\) with \(P_{\text{mean}} = -1.2\) and \(P_{\text{std}} = 1.2\), and weighting factor \(\lambda(\sigma) = (\sigma^*/(\sigma \cdot \sigma_{\text{data}}))^2\) that cancels the output weighting of \(F_{\theta}\) in (1).

**Sampling** To generate an image with the diffusion models, an initial image is generated by sampling from the prior distribution \(x \sim \mathcal{N}(0, \sigma^2_{\text{max}})\), and then the generative ordinary differential equation (ODE) is solved using:

\[
\frac{dx}{ds} = -\sigma \cdot \nabla_x \log p(x|e, \sigma) = \frac{x - D(x; e, \sigma)}{\sigma}
\]

for \(\sigma\) flowing backward from \(\sigma_{\text{max}}\) to \(\sigma_{\text{min}} \approx 0\). Above, \(\nabla_x \log p(x|e, \sigma)\) represents the score function of the corrupted data at noise level \(\sigma\) which is obtained from the denoising model [31, 79]. Above \(\sigma_{\text{max}}\) represents a high noise level at which all the data is completely corrupted, and the mutual information between the input image distribution and the corrupted image distribution is approaching zero. Note that sampling can also be expressed as solving a stochastic differential equation as discussed in Song et al. [73].

**Super-resolution diffusion models** The training of text-conditioned super-resolution diffusion models largely follows the training of text-conditioned diffusion models described above. The major difference is that the super-resolution denoising model also takes the low-resolution image as a conditioning input. Following prior work [57], we apply various corruptions to the low-resolution input image during training [81] to enhance the generalization capability of the super-resolution model.

4. Ensemble of Expert Denoisers

As we discussed in the previous section, text-to-image diffusion models rely on a denoising model to convert samples from a prior Gaussian distribution to images conditioned on an input text prompt. Formally, the generative ODE shown in (2) uses \(D(x; e, \sigma)\) to guide the samples gradually towards images that are aligned with the input conditioning.

The denoising model \(D\) at each noise level \(\sigma\) relies on two sources of information for denoising: the current noisy input image \(x\) and the input text prompt \(e\). Our key observation is that text-to-image diffusion models exhibit a unique temporal dynamic while relying on these two sources. At the beginning of the generation, when \(\sigma\) is large, the input image \(x\) contains mostly noise. Hence, denoising directly from the input visual content is a challenging and ambiguous task. At this stage, \(D\) mostly relies on the input text embedding to infer the direction toward text-aligned images. However, as \(\sigma\) becomes small towards the end of the generation, most coarse-level content is painted by the denoising model. At this stage, \(D\) mostly ignores the text embedding and uses visual features for adding fine-grained details.

We validate this observation in Figure 3 by visualizing the cross-attention maps between visual and text features.
We begin by training a model shared among all noise levels (Sec. 3). Recently, Luhman
To remedy this, we first train a shared model across all noise

general, at level as described above and train experts for each sub-interval.

of their corresponding noise intervals in a similar fashion

p

expert trained on

trained on

and

trained on the noise distributions

using the full noise level distribution denoted as

p

implementation for training the expert denoisers efficiently.

4.1. Efficient Training of Expert Denoisers

We propose a branching strategy based on a binary tree
treatment for training the expert denoisers efficiently. We begin by training a model shared among all noise levels
using the full noise level distribution denoted as p(σ). Then,
we initialize two experts from this baseline model. Let us
call these models the level 1 experts since they are trained
on the first level of the binary tree. These two experts are
trained on the noise distributions pDL1 (σ) and pUL1 (σ), which
are obtained by splitting p(σ) equally by area. So, the expert
trained on pDL1 (σ) specializes in low noise levels, while the
expert trained on pUL1 (σ) specializes in high noise levels. In
our implementation, p(σ) follows a log-normal distribution
(Sec. 3). Recently, Luhman et al. [48] have also trained two
diffusion models for two-stage denoising for image generation,
but their models are trained over images of different
resolutions and separately from the beginning.

Once the level 1 expert models are trained, we split each
of their corresponding noise intervals in a similar fashion
as described above and train experts for each sub-interval.
This process is repeated recursively for multiple levels. In
general, at level l, we split the noise distribution p(σ) into 2l

intervals of equal area given by \{pDLi (σ)\}i=02l−1, with model i
being trained on the distribution pDLi (σ). We call such a model
or node in the binary tree Ei.

Ideally, at each level l, we would have to train 2l models.
However, this is impractical as the model size grows exponen-
tially with the depth of the binary tree. Also, in practice,
we found that models trained at many of the intermediate
intervals do not contribute much toward the performance of
the final system. Therefore, we focus mainly on growing the
tree from the left-most and the right-most nodes at each level
of the binary tree: E0 and E2l−1. The right-most interval
contains samples at high noise levels. As shown in Figures 3
and 4, good denoising at high noise levels is critical for im-
proving text conditioning as core image formation occurs in
this regime. Hence, having a dedicated model in this regime
is desired. Similarly, we also focus on training the models at
lower noise levels as the final steps of denoising happen in
this regime during sampling. So, good models are needed
to get sharp results. Finally, we train a single model on all
the intermediate noise intervals that are between the two
extreme intervals.

In a nutshell, our final system would have an ensemble
of three expert denoisers: an expert denoiser focusing on
the low noise levels (given by the leftmost interval in the

Figure 5. eDiff-I consists of a base diffusion model that generates
images in 64×64 resolution. This is followed by two super-
resolution diffusion models that upsample the images to 256×256
and 1024×1024 resolution, respectively, referred to as SR256 and
SR1024 through the paper. All models are conditioned on text
through both T5 and CLIP text embeddings. eDiff-I also allows
the user to optionally provide an additional CLIP image embed-
ing. This can enable detailed stylistic control over the output (see
Figures 1 and 16).
binary tree), an expert denoiser focusing on high noise levels (given by the right-most interval in the binary tree), and a single expert denoiser for learning all intermediate noise intervals. A more detailed description of our branching strategy is described in Appendix A. In Sec. 5, we also consider other types of ensemble experts for quantitative evaluation purposes.

4.2. Multiple Conditional Inputs

To train our text-to-image diffusion models, we use the following conditional embeddings during training: (1) T5-XXL [56] text embeddings, (2) CLIP L/14 text embeddings and (3) CLIP L/14 image embeddings. We pre-compute these embeddings for the whole dataset since computing them online is very expensive. Similar to prior work [27, 59, 63], we add the projected conditional embeddings to the time embedding and additionally perform cross attention at multiple resolutions of the denoising model. We use random dropout [74] on each of these embeddings independently during training. When an embedding is dropped, we zero out the whole embedding tensor. When all three embeddings are dropped, it corresponds to unconditional training, which is useful for performing classifier-free guidance [28]. We visualize the input conditioning scheme in Figure 5.

Our complete pipeline consists of a cascade of diffusion models. Specifically, we have a base model that can generate images of $64 \times 64$ resolution and two super-resolution diffusion models that can progressively upsample images to $256 \times 256$ and $1024 \times 1024$ resolutions, respectively (see Figure 5). To train the super-resolution models, we condition on the ground-truth low-resolution inputs that are corrupted by random degradation [81]. Adding degradation during training allows the models to better generalize to remove artifacts that can exist in the outputs generated by our base model. For the base model, we use a modified version of the U-net architecture proposed in Dhariwal et al. [13], while for super-resolution models, we use a modified version of the Efficient U-net architecture proposed in Saharia et al. [63]. More details on the architectures can be found in Appendix A.

4.3. Paint-with-words

We propose a training-free method, named paint-with-words, that enables users to specify the spatial locations of objects. As shown in Figure 6, users can select an arbitrary phrase from the text prompt and doodle on a canvas to create a binary mask corresponding to that phrase. The masks are input to all cross-attention layers and are bilinearly downsampled to match the resolution of each layer. We use those masks to create an input attention matrix $A \in \mathbb{R}^{N_t \times N_i}$, where $N_t$ and $N_i$ are the number of image and text tokens, respectively. Each column in $A$ is generated by flattening the mask corresponding to the phrase that contains the text token of that column. The column is set to zero if the corresponding text token is not contained in any phrases selected by the user. We add the input attention matrix to the original attention matrix in the cross-attention layer, which now computes the output as $\text{softmax}\left(\frac{QK^T + wA}{\sqrt{d_k}}\right)V$, where $Q$ is query embeddings from image tokens, $K$ and $V$ are key and value embeddings from text tokens, $d_k$ is the dimensionality of $Q$ and $K$, and $w$ is a scalar weight that controls the strength of user input attention. Intuitively, when the user paints a phrase on a region, image tokens in that region are encouraged to attend more to the text tokens contained in that phrase. As a result, the semantic concept corresponding to that phrase is more likely to appear in the specified area.

We find it beneficial to use a larger weight at higher noise levels and to make the influence of $A$ irrelevant to the scale of $Q$ and $K$, which corresponds to a schedule that works well empirically:

$$w = w' \cdot \log(1 + \sigma) \cdot \max(QK^T),$$

where $w'$ is a scalar specified by the user.

5. Experiments

First, we discuss optimization, dataset, and evaluation. We then show improved quantitative results compared to previous methods in Sec. 5.1. Then, we perform two sets of ablation studies as well as study the effect of increasing the number of experts over standard image quality metrics. In Sec 5.2, we evaluate image generation performance based on CLIP and/or T5 text embeddings, where we show that having both embeddings lead to the best image generation quality. Finally, in Sec 5.3, we illustrate style transfer applications that are enabled by eDiff-I. In Sec 5.3, we illustrate style transfer applications that are enabled by CLIP image embeddings, and in Sec 5.4, we present image generation results with the introduced paint-with-words method.
Figure 7. 2-Expert-Ensemble vs Baseline. We compare the FID-CLIP score trade-off curves between our 2-Expert-Ensemble model with our baseline model trained without any denoising expert. We sample 5K captions from each benchmark dataset. To make a fair comparison, we compare the baseline model trained for 800K iterations with our 2-Expert-Ensemble model trained for 600K iterations, as both models see the same number of training samples at this point. We show the results on the COCO dataset on the left panel and the results on the visual genome dataset on the right panel. In general, having a lower FID score and a higher CLIP score is more desirable. We observe that our ensemble model consistently outperforms the baseline model on the entire FID-CLIP score trade-off curve.

Figure 8. Effect of conditional embeddings. We plot the FID-CLIP score trade-off curves on the samples generated by our approach using (a) T5 + CLIP encoders, (b) T5 encoder only, and (c) CLIP text encoder only. On the COCO dataset, using T5 and CLIP encoders in isolation gives a similar performance, while using CLIP+T5 gives the best results. On the visual genome dataset, using the T5 encoder in isolation gives better performance than using the CLIP text encoder. This is because the visual genome dataset contains longer and more descriptive captions than the COCO dataset, and the use of a more powerful language model such as T5 becomes more beneficial. Nevertheless, using T5+CLIP performs the best even in this dataset with our model.

Optimization Both the base and super-resolution diffusion models are trained using the AdamW [45] optimizer with a learning rate of 0.0001, weight decay of 0.01, and a batch size of 2048. The base model was trained using 256 NVIDIA A100 GPUs, while the two super-resolution models were trained with 128 NVIDIA A100 GPUs each. Our implementation is based on the Imaginaire library [44] written in PyTorch [54]. More details on other hyperparameters settings are presented in Appendix B.

Datasets We use a collection of public and proprietary datasets to train our model. To ensure high-quality training data, we apply heavy filtering using a pretrained CLIP model to measure the image-text alignment score as well as an aesthetic scorer to rank the image quality. We remove image-text pairs that fail to meet a preset CLIP score threshold and a preset aesthetic score. The final dataset to train our model contains about one billion text-image pairs. All the images have the shortest side greater than 64 pixels. We use all of them to train our base model. We only use images with the shortest side greater than 256 and 1024 pixels to train our SR256 and SR1024 models, respectively. For training our base and SR256 models, we perform resize-central crop. Images are first resized so that the shortest side has the same number of pixels as the input image side. For training the SR1024 model, we randomly crop $256 \times 256$ regions during training and apply it on $1024 \times 1024$ resolution during inference. We use COCO and Visual Genome datasets for evaluation, which are excluded from our training datasets for measuring zero-shot text-to-image generation performance.
An origami of a monkey dressed as a monk riding a bike on a mountain.

A 4k dslr photo of two teddy bears wearing a sports jersey with the text “eDiffi” written on it. They are on a soccer field.

Figure 9. 2-Expert-Ensemble vs Baseline. We compare the image synthesis capability of our model (two rightmost panels) with our baseline (two left panels). We find that the baseline model does not generate mountains in one of the examples on the top row and produces incorrect text in the bottom row. Our 2-expert-ensemble improves the generation capability in both these cases.

Evaluation We use MS-COCO [42] dataset for most of the evaluation. Consistent with prior work [57, 63], we report zero-shot FID-30K [24] in which 30K captions are drawn randomly from the COCO validation set. We use the captions as inputs for synthesizing images. We compute the FID between these generated samples and the reference 30K ground truth images. We also report the CLIP score, which measures the average similarity between the generated samples and the corresponding input captions using features extracted from a pre-trained CLIP model. In addition to the MS-COCO validation set, we also report the CLIP and FID scores on Visual Genome dataset [40], which is a challenging dataset containing images and paired long captions.

5.1. Main Results

First, we study the effectiveness of our ensemble model by plotting the FID-CLIP score trade-off curve and comparing it with our baseline model, which does not use our proposed ensemble-of-expert-denoisers scheme but shares all the remaining design choices. The trade-off curves are generated by performing a sweep over classifier-free guidance values in the range \( \{0, 0.5, 1.0, \ldots, 9.5, 10.0\} \). In this experiment, we train an ensemble of four expert models by splitting the baseline model trained to 500K iterations and training each child model for 50K iterations. We further split each of these child models at 550K iterations and train the resulting four models for another 50K steps. This results in four expert models, \( E_2^1, E_2^2, E_2^3, \) and \( E_2^4 \), trained for 600K steps. To make a fair comparison, we evaluate the ensemble model against our baseline model trained for 800K steps, as this would correspond to both models seeing the same number of training samples. As shown in Figure 7, our ensemble model outperforms the baseline model by a significant margin on the entire trade-off curve.

Table 1. Zero-shot FID comparison with recent state-of-the-art methods on the COCO 2014 validation dataset. We include the text encoder size in our model parameter size calculation.

| Model              | # of params | Zero-shot FID ↓ |
|--------------------|-------------|-----------------|
| GLIDE [51]         | 5B          | 12.24           |
| Make-A-Scene [17]  | 4B          | 11.84           |
| DALL·E 2 [57]      | 6.5B        | 10.39           |
| Stable Diffusion [59]| 1.4B       | 8.59            |
| Imagen [63]        | 7.9B        | 7.27            |
| Parti [86]         | 20B         | 7.23            |
| eDiff-I-Config-A   | 6.8B        | 7.35            |
| eDiff-I-Config-B   | 7.1B        | 7.26            |
| eDiff-I-Config-C   | 8.1B        | 7.11            |
| eDiff-I-Config-D   | 9.1B        | **6.95**        |
Next, we report the FID-30K results of eDiff-I computed on 256×256 resolution images on the MS-COCO dataset and compared it with the state-of-the-art methods in Table 1. We experiment with the following model settings:

- **eDiff-I-Config-A** Our baseline model for the base model generates images at 64×64 resolution. The outputs are upsampled with our baseline SR256 model.
- **eDiff-I-Config-B** The same base model as in eDiff-I-Config-A. The ensemble SR256 model consists of two experts: $E^1$ and $E^2$.
- **eDiff-I-Config-C** Our 2-expert ensemble base model generates images at 64×64 resolution. This ensemble model consists of an expert model trained at leaf nodes $E^9_{511}$ and a complement model trained on all other noise levels except $E^9_{511}$. The outputs are upsampled with our ensemble SR256 model as in eDiff-I-Config-B.
- **eDiff-I-Config-D** Our 3-expert ensemble base model generates images at 64×64 resolution. The 3-expert ensemble model consists of $E^9_{511}$ model (high noise regime model), $E^9_{3}$ (low noise regime model), and an expert denoiser model covering the noise levels in-between. The outputs are upsampled with our ensemble SR256 model as in eDiff-I-Config-B.

We observe that our eDiff-I-Config-A model outperforms GLIDE [51], DALL-E 2 [57], Make-a-Scene [17], and Stable Diffusion [59], and achieves FID slightly higher than that of Imagen [63] and Parti [86]. By applying the proposed ensemble scheme to the SR256 model (eDiff-I-Config-B), we achieve an FID score of 7.26, which is slightly better than Imagen. As we apply the proposed 2-expert ensemble scheme to build eDiff-I-Config-C, which has roughly the same model size as Imagen, we outperform both Imagen and Parti by an FID score of 0.16 and 0.12, respectively. Our eDiff-I-Config-D model achieves the best FID of 7.04. In Figure 9, we qualitatively compare the results of eDiff-I-Config-A with those of eDiff-I-Config-C. We observe that our ensemble of expert denoisers generates improved results compared with the baseline.

We now report qualitative comparison results using our best eDiff-I configuration with two publicly available text-to-image generative models—Stable Diffusion [59] and DALL-E 2 [57] in Figures 10, 11, and 12. In the presence of multiple entities (Figure 10), Stable Diffusion and DALL-E 2 tend to mix the attributes from different entities or ignore some of the attributes, while eDiff-I can accurately model attributes from all entities. In generating texts (Figure 11), both Stable Diffusion and DALL-E 2 often produce misspellings or ignore words, while eDiff-I correctly generates the texts. Even in the case of long descriptions, eDiff-I can handle long-range dependencies better and perform better than DALL-E 2 and Stable Diffusion.

In Figure 14, we show that eDiff-I can generate images with a variety of styles by using the proper text prompts. Our model can also generate many variations for a given text prompt, as shown in Figure 15.

In Figure 13, we conduct an experiment to illustrate that the proposed ensemble-of-expert-denoisers scheme helps scale the model size without incurring additional computation in the inference time.

### 5.2. CLIP Text and T5 Text

As explained in Sec. 4.2, we use both CLIP text embeddings and T5 text embeddings to train our models. Since we perform random dropout independently on the individual embeddings during training, the model has the capability to generate images when each of the embeddings is used in isolation. In Figure 18, we examine the effect of the individual text embeddings in our model. We observe that images generated using the CLIP text embeddings alone typically have the correct foreground object, but lack in terms of compositionality, counting, and generating text. On the other hand, images generated by using only the T5 text embeddings obtain better compositions but are inferior in generating the foreground objects, such as the breeds of dogs. Using both T5 and CLIP text embeddings, we get the best of both worlds, where our model can use the provided attributes from each of the text embeddings.

Next, we quantitatively evaluate the effect of individual embeddings by plotting the CLIP-FID trade-off curve on MS-COCO and Visual Genome datasets in Figure 8. We observe that, on the MS-COCO dataset, using CLIP and T5 embeddings in isolation results in a similar performance, while using CLIP+T5 embeddings leads to much better trade-off curves. On the visual genome dataset, using T5 embeddings in isolation leads to better performance than using CLIP text embeddings. A closer look at the dataset statistics reveals that the average number of words in each caption of the MS-COCO dataset is 10.62, while it is 61.92 for Visual Genome. So, when the text is more descriptive, the use of T5 embeddings performs better than the CLIP text embeddings. Again, the best performance is obtained by using CLIP+T5 embeddings.

### 5.3. Style transfer

In addition to the T5 and CLIP text embeddings, our model is also conditioned on CLIP image embeddings during training. We find that the use of CLIP image embeddings gives us the ability to do style transfer during synthesis. In Figure 16, we show some of our style transfer results. From a given reference image, we first obtain its CLIP image embedding. We then sample outputs conditioned on both
A photo of two cute teddy bears sitting on top of a grizzly bear in a beautiful forest. Highly detailed fantasy art, 4k, artstation.

There are two Chinese teapots on a table. One pot has a painting of a dragon, while the other pot has a painting of a panda.

A photo of two squirrel warriors dressed as knights fighting on a battlefield. The squirrel on the left holds a stick, while the squirrel on the right holds a long sword. Gray clouds.

A photo of a dog wearing a blue shirt and a cat wearing a red shirt sitting in a park, photorealistic dslr.

Figure 10. Comparison between samples generated by our approach with DALL-E 2 and Stable Diffusion. Category: multiple entities. In this set of results, we generate images with multiple entities. Stable Diffusion and DALL-E 2 tend to mix the attributes corresponding to different entities. For instance, in the last row, the dog and the cat both wear red shirts in DALL-E 2 outputs, while Stable Diffusion does not even generate a cat. Our model, on the other hand, produces all entities consistent with the captions. Note that we generated multiple outputs from each method and picked the best one to include in the figure.
A photo of two monkeys sitting on a tree. They are holding a wooden board that says “Best friends”, 4K dslr.

A photo of a golden retriever puppy wearing a green shirt. The shirt has text that says “NVIDIA rocks”. Background office. 4K dslr.

An ice sculpture is made with the text “Happy Holidays”. Christmas decorations in the background. Dslr photo.

A dslr photo of a colorful Graffiti on a wall with the text “Peace love”. There is a painting of a bulldog with sunglasses next to it.

Figure 11. Comparison between samples generated by our approach with DALL-E 2 and Stable Diffusion. Category: text. We observe that both DALL-E 2 and Stable Diffusion fail at generating text. These models either produce misspellings or do not even generate text. For example, as spelling “NVIDIA rocks”, Stable Diffusion failed to generate any text, while DALL-E 2 only generated “NIDCKA VIDA”. Only our model succeeds in generating the target. Overall, we observe that our model can generate English text on a wide range of samples. Note that we generated multiple outputs from each method and picked the best one to include in the figure.
Figure 12. Comparison between samples generated by our approach with DALL-E 2 and Stable Diffusion. Category: long detailed captions. On long descriptions, Stable Diffusion and DALL-E 2 sometimes fail to create images with all attributes mentioned in the caption. For instance, in the last example, neither method generates sushi on top of pasta. Our method can handle long detailed descriptions better than other methods. Note that we generated multiple outputs from each method and picked the best one to include in the figure.
We use the same base model architecture but increase the number of residual blocks there. We perform feed-forward evaluations 100 times on an NVIDIA A100 GPU and report the average feed-forward time in the curve. As expected, the feed-forward evaluation time increases with the network capacity. The second scheme is our proposed scheme, where we simply use different numbers of denoising experts for different noise levels, and hence the per-network feed-forward evaluation time remains constant.

Figure 13. Model inference time by base model capacity. In this figure, we compare the network feed-forward evaluation time for two schemes of increasing network capacity in diffusion models. The first scheme is through increasing the depth of the model. We use the same base model architecture but increase the number of residual blocks there. We perform feed-forward evaluations 100 times on an NVIDIA A100 GPU and report the average feed-forward time in the curve. As expected, the feed-forward evaluation time increases with the network capacity. The second scheme is our proposed scheme, where we simply use different numbers of denoising experts for different noise levels, and hence the per-network feed-forward evaluation time remains constant.

5.4. Paint-with-words

We show some results produced by our “paint-with-words” approach in Fig. 17. Although the doodles are very coarse and do not contain the exact shape of objects, our method is still able to synthesize high-quality images that have the same rough layout. In most scenarios, this is more convenient than segmentation-to-image methods [30,32,53], which are likely to fail when user-drawn shapes are different from shapes of real objects. Compared with text-conditioned inpainting methods [3,6,51] that apply a single concept to an image region, “paint-with-words” can generate the whole image containing multiple concepts in a single pass from scratch, without the need to start from an input image.

Societal impact We hope that eDiff-I can serve as a powerful tool for digital artists for content creation and to express their creativity freely. Modern text-to-image diffusion models like ours have the potential to democratize artistic expression by offering the user the ability to produce detailed and high-quality imagery without the need for specialized skills. We envision that eDiff-I can benefit designers, photographers, and content creators.

However, state-of-the-art text-to-image generative models like eDiff-I need to be applied with an abundance of caution. For instance, they can also be used for advanced photo manipulation for malicious purposes or to create deceptive or harmful content. In fact, the recent progress of generative models and AI-driven image editing has profound implications for image authenticity and beyond. Such challenges can potentially be tackled, for instance, by methods that automatically validate real images and detect manipulated or fake content. Moreover, the extremely large, mostly unfiltered training data sets of current large-scale text-to-image generative models include biases that are captured by the model and also reflected in the generated data. It is, therefore, important to be aware of such biases in the underlying data and counteract them, for example, by actively collecting more representative data or by using bias correction methods.

Acknowledgements We would like to thank Qingsheng Zhang, Robin Rombach, Chen-Hsuan Lin, Mohammad Shoeybi, Tsung-Yi Lin, Wei Ping, Mostofa Patwary, Andrew Tao, Guilin Liu, Vijay Korthikanti, Sanja Fidler, David Luethke, and Jan Kautz for useful discussions. We would also like to thank Margaret Albrecht and Greg Estes for helping iterate our presentation. Thanks also go to Gabriele Leone and Ashlee Martino-Tarr, who are our valuable early testers of eDiff-I. Finally, we would like to thank Amanda Moran, John Dickinson, and Sivakumar Arayandi Thottakara for the computing infrastructure support.
Figure 14. Samples generated by our approach for the caption “A \{X\} photo/painting of a penguin working as a fruit vendor in a tropical village”, where X denotes one of the nine styles listed above. Our model can generate images from a wide range of artistic styles while being consistent with the input text description.
A photo of a dog wearing a blue shirt and a cat wearing a red shirt sitting in a park, photorealistic dslr.

A photo of a hedgehog sitting in a small boat in the middle of a pond. It is wearing a Hawaiian shirt and a straw hat. It is reading a book.

An ice sculpture is made with text “Happy Holidays”. Christmas decorations in the background. Dslr photo.

An oil painting of raccoons dressed as three musketeers in an old French town.

A photo of a lemur wearing a red magician hat and a blue coat in a garden. The lemur is performing a magic trick with cards.

Figure 15. Image variations generated by our method. For each text prompt, we show four random sample variations generated by our method. We find that our model can generate diverse samples for each prompt. In the first row, our model can generate different breeds of dogs and cats, whereas, in the third row, we see that our model can generate the text “Happy Holidays” with different styles.
A photo of two pandas walking on a road.

A detailed oil painting of a beautiful rabbit queen wearing a royal gown in a palace. She is looking outside the window, artistic.

A DSLR photo of a dog playing trumpet from the top of a mountain.

A photo of a teddy bear wearing a casual plain white shirt surfing in the ocean.

Figure 16. Style transfer with CLIP image conditioning. We use CLIP image embeddings extracted from the reference image shown in the first column to condition our generation. Column 2 shows the generations when CLIP image conditioning is active, while column 3 shows results when CLIP image conditioning is inactive. Our model can effectively synthesize samples consistent with the reference image style.
A dramatic oil painting of a road from a magical portal to an abandoned city with purple trees and grass in a starry night.

A Halloween scene of an evil pumpkin. A large red moon in the sky. Bats are flying and zombies are walking out of tombs. Highly detailed fantasy art.

A monster and a teddy bear playing dungeons and dragons around a table in a dark cellar. High quality fantasy art.

A highly detailed digital art of a rabbit mage standing on clouds casting a fire ball.

A red Ferrari car driving on a gravel road in a forest with rainbow beams in the distance.

A squirrel with red boxing gloves and a squirrel with blue boxing gloves fighting in a bar.

Figure 17. Samples generated by *paint-with-words*. Our method allows users to control the location of objects mentioned in the text prompt by selecting phrases and scribbling them on the image. Each phrase can contain multiple words. We show two examples in each row. Each example has the text prompt on the bottom, the scribbles on the left, and the output image on the right.
Figure 18. Comparison between images generated by our approach using (a) CLIP text encoder alone (our CLIP-text-only setting), (b) T5 text encoder alone (our T5-text-only setting), and (c) Both CLIP and T5 text encoders (our default setting). While images generated in the CLIP-text-only setting often contain correct foreground objects, they tend to miss fine-grain details, such as missing the brown jacket and hat in row 1 and the green parrot and colorful lighting in the background in row 2. Images generated in the T5-text-only setting are of higher quality, but they sometimes contain incorrect objects, such as incorrect dog breed in row 3 and tiger rendering in row 4. Our default setting that combines the strength of the CLIP text encoder and T5 text encoder produces images that better match the input text descriptions and contain the least amount of artifacts.
[63] Chitwan Saharia, William Chan, Saurabh Saxena, Lali Li, Jay Whang, Emily Denton, Seyed Kamyar Seyed Ghasemipour, Burcu Karalog Ayan, S. Sara Mahmoud, Rapha Gontijo Lopes, Tim Salimans, Jonathan Ho, David J. Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. IEEE Trans. Pattern Analysis and Machine Intelligence, 2022.

[64] Chitwan Saharia, Jonathan Ho, William Chan, Tim Salimans, David J. Fleet, and Mohammad Norouzi. Image super-resolution via iterative refinement. In Proc. NeurIPS, 2021.

[65] Noam Shazeer, Azalia Mirhoseini, Krzysztof Mazarie, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In Proc. ICLR, 2017.

[66] Shelly Sheynin, Oran Ashual, Adam Polyak, Uriel Singer, Oran Gafni, Eliya Nachmani, and Yaniv Taigman. KNN-Diffusion: Image Generation via Large-Scale Retrieval. arXiv preprint arXiv:2204.02849, 2022.

[67] Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oran Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, and Yaniv Taigman. Make-A-Video: Text-to-video generation without text-video data. arXiv preprint arXiv:2209.14792, 2022.

[68] Abhishek Sinha, Jianming Song, Chenlin Meng, and Stefano Ermon. D2C: Diffusion-decoding models for few-shot conditional generation. In Proc. NeurIPS, 2021.

[69] Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In Proc. ICML, 2015.

[70] Jianming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In Proc. ICLR, 2021.

[71] Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In Proc. NeurIPS, 2019.

[72] Yang Song and Stefano Ermon. Improved techniques for training score-based generative models. In Proc. NeurIPS, 2020.

[73] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In Proc. ICLR, 2021.

[74] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1):1929–1958, 2014.

[75] Rich Sutton. The bitter lesson. http://www.incompleteideas.net/IncIdeas/BitterLesson.html, March 2019.

[76] Yusuke Tashiro, Jianming Song, Yang Song, and Stefano Ermon. CSDF: Conditional score-based diffusion models for probabilistic time series imputation. In Proc. NeurIPS, 2021.

[77] Arash Vahdat, Karsten Kreis, and Jan Kautz. Score-based generative modeling in latent space. In Proc. NeurIPS, 2021.
A. Network Architecture

For our diffusion models, we modify the U-net architecture proposed in Dhariwal et al. [13] with the following changes:

1. Global conditioning: We add the projected pooled CLIP text embedding and CLIP image embedding along with the time step embedding in our model. Different from Saharia et al. [63], we do not use pooled T5 embeddings. CLIP text embeddings are trained to be well aligned with images, and hence, using them as global conditioning embeddings are more informative than using T5.

2. Attention blocks: After every self-attention block in the U-net model of Dhariwal et al. [13], we add a cross-attention block to perform cross-attention between image embeddings and the conditioning embeddings. The keys in the cross-attention layers are the concatenation of pre-pooled CLIP text embeddings (77 tokens), T5 embeddings (113 tokens), and pooled CLIP image embedding (1 token). In addition to these, we also add a learnable null embedding, which the model can attend to when it does not need to use any of the conditioning embeddings.

In addition, to make the super-resolution models more efficient during training and inference, we use the block structure of Efficient U-net architecture proposed in Saharia et al. [63]. Following the prior works [57, 63], we train the SR1024 model using random patches of size $256 \times 256$ during training and apply it on $1024 \times 1024$ resolution during inference. We also remove the self-attention layers and only have the cross-attention layers in this network, as computing self-attention during inference is very expensive. The U-net configurations we use for all our models are provided in Tables 2, 3, and 4 respectively.

| Table 2. Base model architecture |
|----------------------------------|
| Channel mult | [1, 2, 4, 4] |
| Dropout | 0 |
| Number of channels | 256 |
| Number of residual blocks | 3 |
| Self attention resolutions | [32, 16, 8] |
| Cross attention resolutions | [32, 16, 8] |
| Use scale shift norm | True |

| Table 3. SR256 model architecture |
|----------------------------------|
| Channel multiplier | [1, 2, 4, 8] |
| Block multiplier | [1, 2, 4, 4] |
| Dropout | 0 |
| Number of channels | 128 |
| Number of res blocks | 2 |
| Self attention resolutions | [32] |
| Cross attention resolutions | [32] |
| Use scale shift norm | True |

| Table 4. SR1024 model architecture |
|----------------------------------|
| Patch size | $256 \times 256$ |
| Channel multiplier | [1, 2, 4, 4] |
| Block multiplier | [1, 2, 4, 4] |
| Dropout | 0 |
| Number of channels | 128 |
| Number of res blocks | 2 |
| Cross attention resolutions | [32] |
| Use scale shift norm | True |
B. Ensemble training schedule

As discussed in Sec 4.1, we use a binary-tree-based branching strategy for training our ensemble model. In Tables 5 and 6, we list the exact training schedule we use to train our models. Each entry in the configuration is a tuple containing $E^{level}_{id}$ in the binary tree. This corresponds to the training out model with the noise distribution $p^{level}_{interval id} (\sigma)$. Models denoted as $M_C$ are the intermediate noise models. For these models, the configuration with a negative sign indicates all noise levels other than the one indicated. For instance, $-(9, 511)$ denotes all noise levels other than the one included in $p^{9}_{511} (\sigma)$. That is the noise is sampled from the complementary distribution $p(\sigma) \setminus p^{9}_{511} (\sigma)$.

Table 5. Training schedule for the base model. Each configuration is a tuple of (level id, interval id) in the binary tree. This configuration denotes the model trained with the noise distribution $p^{level}_{interval id} (\sigma)$. The configuration denoted with a negative sign stands for complementary distribution i.e., $p(\sigma) \setminus p^{level}_{interval id} (\sigma)$. The third column denotes the model from which the current model is initialized. The fourth column denotes the number of iterations we train our model.

| Model id | Config | Initialized from | # iterations |
|----------|--------|------------------|--------------|
| $M(0, 0)$ | (0, 0) | $M(0, 0)$ | 500K |
| $M(1, 0)$ | (1, 0) | $M(0, 0)$ | 50K |
| $M(1, 1)$ | (1, 1) | $M(0, 0)$ | 120K |
| $M(2, 0)$ | (2, 0) | $M(1, 0)$ | 50K |
| $M(2, 1)$ | (2, 1) | $M(1, 0)$ | 130K |
| $M(2, 2)$ | (2, 2) | $M(1, 1)$ | 120K |
| $M(2, 3)$ | (2, 3) | $M(1, 1)$ | 160K |
| $M(3, 0)$ | (3, 0) | $M(2, 0)$ | 40K |
| $M(3, 7)$ | (3, 7) | $M(2, 0)$ | 40K |
| $M(4, 0)$ | (4, 0) | $M(3, 0)$ | 110K |
| $M(4, 15)$ | (4, 15) | $M(3, 7)$ | 190K |
| $M(5, 31)$ | (5, 31) | $M(4, 15)$ | 100K |
| $M(9, 511)$ | (9, 511) | $M(5, 31)$ | 50K |

| $M_C(9, 511)$ | $-(9, 511)$ | Base model at 1.45M | 50K |

$M_C(9, 511)$ denotes the base model.

| Model id | Config | Initialized from | # iterations |
|----------|--------|------------------|--------------|
| $M(1, 1)$ | (1, 1) | $M(0, 0)$ | 300K |
| $M(1, 1)$ | (1, 1) | $M(0, 0)$ | 300K |

Table 6. Training schedule for the super-resolution models. Each configuration is a tuple of (level id, interval id) in the binary tree. The third column denotes the model from which the current model is initialized. The fourth column denotes the number of iterations we train our model.

| Model id | Config | Initialized from | # iterations |
|----------|--------|------------------|--------------|
| $M(0, 0)$ | (0, 0) | $M(0, 0)$ | 2M |
| $M(1, 0)$ | (1, 0) | $M(0, 0)$ | 300K |
| $M(1, 1)$ | (1, 1) | $M(0, 0)$ | 300K |

B.1. Hyper-parameters

The hyperparameters we use for training all our models are provided in Table 7.

Table 7. Hyperparameters

| Hyperparameter | Value |
|----------------|-------|
| Optimizer      | AdamW |
| Learning rate  | 0.0001 |
| Weight decay   | 0.01 |
| Betas          | (0.9, 0.999) |
| EMA            | 0.9999 |
| CLIP text embedding dropout rate | 0.2 |
| T5 text embedding dropout rate | 0.25 |
| CLIP image embedding dropout rate | 0.9 |
| Gradient checkpointing | Enabled |
| Number of iterations for base model | 1.9M |
| Number of iterations for SR256 model | 2M |
| Number of iterations for SR1024 | 1.7M |
| Sampler for base model | DEIS [89], 3kutta, Order 6, 25 steps |
| Sampler for super-resolution models | DEIS, 3kutta, Order 3, 10 steps |