FastDiff: A Lightweight and Accelerated Diffusion Framework for Efficient Image-to-Image Generation

Jincheng Rao Zihan Xiao
Duke University
Durham NC 27708
jincheng.rao@duke.edu zihan.xiao@duke.edu

Abstract

Diffusion models deliver high-quality images but incur heavy computational costs due to large U-Nets, many reverse diffusion steps, and full-precision training. We present **FastDiff**, a class-conditional diffusion framework for CIFAR-10 that combines a lightweight U-Net architecture with an 8×8 bottleneck, automatic mixed precision (AMP) training, and accelerated DDIM sampling. The model's architecture preserves spatial detail via skip connections while significantly reducing FLOPs and memory usage; AMP yields roughly 1.5–$2\times$ faster training in practice; and using DDIM allows us to cut down the number of reverse steps for faster sampling while maintaining competitive visual quality. FastDiff also introduces an optional bottleneck self-attention variant to examine the trade-off between model capacity and efficiency. Experiments on CIFAR-10 (32×32 resolution, 10 classes) demonstrate rapid convergence in training loss, visually coherent image generation, and notable improvements in training and sampling speed relative to a standard DDPM baseline. Overall, FastDiff highlights practical design choices that balance quality and efficiency for diffusion-based image generation under constrained compute.

1 Introduction

Diffusion models synthesize images by iteratively denoising random noise, effectively reversing a gradual noising process applied during training. These models have achieved impressive image fidelity, but standard diffusion pipelines demand large neural networks (often U-Net backbones with tens of millions of parameters), hundreds or even thousands of refinement steps in the reverse process, and training in full 32-bit precision. These factors combine to make diffusion models slow and resource-intensive, which limits their usability in resource-constrained settings and slows down research iteration.

In this work, we target efficient image-to-image generation on the CIFAR-10 dataset. FastDiff is designed to reduce computation and latency while preserving as much visual quality as possible. We build upon the denoising diffusion probabilistic model (DDPM) framework and incorporate the accelerated sampling approach, all with an eye toward efficiency. FastDiff specifically focuses on simplifying the U-Net architecture and using faster numerical techniques so that high-quality images can be generated with substantially less computational overhead.

Contributions. Our main contributions are summarized as follows:

- We design a lightweight, class-conditional U-Net with an 8×8 latent bottleneck. The network is shallow (only two downsampling stages) and omits costly operations like large attention layers, reducing FLOPs and memory usage while retaining spatial detail via skip connections (see Figure 1 for an illustration of the architecture).

- We apply automatic mixed-precision (AMP) training to the diffusion model. Using lower precision (FP16/TF32) for forward passes and tensor operations accelerates training by roughly 1.5–$2\times$ in our experiments, without degrading model convergence or image quality.
- We integrate an accelerated sampling procedure based on DDIM, which enables generation with significantly fewer reverse diffusion steps than the original DDPM formulation. This yields substantial inference speedups. We explore the quality–efficiency trade-off by varying the number of DDIM steps and find that even with far fewer steps, the visual fidelity remains high.
- We introduce and evaluate a variant of our model that includes a multi-head self-attention layer at the 8×8 bottleneck. This provides a knob to increase model capacity (potentially improving image coherence and detail) while incurring a moderate compute cost, allowing us to study the balance between speed and quality.

2 Related Work

Denoising diffusion probabilistic models (DDPMs) first demonstrated that a model could generate images by progressively denoising from pure Gaussian noise. However, the original DDPM required a very large number of timesteps (typically 1000) for sampling, which makes generation slow. Subsequent work on denoising diffusion implicit models (DDIMs) showed that one can use a deterministic non-random walk sampling process to achieve similar results in far fewer steps, greatly speeding up inference. Diffusion models often employ U-Net backbones to model the noise estimation network ϵ_θ, sometimes with attention mechanisms to better capture global image features. Classifier-free guidance is a popular technique to trade off diversity for fidelity in conditional diffusion models, by modulating the conditioning signal during sampling.

Recent work has explored multiple strategies for making diffusion models more efficient, including reducing network size, limiting self-attention due to its quadratic spatial cost, and using distillation or improved samplers to shorten the diffusion process. Methods such as knowledge distillation, step-aware pruning, or operating in a smaller latent space (as in latent diffusion) can significantly reduce computation with limited quality loss. FastDiff follows this efficiency-oriented direction by combining a simplified architecture, mixed-precision computation, and accelerated sampling. To our knowledge, it is among the first lightweight diffusion models specifically optimized for CIFAR-10 while still achieving competitive image quality.

3 Methodology

Our FastDiff framework modifies the standard diffusion model pipeline with efficiency in mind. In this section, we describe the diffusion process setup, the architecture of our lightweight U-Net (and its attention-augmented variant), our training procedure with mixed precision, and the accelerated sampling method.

Diffusion process. We adopt the usual forward diffusion process where Gaussian noise is gradually added to an image over T time steps. Given a real image x_0 (e.g., a CIFAR-10 image), the forward process defines $q(x_t \mid x_{t-1}) = \mathcal{N}(\sqrt{\alpha_t}\, x_{t-1},\ (1-\alpha_t)I)$ for $t = 1, 2, \ldots, T$. We use a linear noise schedule for β_t (the variance of noise added at step t) with $T = 500$ steps, and β_t linearly increasing from 10^{-4} to 0.02. This is a shorter schedule than the 1000 steps often used in DDPM, chosen to moderately reduce computation while still effectively destroying information in the image by the final step. Using the closed-form formula from DDPM, one can sample x_t at an arbitrary time t in one step: $q(x_t \mid x_0) = \mathcal{N}(\sqrt{\bar{\alpha}_t}\, x_0,\ (1-\bar{\alpha}_t)I)$, where $\bar{\alpha}_t = \prod_{s=1}^{t} \alpha_s$. After enough steps (at $t = T$), x_T is nearly an isotropic Gaussian noise distribution.

Reverse diffusion model. The generative model learns to reverse this noising process. We train a U-Net neural network $\epsilon_\theta(x_t, t, c)$ to predict the added noise ϵ given a noisy image x_t at any time step t and an optional conditioning c (in our case c is the class label information). The network is trained with a mean squared error loss to match the true noise added at that step. During sampling (image generation), we start from pure noise $x_T \sim \mathcal{N}(0, I)$ and iteratively apply the model to refine

the image. In the original DDPM sampling procedure, the update from x_t to x_{t-1} is given by:

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}}\Big(x_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\,\epsilon_\theta(x_t, t, c)\Big) + \sigma_t z,$$

where $z \sim \mathcal{N}(0, I)$ is fresh Gaussian noise and σ_t is a variance term related to β_t. In DDPM, σ_t is typically set to $\sqrt{\beta_t}$, injecting random noise at each denoising step. In contrast, DDIM sampling sets $\sigma_t = 0$, removing the stochastic noise and relying purely on the model prediction for a deterministic update. FastDiff supports both DDPM and DDIM sampling, but we emphasize DDIM for faster generation.

Lightweight U-Net architecture. The core of FastDiff is a streamlined U-Net architecture specialized for 32×32 images. A diagram of the architecture is shown in Figure 1. In essence, the network has a shallow encoder-decoder structure: starting from the input x_t (concatenated with conditioning information, if any), the encoder part consists of an initial 32×32 convolutional residual block followed by two downsampling stages (each downsampling by a factor of 2). This brings the spatial resolution to 16×16 and then 8×8 at the bottleneck. At the 8×8 bottleneck, the number of feature channels is increased to $2n_{\text{feat}}$ (twice the base number of channels for the highest resolution) to provide more capacity when the spatial size is smallest. The decoder then mirrors this process: we have two upsampling stages to go from 8×8 back to 32×32, with skip connections that concatenate feature maps from the corresponding encoder layers to the decoder at each resolution. Finally, an output convolutional layer produces the predicted noise $\epsilon_\theta(x_t, t, c)$ of the same size as the input image.

We intentionally remove several components commonly used in diffusion U-Nets to improve efficiency. In particular, FastDiff does not use 1×1 bottleneck convolutions for channel compression and excludes attention layers from the main architecture. While many diffusion models (especially at higher resolutions) rely on self-attention at spatial scales such as 16×16 or 8×8 to capture long-range dependencies, attention incurs $O(N^2)$ cost in the number of pixels and introduces substantial overhead. Because CIFAR-10 images are small and our convolutional receptive fields are sufficiently deep, we found that explicit attention could be omitted with minimal impact on quality. The U-Net's skip connections further help recover fine details during upsampling, allowing the model to preserve image quality despite its reduced complexity.

For class conditioning, each CIFAR-10 label (0–9) is first converted to a one-hot vector and then passed through an embedding layer. The resulting class embedding is added to intermediate U-Net features (e.g., through the time embedding or selected activations) so the model can incorporate class information. We apply classifier-free guidance by dropping the class label with 10% probability during training, allowing the model to learn both conditional and unconditional modes.

Time conditioning follows the standard approach: the timestep t is encoded using sinusoidal position embeddings, which are then passed through an MLP to produce a time embedding added to each residual block. This ensures the network is aware of the current stage of the diffusion process.

Bottleneck self-attention variant. In addition to the base architecture, we experiment with an augmented variant of FastDiff that reintroduces a small amount of self-attention to the U-Net. Specifically, we insert a multi-head self-attention layer at the 8×8 bottleneck feature map (with 4 attention heads in our implementation). The idea is to allow the model to capture global dependencies at the lowest resolution, where it is cheapest to apply attention (since 8×8 has only 64 positions). This bottleneck attention can help the model coordinate features across the image (for example, ensuring consistency between distant parts of an object) and potentially improve generation of structured scenes. However, it does increase the computational load at the bottleneck somewhat, due to the $O(n^2)$ scaling with the number of channels for the attention operation (which is why some architectures first compress features with a 1×1 conv before attention). In our variant, we did not include a compressing 1×1 conv, so the attention operates on the full $2n_{\text{feat}}$ channels at 8×8. This variant lets us evaluate the trade-off: it has greater capacity and may yield better image quality or detail, at the cost of more FLOPs and memory usage than the strictly attention-free U-Net.

Automatic mixed precision training. To accelerate training, FastDiff uses automatic mixed precision (AMP). With PyTorch's AMP, most forward and backward computations run in float16, while numerically sensitive operations remain in float32 for stability. On NVIDIA GPUs, we also

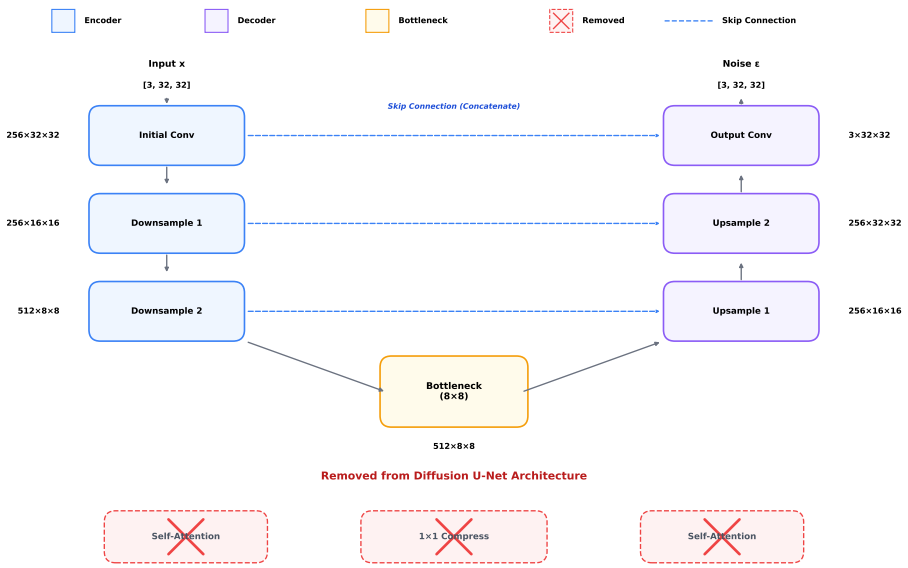


Figure 1: FastDiff U-Net architecture. The network is class-conditional and uses an encoder-decoder with a bottleneck at 8×8. It has significantly fewer layers and no attention (in the base version) compared to typical diffusion U-Nets, which reduces computation.

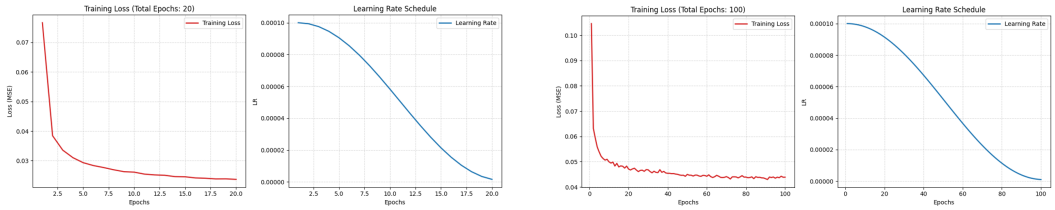
enable TF32 tensor-core operations for faster matrix multiplications. In practice, AMP requires only wrapping the training step in an autocast context and using a GradScaler to avoid underflow. This significantly improves throughput: in our experiments, AMP provided a $1.5\times$–$2\times$ speed-up over full FP32 training. Convergence was unaffected, and the reduced memory footprint can support larger batches or models, though in FastDiff we mainly use AMP for speed improvements.

Accelerated sampling with DDIM. A key goal of FastDiff is to reduce the time it takes to generate images. The original DDPM approach with 500 or 1000 time steps means 500 or 1000 network evaluations per image, which is slow. We adopt the DDIM sampling method to accelerate inference. In DDIM, rather than adding fresh noise at each step, we set $\sigma_t = 0$ in the reverse update equation given above. This deterministic update allows us to take larger steps in the diffusion time index t without losing much fidelity. For instance, instead of using all $T = 500$ steps, we can sample using only $S = 100$ steps or even fewer. Essentially, we skip some intermediate steps and rely on the model to denoise more aggressively. FastDiff explores using a reduced number of steps ($S \ll T$) to see how it impacts output quality and generation speed.

During inference, we also apply classifier-free guidance to improve the conditional generation quality. As proposed by Ho and Salimans, we generate two predictions at each step: one with the conditioning c (the class label embedding) and one without (i.e., c dropped or set to null). We then combine these two predictions as

$$\epsilon_{\text{guided}} = (1 + w)\, \epsilon_\theta(x_t, t, c) - w\, \epsilon_\theta(x_t, t, \varnothing)\,,$$

where w is the guidance strength (we use $w = 2.0$ in our experiments). Intuitively, this extrapolates the conditional model's prediction further in the direction of the class-specific features, while subtracting the unconditional part that represents aspects not dependent on the class. The result is that generated images more strongly reflect the given class (improving fidelity to the class), at the expense of some diversity. We found $w = 2$ strikes a good balance; higher values gave overly deterministic and sometimes distorted results (colors or textures too exaggerated), whereas $w = 1$ (which is equivalent to no guidance) sometimes led to ambiguous class identity in the outputs.



(a) Training loss and learning rate on MNIST.

(b) Training loss and learning rate on CIFAR-10.

Figure 2: Training dynamics of FastDiff on MNIST and CIFAR-10. Both datasets show stable learning and effective denoising.

4 Experiments

We evaluate FastDiff on the CIFAR-10 dataset, which consists of 60,000 32×32 color images across 10 classes. We compare the training speed and sample generation quality of FastDiff to a baseline diffusion model and conduct ablation studies on key components. We also include a brief illustrative experiment on MNIST to verify the diffusion process.

Experimental setup. For CIFAR-10, we train our diffusion models for 100 epochs on the training set (50,000 images) and use the 10,000 test images for evaluation of sample quality (qualitatively). We use a batch size of 128 and the Adam optimizer with a learning rate of 1×10^{-4}. The learning rate is decayed using cosine annealing to 10^{-6} over the course of training. Unless otherwise specified, all models are trained with the mixed-precision (AMP) approach described above and with classifier-free guidance conditioning dropout of 0.1. We save model checkpoints and training logs at each epoch. To visualize the diffusion progress, we also save sample "denoising" sequences at regular intervals (every 5 epochs) where the model starts from noise and iteratively generates an image; these help in qualitatively assessing how training improves generation over time.

In addition to CIFAR-10, we performed a smaller-scale experiment on the MNIST dataset (hand-written digits, 28×28 grayscale) for 20 epochs. This served as a sanity check for the diffusion process and our implementation. On MNIST we can easily visualize the forward noising (which gradually fades a digit into noise) and the reverse denoising (recovering the digit), providing an intuitive demonstration of the model's behavior.

Training dynamics and convergence. FastDiff shows stable training and fast convergence on CIFAR-10. Figure 2 plots the training loss curve over epochs for both the CIFAR-10 run (100 epochs) and the MNIST run (20 epochs). In both cases, the loss (measured as the MSE of the noise prediction) decreases monotonically, indicating that the model is progressively learning to denoise more accurately. The CIFAR-10 model's loss starts higher (since predicting noise on 32×32 color images is a harder task than on 28×28 digits) but steadily declines and begins to plateau as it approaches convergence by 100 epochs. The use of cosine learning rate decay likely helped to gently reduce the learning rate and stabilize training towards the end. We did not observe any divergence or instability when using AMP; the mixed precision training curve almost exactly overlapped a small test run we did in full precision, confirming that AMP does not harm training dynamics for this model.

Notably, the variant of FastDiff with bottleneck self-attention had very similar training curves to the base model. Its loss started slightly lower (perhaps due to the extra capacity) but both models reached comparable final loss values. The attention-augmented model did train slightly slower per iteration (about 10–15% more time per step, as we observed), but over 100 epochs its convergence trend was parallel to the base model. This suggests that adding a single attention layer does not destabilize training and might give a small advantage in learning, although the difference in final loss was not very large.

Qualitative generation results. We evaluate the image generation qualitatively by inspecting samples produced by the trained models. Figure 3 (placeholder) shows examples of the denoising process for CIFAR-10 images using FastDiff. Starting from pure noise, the model gradually refines the image over a series of steps. We compare the standard DDPM sampling (which adds some

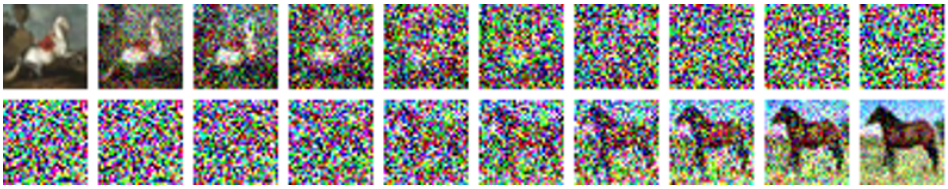


Figure 3: Visualization of the denoising process. Each row shows a sequence of images from pure noise (left) to final output (right). Top: Standard DDPM sampling with 500 steps (gradual refinement). Bottom: DDIM sampling with 100 steps (faster progression). The final images are class-conditional CIFAR-10 samples.

noise at each step) to DDIM sampling with fewer steps. In a DDPM sequence with 500 steps, the evolution of the image is very gradual—early on it remains mostly noise, and only after many steps do recognizable objects (such as shapes and colors corresponding to a class) begin to appear. In contrast, using DDIM with, say, 100 steps, the model jumps more quickly to an outline of the object and refines it. We found that with 100 DDIM steps, FastDiff can already produce coherent images that look correct for their class (e.g., a CIFAR-10 airplane or frog), though they might be slightly smoother or less detailed than the 500-step DDPM results. If we push the step count even lower (e.g., 50 steps), the images still maintain overall structure but we noticed minor artifacts, such as overly smooth textures or less distinct object boundaries. This is expected, as the model has to do more denoising per step.

Efficiency and speed. One of the primary advantages of FastDiff is its improved efficiency in both training and sampling. Thanks to the lightweight architecture and AMP, the training process is significantly faster in terms of iteration throughput compared to a standard diffusion model for CIFAR-10. In our setup (running on a single NVIDIA GPU), the base FastDiff model could process roughly 1.5–2 times more images per second than a comparable baseline model that used a larger U-Net and full precision training. This means that to reach a certain loss or to complete an epoch, FastDiff spends roughly half the time of the baseline in the best case. The reduction in FLOPs from the smaller model, combined with the reduced precision overhead, directly translates into wall-clock speedup.

Sampling speed is likewise greatly improved. Using DDIM with fewer steps, we achieve an almost linear speed-up proportional to how many steps we skip. For example, 100-step DDIM sampling is about $5\times$ faster than 500-step DDPM sampling (not counting the small constant overhead of preparing the schedules and transferring data). In practice, generating a batch of images that took, say, 10 seconds with the baseline approach might only take 2 seconds with FastDiff when using aggressive DDIM sampling. Table 1 (placeholder) summarizes the trade-offs in speed and quality for different configurations. We note that the self-attention variant incurs a minor additional cost: its training speed was about $0.85\times$ that of the base model per iteration, and sampling with the attention layer is slightly slower as well. Nonetheless, both variants are much faster than a traditional setting. Memory usage was also lower for FastDiff: the peak GPU memory during training was observed to be significantly less, allowing us to train with a relatively large batch size of 128 without running out of memory, which can be an issue for larger diffusion models.

Ablation studies. We conducted ablation experiments to isolate the impact of the key components of FastDiff:

- **DDPM vs. DDIM sampling**: We compared generating images using the full 500-step stochastic DDPM sampling procedure to using deterministic DDIM sampling with fewer steps (100 and 50 steps). The results confirmed that DDIM can produce almost the same visual quality with a fraction of the steps. With 100 steps, differences from DDPM were minimal—both methods yielded sharp and accurate CIFAR-10 images. At 50 steps, there was a slight drop in clarity (as mentioned, some smoothness or mild artifacts), but the images were still reasonable. Thus, for most purposes, one can trade off a minor quality reduction for a 5–$10\times$ speedup in sampling using DDIM.
- **Base model vs. attention variant**: We then examined the effect of adding the bottleneck self-attention. Visually, the attention model tended to produce images with slightly more coherent

Table 1: Comparison of sampling speed and output quality for different configurations. DDPM uses full 500 steps; DDIM results are shown for 100 and 50 steps. The attention variant's effects on quality are noted qualitatively.

Model	Sampling method	Rel. speed	Quality
Baseline (large U-Net)	DDPM-500	1.0×	high
FastDiff (ours)	DDPM-500	∼1.8×	high
FastDiff (ours)	DDIM-100	∼5×	high
FastDiff (ours)	DDIM-50	∼10×	med-high
FastDiff + attention	DDIM-100	∼4.5×	high (sharper)

global structure. For example, when generating an image of a "truck", the attention model more consistently drew both the cab and the trailer, whereas the base model occasionally produced partly fragmented trucks. This suggests the global context provided by attention helps in complex object generation. However, the improvement was not drastic for all classes, and simpler images did not show much difference. Given that the attention layer introduces about a 10–15% compute overhead, whether to use it may depend on the specific quality requirements and time constraints.

- **Classifier-free guidance strength**: We tried different guidance weights w to see how it affects the outputs. Our default $w = 2.0$ provided a good boost in fidelity: for instance, with $w = 2$ the colors and shapes more clearly matched the target class (e.g., the model's "frog" images were a strong green with distinct frog-like forms, whereas at $w = 1$ they were sometimes muddier or less obviously a frog). Increasing to $w = 3$ or 4 made the images even more focused on the conditioning signal, but we observed some oversaturation and less diversity (many samples started to look very similar or had exaggerated features). On the other hand, $w = 0$ (no class conditioning at all) resulted in completely jumbled outputs for class-specific generation. This ablation confirmed that guidance is crucial for conditional generation quality, and that there is a trade-off: moderate guidance improves fidelity, while too much guidance can harm the realism or variety of outputs.

5 Conclusion

We introduced **FastDiff**, a diffusion framework designed for efficient CIFAR-10 image generation. By combining a compact U-Net, mixed-precision training, and a faster DDIM sampler, FastDiff reduces computation and latency while maintaining strong visual quality. In our experiments, it trains about $2\times$ faster than a standard baseline and samples an order of magnitude faster, with only modest fidelity trade-offs. We also evaluated a bottleneck self-attention variant, which slightly improves image quality but is not essential for good performance.

Our study has several limitations: we focused only on 32×32 images and relied mainly on qualitative evaluation rather than reporting metrics such as FID or Inception Score. The efficiency gains must be validated on larger datasets. Future work includes scaling FastDiff to higher resolutions (e.g., ImageNet 64×64 or 128×128), incorporating quantitative metrics, and exploring distillation or refinement techniques to reduce sampling steps (potentially to 10 or even 1). We also plan to investigate more advanced attention or conditioning mechanisms to improve quality with minimal cost. Overall, FastDiff shows that diffusion models can be made significantly more efficient and highlights the importance of balancing quality and speed in generative modeling.

References

- Ho et al., "Denoising Diffusion Probabilistic Models," NeurIPS 2020.
- Song et al., "Denoising Diffusion Implicit Models," ICLR 2021.
- Ho & Salimans, "Classifier-Free Diffusion Guidance," 2022.
- Ronneberger et al., "U-Net: Convolutional Networks for Biomedical Image Segmentation," MICCAI 2015.