TOWARDS EFFICIENT DIFFUSION MODELS FOR TEXT-CONDITIONED IMAGE SYNTHESIS: ARCHITECTURE DESIGN, SAMPLERS AND DISTILLATION

ABSTRACT

The emerging latent diffusion models for image synthesis have shown great image quality. However, compared with other generative models like generative adversarial networks, the inference time required for diffusion models is relatively high. On the other hand, the sizes of diffusion models from the industry are increasing, leading to higher training and inference costs. In this review, we focus on the methods that improve the efficiency of the backbone, and faster solvers to reduce the number of steps required for generating high-quality images.

1 INTRODUCTION

In recent years, latent diffusion models(LDMs)(1) have become prominent in text-conditioned image synthesis, as higher resolution and more realistic photos can be generated. However, this led to larger sizes of LDMs from the industry, well-known LDMs like Stable Diffusion 3.5 with 8.1B parameters, Flux.1(2) with 12B parameters and Playground v3 with 24B parameters(3), which are computationally expensive and unlikely to be deployed on devices with consumer-grade and mobile hardware. The way LDMs sample the previous time latent varibales, by iterative sampling and denoising the latent variables, requires multiple steps. Also, for the state-of-the-art LDM backbones, they contain attention-based layers are inefficient due to the quadratic time complexity, making the sampling way more inefficent. Therefore, research on accelerating LDMs are mainly in the following areas:

- The backbone architecture: The backbone of the LDM is responsible for noise prediction. The state-of-the-art backbone of LDMs, Diffusion Transformer(DiT), due to the quadratic time complexity of self-attention of the transformer architecture, the noise prediction phase becomes inefficient for high-resolution image generation. Hence designing efficient backbone architectures becomes a trend in DM-related research.
- Faster sampling: To accelerate the sampling during the inference phase, the two main areas are:
  - Efficient solvers: The original explicit, stochastic sampling method from Denoising Diffusion Probablistic Model(DDPM)(4) is tremendously time-consuming, requiring thousands of Number of Function Evaluations(NFE) to generate high-fidelity images. The later implicit and deterministic method, Denoising Diffusion Implicit Model(DDIM)(5) only requires around hundreds of NFE. In further research on solvers, reducing NFE for high-fidelity images is still a dominant topic in image synthesis.
  - Distillation: Distillation has shown great potential for efficient sampling. In recent implementations of distillation methods, the distilled models can produce high-quality images even with a single step. The state-of-the-art distillation approach can also reduce the size of the model, making lightweight but powerful models.

This review aims to discuss topics related to more efficient diffusion models, specifically for faster inference time, with a focus on faster sampling methods and training methods that can accelerate inference phase. In Section 2, the background of the latent diffusion models, sampling methods in the reverse process and the backbone are introduced to better understand why inferencing of latent diffusion models are inefficient. In Section 3, the state-of-the-art architecture designs, sampler methods, and distillation methods for faster sampling are discussed, with rectified flow, a concept that further improves the sampling. In Section 4, the potential future works and the conclusion for the review are given.

2 RELATIVE BACKGROUND

This section gives the basic concepts of text-conditional LDMs, with the original diffusion model algorithms, state-of-the-art backbone architectures and previous works on efficiency improvements on the sampling methods.

2.1 LATENT DIFFUSION MODEL

Denoising Diffusion Models (DMs), initially introduced by Sohl-Dickstein et al.(6; 7), are a class of unsupervised generative models that operate in two joint distributions act as two stages: a forward process in Eq.1 and a reverse process in Eq. 2.

$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}), \quad q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \tag{1}$$

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t), \quad p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t)) \tag{2}$$

which is a Markov Chain gradually perturbed with Gaussian noise to destroy its structure with a total of T steps to obtain $\mathbf{x}_T$, and a reverse process, where the model restores the perturbed $\mathbf{x}_T$ step by step to $\mathbf{x}_0$, to generate a sample from a normally distributed variable that $p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_t; \mathbf{0}, \mathbf{I})$, as shown in Figure 1.

To reduce computational costs and learn important representations, we adopt the diffusion models in the latent space, in which the encoder part of the autoencoder $\mathcal{E}$ is used to compress the data into lower-dimensional variables, also known as latent variables, and restore latent variables with the decoder part $\mathcal{D}$ to the original sizes, therefore, they are also known as latent diffusion models(LDMs), which were proposed by Rombach et al.(1) for text-conditioned image synthesis, where the architecture is shown in Figure 2.

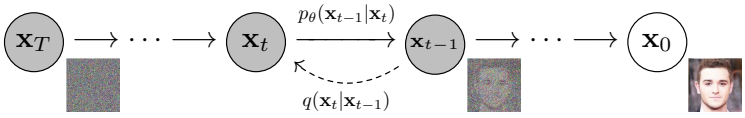


Figure 1: The forward and reverse process of the diffusion model. Figure from Ho et al.(7)

For text-conditioned image synthesis tasks, the LDM has a structure that includes the text encoder to ensure the text-conditioned latent variable, the backbone that works as the noise predictor, the noise scheduler during the training phase to define the noise schedule, the sampler during the inference phase in the reverse process, and the autoencoder for data compression during the training phase and generative images from the latent variable. Moreover, Ho and Salimans found that jointly training a conditional and unconditional model, and sampling with a combination of predicted results from the conditional and unconditional model, can generate images that attain a trade-off between sample quality and diversity, and such method, which is called Classifier-Free Guidance(CFG)(8).

2.2 BACKBONE OF LDMS

The backbone of the LDM is a denoising neural network to predict noise $\epsilon_\theta(\mathbf{z}_t, t)$ given the timestep t and the latent variable related to time $\mathbf{z}_t$. In the original LDM implementation, the transformer architecture(9), which is overwhelmingly popular in tasks related to natural languages, as well as in previous research in image synthesis for GAN and autoregressive models, combined with U-Net(10), a convolutional neural network architecture originally used in biomedical image segmentation as the backbone. Later, Peebles and Xie proposed Diffusion Transformer(DiT) (11) by adopting the Vision Transformer (ViT) architecture(12) to first "patchify" the latent variable into a sequence of tokens with positional embedding methods of ViT, with adaptive layer norm and the adaLN-zero block by zero-initializing the final convolutional layer to speed up training. In the class-conditional image synthesis experiment trained on ImageNet, DiT has outperformed all other backbones for LDMs, with the notable scaling law of Transformer applied to diffusion models. Both of the state-of-the-art

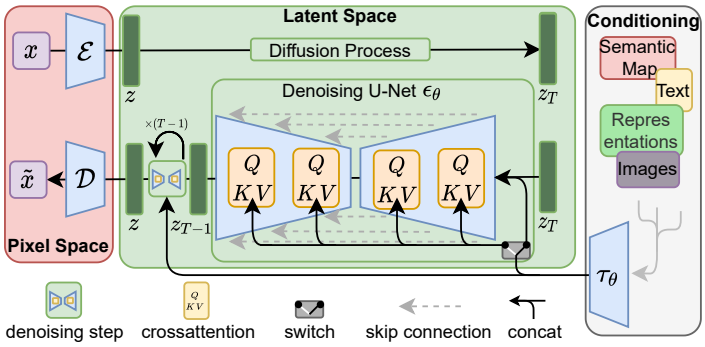


Figure 2: The architecture of the LDM. Figure from Rombach et al.(1)

backbones contain attention-based layers with original softmax attention from Vaswani et al.(13) with query Q, key K and value V:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^\top}{\sqrt{d}})V \tag{3}$$

where the U-Net in LDM and DiT use cross-attention and self-attention respectively. The drawback of attention-based layers is due to the multiplication of matrices $QK^\top$, which makes attention-based layers inefficient with the time complexity $\mathcal{O}(N^2)$.

2.3 Sampling method of LDMs

In the original DDPM method(7), the sampling phase at time t for the previous state $\mathbf{x}_{t-1}$ with t from T to 0 is defined as:

$$\mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}}(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)) + \sigma_t \mathbf{z}, \quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \text{ if } t > 1 \text{ else } \mathbf{z} = 0 \tag{4}$$

Where α_t is equal to $1 - \beta_t$ where β_t is the noise schedule, σ^2 is typically set to $\sigma^2 = \tilde{\beta}_t = \beta_t \cdot \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}$, $\bar{\alpha}$ is the product of α_t from 1 to current t and $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ is the predicted noise given by the backbone. However, this sampling method is stochastic and explicit. Song et al. first proposed score-based generative modelling with Stochastic Differential Equations(SDEs)(14) which samples by solving SDE $\mathrm{d}\mathbf{x}$ based on the deterministic term with drift coefficient and a stochastic term with $\bar{\mathbf{w}}$ denoted as a standard Wiener process, also known as Brownian Motion, in the reverse-time direction, and the term $\nabla_{\mathbf{x}} \log p_t(\mathbf{x})$ is the score function, which is approximated by the backbone output $\nabla_{\mathbf{x}} \log p_t(\mathbf{x}) \approx \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$ as shown in Eq.5. For solving SDE, they found methods for SDEs such as Euler-Maruyama and Runge-Kutta methods can be applied to the score-based function. They also proved that the SDE can be converted into a probability flow ODE as shown in Eq.6 with the Fokker-Planck equation enabling faster sampling.

$$\mathrm{d}\mathbf{x} = [f(\mathbf{x}, t) - g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}}, \tag{5}$$

$$\mathrm{d}\mathbf{x} = \left[f(\mathbf{x}, t) - \frac{1}{2} g(t)^2 \nabla_{\mathbf{x}} \log p_t(\mathbf{x})\right] \mathrm{d}t, \tag{6}$$

For DDIM(5), it can be considered a special case of a generalized DDPM as shown in Eq.7. which in the case of DDIM, where σ_t is defined by a hyperparameter η that $\sigma_t(\eta) = \eta\sqrt{\frac{1-\alpha_{t-1}}{1-\alpha_t}}\sqrt{\frac{1-\alpha_t}{\alpha_t}}$, and for DDIM, η is set to 0, hence the noise term $\sigma_t \mathbf{z}$ is omitted, and makes the sampling implicit and deterministic. In the DDIM work, Song et al. find that DDIM is also a special case of the probability ODE from the work of score-based diffusion. Sampling with DDIM, the number required for generating high-fidelity images is significantly reduced from around 1000 steps with DDPM to around a hundred steps.

$$\mathbf{x}_{t-1} = \sqrt{\alpha_{t-1}} \underbrace{\left(\frac{\mathbf{x}_t - \sqrt{1-\alpha_t}\,\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}{\sqrt{\alpha_t}} \right)}_{\text{“predicted } \mathbf{x}_0\text{”}} + \underbrace{\sqrt{1-\alpha_{t-1}-\sigma_t^2} \cdot \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)}_{\text{“direction pointing to } x_t\text{”}} + \underbrace{\sigma_t \mathbf{z}}_{\text{random noise}} \tag{7}$$

3 Literature Review

This section is the literature review of the efficient design for the text-conditioned image synthesis LDMs, including the methods for efficient backbones, efficient sampling methods and distillation methods for LDMs.

3.1 Efficient Backbone

Research on efficient backbone architectures has become popular due to the quadratic time complexity of the self-attention blocks of both state-of-the-art DiT and U-Net architectures for LDM. There are two main areas of improvement:

- One focuses on linear time attention based on the Transformer with linear time complexity during inference, for instance, the linear attention(15) by substituting the original softmax attention into a kernel method with linear time complexity in Eq. 8 using the elu activation function for the kernel $\phi(\cdot) = \text{elu}(\cdot) + 1$. In the work by Xie et al.(16), they adopt the linearized attention using the ReLU activation function $\phi(\cdot) = \text{ReLU}(\cdot)$ for DiT. They also discovered that the linear attention model suffered from slower convergence. Therefore, they substitute the Multilayer-Perceptron Feed-Forward Network(MLP-FFN) block of DiT with a Mix-FFN block(17), by using a 3x3 convolutional block with gated units. They even found that the positional embedding for the patchified tokens can be omitted with the Mix-FFN block.

$$\text{Attention}(Q, K, V) = \frac{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j) V_j^T}{\phi(Q_i)^T \sum_{j=1}^{i} \phi(K_j)} \tag{8}$$

- Other than improving the efficiency of the Transformer architecture, one focuses on attention-free architectures. Discretized State Space Models(SSMs) proposed by Gu et al.(18) have been considered an efficient substitute for the Transformer architecture with relatively lower time complexity during the inference phase, and have shown better performance for addressing long-sequence dependencies. Yan et al. proposed DiffuSSM by constructing bidirectional SSM blocks and found bidirectional structure state-of-the-art performance with the time complexity of $\mathcal{O}(N \log N)$(19). In the recent work, Liu et al. proposed a novel architecture LinFusion(20) by applying generalized linear attention within the Mamba-2 architecture for SSMs, further improving efficiency.

3.2 Efficient Sampling

This section is about efficient sampling methods, including SDE/ODE solvers and distillation for LDMs.

3.2.1 Samplers with Few Time Steps & Rectified Flow

The sampling methods given in Section 2.3 are all first-order Euler methods for ODE solvers, and are still extremely slow as they generate images in hundreds or thousands of steps with reasonable quality, showing the score-based sampling and DDIM still need further improvements. DPM-Solver proposed by Lu et al.(21) enables higher-order ODE solvers for both continuous-time and discrete-time DMs, by approximating the diffusion ODE using the Taylor expansion over the exponentially weighted integral $\int e^{-\lambda} \hat{\boldsymbol{\epsilon}}_\theta(\hat{\mathbf{x}}_\lambda, \lambda) \mathrm{d}\lambda$ of the change of Signal-Noise Ratio(SNR $= \frac{\alpha^2}{\sigma^2}$) between the current time t and the previous time $t-1$:

$$\mathbf{x}_{t_{i-1}\to t_i} = \frac{\alpha_{t_i}}{\alpha_{t_{i-1}}}\tilde{\mathbf{x}}_{t_{i-1}} - \alpha_{t_i}\sum_{n=0}^{k-1}\boldsymbol{\epsilon}_\theta^{(n)}\left(\hat{\mathbf{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}}\right)\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\,\mathrm{d}\lambda + \mathcal{O}(h_i^{k+1}) \quad (9)$$

where $\lambda_t = \log\frac{\alpha_t}{\sigma_t}$ is the one half of the log SNR, $h_i = \lambda_{t_i} - \lambda_{t_{i-1}}$, and $\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(\hat{\mathbf{x}}_\lambda, \lambda) = \frac{\mathrm{d}^n\hat{\boldsymbol{\epsilon}}_\theta(\hat{\mathbf{x}}_\lambda,\lambda)}{\mathrm{d}\lambda^n}$ is the n-th order derivative of $\hat{\boldsymbol{\epsilon}}_\theta(\hat{\mathbf{x}}_\lambda, \lambda)$, $h_i = \lambda_t - \lambda_{t-1}$ and $\mathcal{O}(h_i^{k+1})$ is the error term of the Taylor expansion, typically omitted. Zhang and Chen discovered that the Euler method for solving ODE suffered from drawbacks such as low accuracy and volatility when not choosing a small step size(22). To handle this, they proposed Diffusion Exponential Integrator Samplers(DEIS) to adopt exponential integrators for the probability flow ODE, which also enables higher-order ODE solvers for multiple variants using the Runge-Kutta method, Heun's method and Adam-Bashforth method. DEIS is also a multistep method solver that saves the previous noise prediction outputs into the buffer and reuses it during the reverse process as this extrapolation shows better results. Lu et al., based on their previous work on DPM-Solver, proposed DPM-Solver++(23) as they found, for conditional LDMs using classifier-free guidance, the higher-order sampling methods produce unstable results and more time required for generating high-fidelity images when the classifier-free guidance scale is large. They combine the multistep way of solving ODE of DEIS and, instead of using the change-of-variable for the noise $\hat{\boldsymbol{\epsilon}}_\theta$, DPM-Solver++ uses the change-of-variable of the predicted data $\hat{\mathbf{x}}_\theta \approx \mathbf{x}_\theta := \frac{\mathbf{x}_t - \sigma_t\boldsymbol{\epsilon}_\theta}{\alpha_t}$ and substitute $\hat{\boldsymbol{\epsilon}}_\theta^{(n)}(.)$ into $\hat{\mathbf{x}}_\theta^{(n)}(.)$. They also derived a SDE version of DPM-Solver++, based on the score-based SDE method by Song et al.(14). This method has shown that higher-order DPM-Solver++ can stabilize the reverse process compared with first-order DDIM and higher-order DEIS and DPM-Solvers when choosing a large guidance scale. An overview of the details of higher-order samplers is arranged in Table 1.

$$\tilde{\mathbf{x}}_{t_i} = \frac{\sigma_{t_i}}{\sigma_{t_{i-1}}}\tilde{\mathbf{x}}_{t_{i-1}} + \sigma_{t_i}\sum_{n=0}^{k-1}\mathbf{x}_\theta^{(n)}(\hat{\mathbf{x}}_{\lambda_{t_{i-1}}}, \lambda_{t_{i-1}})\int_{\lambda_{t_{i-1}}}^{\lambda_{t_i}} e^{-\lambda}\frac{(\lambda-\lambda_{t_{i-1}})^n}{n!}\,\mathrm{d}\lambda + \mathcal{O}(h_i^{k+1}) \quad (10)$$

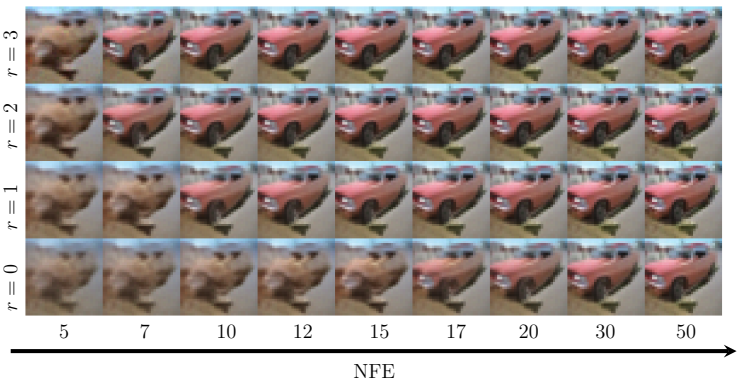


Figure 3: By increasing the number of extrapolation r for sampling, the solver can generate better results under the same NFE. Figure from Zhang and Chen(22).

	DEIS (Zhang & Chen, 2023 (22))	**DPM-Solver** (Lu et al., 2022 (21))	**DPM-Solver++** (Lu et al., 2023 (23))	**SDE-DPM-Solver++** (Lu et al., 2023 (23))
First-Order	DDIM ($\eta = 0$)	DDIM ($\eta = 0$)	DDIM ($\eta = 0$)	DDIM ($\eta = \sigma_t\sqrt{1-e^{-2h}}$)
Model Type	$\boldsymbol{\epsilon}_\theta$	$\boldsymbol{\epsilon}_\theta$	$\mathbf{x}_\theta$	$\mathbf{x}_\theta$
Taylor Expansion	$\boldsymbol{\epsilon}_\theta$ for t	$\hat{\boldsymbol{\epsilon}}_\theta$ for λ	$\hat{\mathbf{x}}_\theta$ for λ	$\hat{\mathbf{x}}_\theta$ for λ
Solver Type (High-Order)	Multistep	Singlestep	Singlestep + Multistep	Multistep

Table 1: Comparison of samplers and their characteristics. Table from Lu et al.(23)

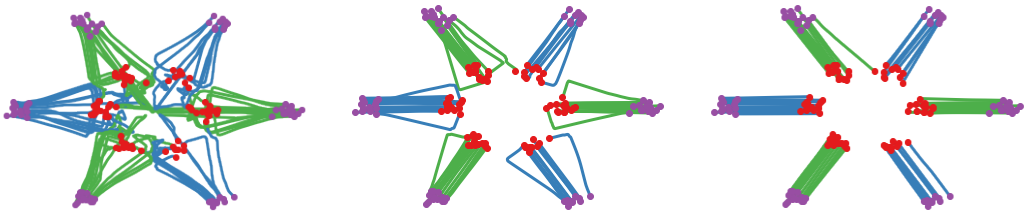
The recent contribution by Liu et al., entitled "Rectified Flow", has been recognised as a straightforward and effective methodology for solving ODE given two distributions π_0, π_1. The rectified flow

technique, based on flow matching that combines continuous normalizing flows(24) and diffusion model, involves learning the ODE that adheres to the flow $\boldsymbol{Z}$ between points X_0, X_1 sampled from distributions π_0 and π_1, focusing with the training objective of the parameters of the velocity prediction model $\hat{\theta}$ to minimize the expectation of transportation costs between X_0 and X_1 shown in Equation 11, and later samples the rectified flow pair (Z_0, Z_1) based on Equation 12:

$$\hat{\theta} = \arg\min_{\theta} \mathbb{E}\left[\|X_1 - X_0 - \mathbf{v}\left(tX_1 + (1-t)X_0, t\right)\|^2\right], \quad \text{with } t \sim \text{Uniform}([0,1]). \tag{11}$$

$$\mathrm{d}Z_t = \mathbf{v}_{\hat{\theta}}(Z_t, t)\mathrm{d}t, \quad \text{starting from } Z_0 \sim \pi_0 \text{ or } Z_1 \sim \pi_1 \tag{12}$$

This method represents the shortest trajectory between two distributions and enables sampling without the need for discretization of time, thus enhancing computational efficiency (25). Their research demonstrated that through the recursive application of rectified flow, the method can make straighter trajectories, which in turn optimizes the sampling process, as illustrated in Figure 4. The studies by Albergo and Vanden-Eijnden (26), as well as by Lipman et al. (27), share analogous objectives and results with those of Liu et al..



(a) 1-Rectified Flow
$\boldsymbol{Z}^1 = \text{RectFlow}((X_0, X_1))$

(b) 2-Rectified Flow
$\boldsymbol{Z}^2 = \text{RectFlow}((Z_0^1, Z_1^1))$

(c) 3-Rectified Flow
$\boldsymbol{Z}^3 = \text{RectFlow}((Z_0^2, Z_1^2))$

Figure 4: More iterations of rectified flow make straighter paths. (a): Reflow with single iterarion. (b): Reflow with two iterations. (c): Reflow with three iterations. Figure from Liu et al.(25).

Through the integration of rectified flow and an ODE solver, Esser et al.(28) amalgamated the flow matching method with multimodal DiT, and introduced logit-normal sampling, mode sampling and cosine schedule for sampling timesteps during the training phase, and using simple Euler ODE solver in the inference phase. Based on this work, Xie et al. proposed Flow-DPM-Solver++ applying DPM-Solver++ with transformation over data prediction with flow matching, and performed an experiment to compare with the work of Esser et al.(16), by introducing the velocity prediction model $\mathbf{v}_\theta$ and a hyperparameter named the timestep shift factor s. Flow-DPM-Sovler++ has shown faster convergence with $14 \sim 20$ steps, compared with the Flow-Euler method by Esser et al. which requires $28 \sim 50$ steps and even gives a worse result.

3.2.2 Distillation

Knowledge distillation is a training framework first proposed by Hinton et al.(29) to train a smaller and more efficient distilled "student" model from a larger, slow "teacher" model, and ensure the student model achieves similar performance compared with the teacher model. Luhman and Luhman simply transfer the knowledge distillation methodology to the LDMs(30), performs knowledge distillation for DDIM teacher model for training one-step sampling student model. Salimans and Ho(31) proposed progressive distillation, by distilling multiple steps into single steps progressively to reduce the number of steps with constant model size: Given the parameters of the teacher model η [1], the data perturbed with the noise scheduled at timestep t: $\mathbf{z}_t$ and the numbers of student sampling steps N, the student model first initialized its parameters θ to match up with the teacher's ($\theta \leftarrow \eta$). Then, by sampling two DDIM steps using the teacher model, the sampled data $\tilde{\mathbf{x}}$ then applied as the target and train the student model with the objective $w(\lambda_t)||\tilde{\mathbf{x}} - \hat{\mathbf{x}}_\theta(\mathbf{z}_t)||_2^2$, where $w(\lambda_t)$is the reconstruction loss weight defined as $w(\lambda_t) = \exp(\lambda_t)$. Then this process can be iteratively applied by promoting the student model to the teacher model($\eta \leftarrow \theta$), as shown in Figure 5. Following this research direction, Li et al. proposed CFG-Aware step distillation, which adapts the classifier-free

[1] In Section 3.2.2, η is defined as the parameters of the teacher model, where the previous parts η is the parameter for generalized DDIM for samplers.

guidance to distill with a combination of unconditional and conditional outputs. They also improved the training objective by adding the distillation loss with a weighted denoising loss.

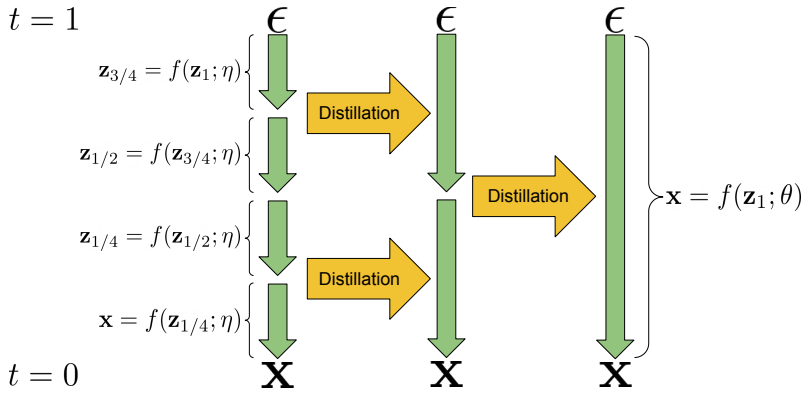


Figure 5: Two-iteration progressive distillation by distilling four Euler steps into a single Euler step. Figure from Salimans and Ho(31).

Inspired by the score distillation sampling approach from DreamFusion proposed by Poole et al.(32)(also known as score jacobian chaining from the work by Wang et al.(33)) which is based on probability density distillation by omitting the jacobian term of the gradient of the diffusion loss from training for an efficient gradient, Sauer et al proposed Adversarial Distillation Diffusion(ADD)(34)by combining adversarial objective $\mathcal{L}^{\text{G}}_{\text{adv}}(\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \theta)$ and the diffusion loss $\mathcal{L}_{\text{distill}}(\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \psi)$, into the overall objective as:

$$\mathcal{L} = \mathcal{L}^{\text{G}}_{\text{adv}}(\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \phi) + \lambda \mathcal{L}_{\text{distill}}(\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \psi) \tag{13}$$

where λ [2] is the hyperparameter for weighting the distillation loss, $\mathbf{x}_s$ is the noisy data by applying forward process from the training data, $\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s)$ is the generated image by the student model with $\mathbf{x}_s$ as the initial state, ψ denotes the parameters of the teacher model and ϕ is the parameter of the discriminator head. The adversarial loss $\mathcal{L}^{\text{G}}_{\text{adv}}(\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \phi)$ and the loss $\mathcal{L}^{\text{D}}_{\text{adv}}((\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \phi)$ for a set of discriminators $\mathcal{D}_{\phi,k}$ are:

$$\mathcal{L}^{\text{G}}_{\text{adv}}(\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \phi) = -\mathbb{E}_{s,\psi,\mathbf{x}_0}\left[\sum_k \mathcal{D}_{\phi,k}(F_k(\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s)))\right] \tag{14}$$

$$\begin{aligned}\mathcal{L}^{\text{D}}_{\text{adv}}((\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \phi) = \mathbb{E}_{\mathbf{x}_0}\left[\sum_k \max(0, 1 - \mathcal{D}_{\phi,k}(F_k(\mathbf{x}_0)) + \gamma R1(\phi)\right] \\ + \mathbb{E}_{\hat{\mathbf{x}}_\theta}\left[\sum_k \max(0, 1 + \mathcal{D}_{\phi,k}(F_k(\hat{\mathbf{x}}_\theta)))\right]\end{aligned} \tag{15}$$

where F_k is a pretrained feature network and R1 is the R1 gradient penalty(35) with a hyperparameter γ. The distillation loss $\mathcal{L}_{\text{distill}}(\hat{\mathbf{x}}_\theta(\mathbf{x}_s, s), \psi) = \mathbb{E}_{t,\epsilon'}\left[c(t)d(\hat{\mathbf{x}}_\theta, \hat{\mathbf{x}}_\psi(\text{stopgrad}(\hat{\mathbf{x}}_{\theta,t}); t)\right]$ is controlled with a stopgrad(.) operator to avoid updating the weights of the teacher model. This distillation loss is considered a special case of the score distillation diffusion loss by setting $d(x, y) = ||x - y||_2^2$ and choosing a specific choice for $c(t)$. Figure 6 depicts the whole training process, and note that the pixel space variables are used for calculating the losses instead of the latent space variables, as Yao et al. in their work of ARTIC3D(36) have shown the latent variables backpropagated noisy gradients through the encoder, where using pixel-space variables can mitigate this problem. This novel adversarial approach combined with score distillation has given great performances and only requires 1-4 sampling steps for high-fidelity images, which outperforms the DPM-Solver, which requires more than 10 steps, and progressive distillation with worse metrics results under the same and more numbers of sampling steps, as shown in Table 2.

[2] In Section 3.2.2, λ is the weighting factor for the distillation loss, where in the previous sections λ is the half of log SNR.

Distillation methods	Number of steps	latency(s)	FID ↓	CLIP ↑
DPM-Solver(21)	25	0.88	20.1	0.318
	8	0.34	31.7	0.320
Progressive Distillation(31)	1	0.09	37.2	0.275
	2	0.13	26.0	0.297
	4	0.21	26.4	0.300
CFG-Aware Distillation(37)	8	0.34	24.2	0.300
Adversarial Distillation Diffusion(34)	1	0.09	19.7	0.326

Table 2: The comparisons between distillation methods with DPM-Solver. ↑ denotes the higher score the better performance, and vice versa for ↓. Table from Sauer et al. (34).

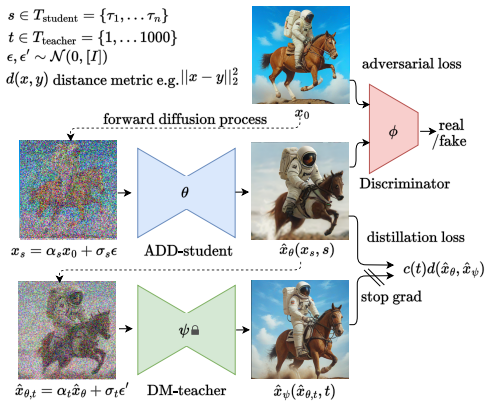


Figure 6: Adverarial Diffusion Distillation. Figure from Sauer et al.(34).

4 Summary & Conclusion

We have discussed the topics related to designing efficient text-conditioned LDMs, first give a general background of the text-conditioned LDMs, from the original pixel-space forward pass and reverse pass, to the state-of-the-art backbones and samplers with their efficiency problems. Then we reviewed three main topics: Efficient backbones for noise prediction with linearized attention, the novel SSM approach with the adaptation of architecture design; The faster sampling approach by designing faster training-free samplers from generalized DDIM, with a simulation-free training objective with flow matching to minimize the expectation of the costs between the sampled points from two distributions and speed up the sampling; The distillation methods with multiple variants to reduce the number of steps needed for image synthesis.

For potential future work topics, we have organized four main areas, with an additional topic that is not discussed but a novel and powerful approach:

1. Further improvement on the backbone architecture: Design efficient noise-predicting backbone based on linearized attention, SSMs, or propose novel efficient architectures that outperform current architectures, such as based on U-DiT architecture by Tian et al.(38) combining U-Net and DiT but better performance.
2. Faster training-free samplers: Based on the state-of-the-art higher-order samplers, find faster samplers that can sample high-fidelity images in fewer steps.
3. Distillation methods with better performance: Seek distillation methods that can beat the current adversarial distillation approach, aiming to synthesize better quality images under 1 step of sampling.
4. Enhancement of autoencoder architecture: Recent research conducted by Chen et al. has provided significant insights into advancing autoencoder design(39), as they discovered that the variational autoencoder used by stable diffusion models are hard to optimize and return low-quality reconstructed images when increasing the ratio of the spatial compression. They proposed deep compression autoencoder by integrating residual connection and

Decoupled High-Resolution Adaptation to the autoencoder training to address this drawback and have shown unaffected reconstruction results with higher spatial compression ratio, as well as faster inference speed when applied to text-conditioned LDMs.

REFERENCES

[1] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. *CoRR*, abs/2112.10752, 2021.

[2] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2023.

[3] Bingchen Liu, Ehsan Akhgari, Alexander Visheratin, Aleks Kamko, Linmiao Xu, Shivam Shrirao, Chase Lambert, Joao Souza, Suhail Doshi, and Daiqing Li. Playground v3: Improving text-to-image alignment with deep-fusion large language models, 2024.

[4] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020.

[5] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *CoRR*, abs/2010.02502, 2020.

[6] Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. *CoRR*, abs/1503.03585, 2015.

[7] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *CoRR*, abs/2006.11239, 2020.

[8] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[9] Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis, 2021.

[10] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015.

[11] William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023.

[12] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *CoRR*, abs/2010.11929, 2020.

[13] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[14] Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *CoRR*, abs/2011.13456, 2020.

[15] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention, 2020.

[16] Enze Xie, Junsong Chen, Junyu Chen, Han Cai, Haotian Tang, Yujun Lin, Zhekai Zhang, Muyang Li, Ligeng Zhu, Yao Lu, and Song Han. Sana: Efficient high-resolution image synthesis with linear diffusion transformers, 2024.

[17] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, José M. Álvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. *CoRR*, abs/2105.15203, 2021.

[18] Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022.

[19] Jing Nathan Yan, Jiatao Gu, and Alexander M. Rush. Diffusion models without attention, 2023.

[20] Songhua Liu, Weihao Yu, Zhenxiong Tan, and Xinchao Wang. Linfusion: 1 gpu, 1 minute, 16k image, 2024.

[21] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps, 2022.

[22] Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator, 2023.

[23] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models, 2023.

[24] Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David Duvenaud. Neural ordinary differential equations, 2019.

[25] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022.

[26] Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants, 2023.

[27] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023.

[28] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, Dustin Podell, Tim Dockhorn, Zion English, Kyle Lacey, Alex Goodwin, Yannik Marek, and Robin Rombach. Scaling rectified flow transformers for high-resolution image synthesis, 2024.

[29] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.

[30] Eric Luhman and Troy Luhman. Knowledge distillation in iterative generative models for improved sampling speed, 2021.

[31] Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. *CoRR*, abs/2202.00512, 2022.

[32] Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. Dreamfusion: Text-to-3d using 2d diffusion, 2022.

[33] Haochen Wang, Xiaodan Du, Jiahao Li, Raymond A. Yeh, and Greg Shakhnarovich. Score jacobian chaining: Lifting pretrained 2d diffusion models for 3d generation, 2022.

[34] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation, 2023.

[35] Lars Mescheder, Andreas Geiger, and Sebastian Nowozin. Which training methods for gans do actually converge?, 2018.

[36] Chun-Han Yao, Amit Raj, Wei-Chih Hung, Yuanzhen Li, Michael Rubinstein, Ming-Hsuan Yang, and Varun Jampani. Artic3d: Learning robust articulated 3d shapes from noisy web image collections, 2023.

[37] Yanyu Li, Huan Wang, Qing Jin, Ju Hu, Pavlo Chemerys, Yun Fu, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Snapfusion: Text-to-image diffusion model on mobile devices within two seconds, 2023.

[38] Yuchuan Tian, Zhijun Tu, Hanting Chen, Jie Hu, Chao Xu, and Yunhe Wang. U-dits: Downsample tokens in u-shaped diffusion transformers, 2024.

[39] Junyu Chen, Han Cai, Junsong Chen, Enze Xie, Shang Yang, Haotian Tang, Muyang Li, Yao Lu, and Song Han. Deep compression autoencoder for efficient high-resolution diffusion models, 2025.