RePiD: Efficient Recursive Pixel-Space Diffusion via Hierarchical Patch Denoising

Duoyou Chen, Yaxue Guo, Can Zhang, Cheng Chen[†]

University of Science and Technology Beijing

[†]Corresponding author

chencheng@ustb.edu.cn

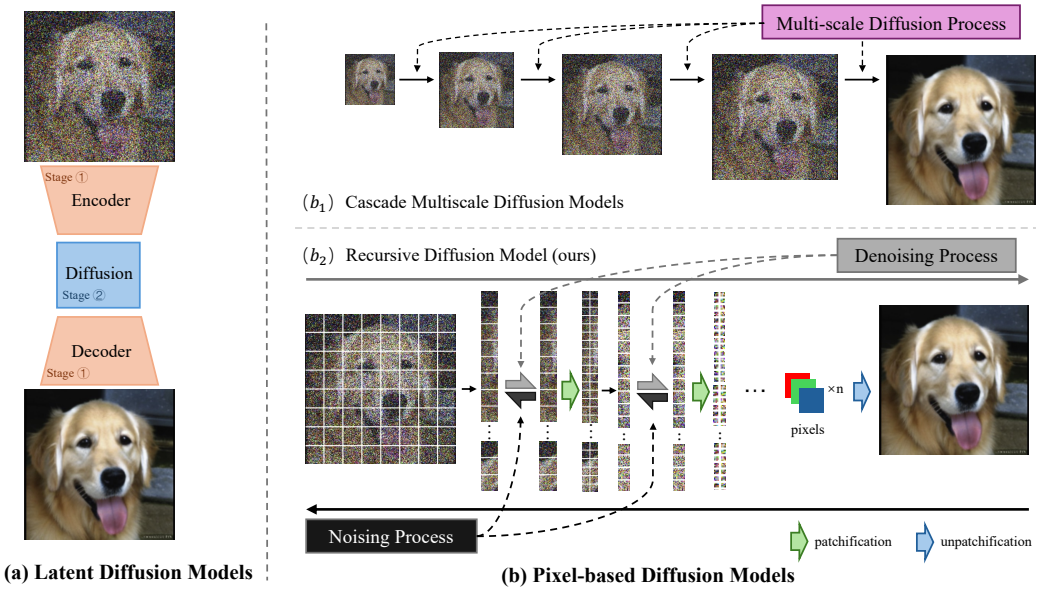

Figure 1: Comparison of Design Paradigms between latent diffusion models, pixel-based multi-scale diffusion models, and RePiD. (a) indicates the structure of latent diffusion models. (b) shows current pixel-based paradigm which splits the task into different scales.

Abstract

Recent pixel-space generative models avoid the reconstruction bottleneck of latent diffusion, but direct denoising over high-dimensional pixel tokens remains computationally demanding. This paper studies a recursive alternative for efficient pixel-space diffusion. We propose RePiD, a hierarchical patch-level diffusion framework that recursively partitions an image and performs the core denoising computation on lower-dimensional pixel groups instead of repeatedly operating on full-resolution images. To preserve spatial coherence across patch-wise processing, RePiD introduces a neighborhood embedding module that conditions each patch on its surrounding regions. Experiments on class-to-image generation and unpaired image-to-image translation show that RePiD provides a favorable quality–efficiency trade-off for direct pixel-space generation. These results suggest that recursive patch-level denoising is a practical design direction for efficient pixel-space generative modeling.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

1 Introduction

In recent years, diffusion models have achieved groundbreaking advancements in multimodal generation tasks, particularly demonstrating exceptional performance in the domain of high-quality image generation. However, traditional diffusion models operate directly in the computationally intensive pixel space, resulting in high training and inference costs. To address this issue, latent diffusion models (LDM)[38] have been proposed and have rapidly become a mainstream paradigm in the field. The core concept of LDM is "compress first, then diffuse": it employs a pre-trained autoencoder to compress high-dimensional raw image data into a lower-dimensional and perceptually equivalent latent space[3, 53]. Subsequently, the diffusion process is conducted entirely within this compact latent space, yielding remarkable results, with Stable Diffusion being its most representative application. Nonetheless, this paradigm inevitably incurs information loss during the semantic compression process of the original data[4, 45, 49]. This inherent information bottleneck imposes limitations on the model's fidelity in generation and poses challenges for subsequent fine-tuning and editing, thus becoming a pressing research bottleneck that necessitates resolution in the field.

The limitations of the LDM paradigm have prompted us to consider whether it is possible to study an efficient pixel-space design that reduces the cost of direct denoising, which not only inherits the outstanding generation fidelity of diffusion models but also significantly reduces their exorbitant computational costs[52]. Recently, relevant research has begun to explore this direction, with implementation pathways largely encapsulated by the concept of "multiscale diffusion in pixel space"[1, 12, 18] (as illustrated in Figure 1-b1). However, existing implementation paths vary considerably. One mainstream approach employs a multi-stage cascading strategy: the model initially generates a low-resolution image, which is subsequently upsampled and utilized as a condition to guide the generation of a higher-resolution image in the next stage. Another approach adopts a single-stage unified architecture that achieves cascading multiscale upsampling within a single model[4], thereby circumventing the complexity inherent in coordinating independent networks across different stages in a multi-stage paradigm. Nevertheless, many multi-scale pixel-space approaches still allocate substantial computation to high-resolution stages, leaving room for more efficient pixel-space denoising designs.

To address the aforementioned challenges, we have designed and proposed a recursive diffusion model implemented at the pixel level, termed RePiD. Inspired by fractal generative models[29], RePiD recursively deconstructs images into hierarchical patches and performs the same diffusion process at each hierarchy. Unlike methods that rely on full-resolution diffusion, our core denoising steps are exclusively executed on these lower-dimensional image patches, reducing the computational burden of direct pixel-space generation associated with operating in high-resolution spaces. Moreover, in order to aggregate global context and maintain spatial consistency during patch processing, we introduce a neighborhood encoding mechanism: for each image patch, the model encodes information from its eight neighboring patches. We further employ a transformer architecture as the backbone network of the diffusion model[35], enabling efficient modeling and processing of the intricate, multi-semantic neighborhood relationships among these image patches.

To comprehensively evaluate the performance of our proposed RePiD model, we conducted a series of rigorous quantitative experiments. For class-to-image generation on ImageNet-256, RePiD achieves an FID of 1.87 and an Inception Score of 341.6, while requiring only 43.3 GFLOPs and 3.43s sampling time under our evaluation setting. We further adapt RePiD to unpaired image-to-image translation as a downstream case study, showing its potential applicability beyond class-conditional generation.

The primary contributions of our work can be summarized as follows:

- We introduce RePiD, a recursive pixel-space diffusion framework that decomposes image generation into hierarchical patch-level denoising steps. By shifting the core denoising computation to lower-dimensional pixel groups, RePiD reduces the reliance on repeated full-resolution denoising while preserving competitive sample quality.
- We design a neighborhood embedding module to improve spatial coherence under recursive patch-wise generation. The module conditions each patch on its surrounding regions, mitigating the loss of cross-patch context caused by independent patch-level denoising.
- We evaluate RePiD on class-to-image generation and further adapt it to unpaired image-to-image translation as a downstream case study. The results show that RePiD provides a

favorable quality–efficiency trade-off for direct pixel-space generation and can be extended beyond class-conditional settings.

2 Related Works

2.1 Multi-Scale Diffusion Models

While diffusion models demonstrate exceptional performance in image generation, the high computational cost associated with direct operations in pixel space remains a critical challenge that necessitates attention. The research community has primarily explored two avenues for addressing this issue through multi-scale generation paradigms.

Firstly, latent diffusion models (LDM)[38], exemplified by Stable Diffusion, represent the current mainstream paradigm. The core principle of LDM is "compress first, then diffuse"[3, 41, 53]: by employing a pre-trained autoencoder, high-dimensional pixel data is mapped into a lower-dimensional latent space that is perceptually equivalent, thereby confining the diffusion process to this space and achieving significant enhancements in computational efficiency. Early latent diffusion models predominantly utilized the UNet architecture, but the introduction of DiT[35] marked the first instance of using transformers as the backbone of these models[2]. However, this paradigm inherently constrains the information bandwidth during the semantic compression process, leading to an information bottleneck that ultimately affects the upper limits of generation fidelity.

To mitigate information loss within the latent space, recent works have revisited pixel-space generation through multi-scale and hierarchical designs[1, 4, 8, 12, 45]. Existing approaches include multi-stage cascading strategies, such as Cascaded Diffusion Models and Relay Diffusion[18, 45], as well as unified pixel-space architectures such as PixelFlow[4]. More recent end-to-end pixel diffusion models further improve direct pixel-space generation: JiT[27] applies full-image attention with x-prediction, PixelDiT[54] combines patch-level and pixel-level pathways within each denoising step, DeCo[31] decouples low- and high-frequency modeling, and PixelGen[32] adds perceptual supervision to x-prediction. These methods have substantially improved the feasibility of direct pixel-space generation. However, many pixel-space designs still allocate considerable computation to high-resolution stages. In contrast, RePiD studies whether the core denoising computation can be shifted to recursively partitioned lower-dimensional pixel groups while preserving spatial coherence through neighborhood conditioning.

2.2 Hierarchical Generative Models

Hierarchical modeling serves as an effective strategy for constructing complex data by generating it in a stepwise manner through multiple levels and granularities.[28, 36, 39, 44] Prior to the advent of diffusion models, this concept had already been applied within other generative model frameworks. For instance, in the domain of Variational Autoencoders (VAEs), VQ-VAE-2[37] employs multiple layers of VQ-VAE to separately encode and generate global structures and local details[3]. Similarly, in the Generative Adversarial Networks (GANs) field, Progressive Growing GAN (PGGAN)[13] achieves hierarchical generation from low resolution to high resolution through progressive training[46].

In recent years, beyond the models mentioned in Multi-Scale Diffusion Models, fractal generative models[29] have offered a new theoretical perspective on hierarchical structures. The crux of fractal structures lies in self-similarity, which denotes the sharing of analogous structures across different scales[14]. Such models are designed to share generative logic and parameters recursively or iteratively across multiple scales[29, 43]. This mechanism of parameter sharing enables models to efficiently capture the rich details of high-resolution images without significantly increasing complexity, thereby providing a potential solution to alleviate the high-cost dilemma associated with existing "full-resolution diffusion" processes.

Positioning of RePiD. The contribution of RePiD is not hierarchy itself, but a diffusion-specific recursive denoising mechanism. Cascaded models such as RDM[45] relay diffusion across increasing resolutions, whereas RePiD keeps the image resolution fixed and changes the block size within a single denoising trajectory. FractalMAR[29] recursively composes autoregressive generators, whereas RePiD assigns spatial granularities to diffusion stages and denoises all blocks in parallel. PixelDiT[54] combines patch and pixel pathways within each step, whereas RePiD aligns spatial granularity with diffusion time. RePiD thus couples diffusion time with spatial granularity, decomposes full-resolution

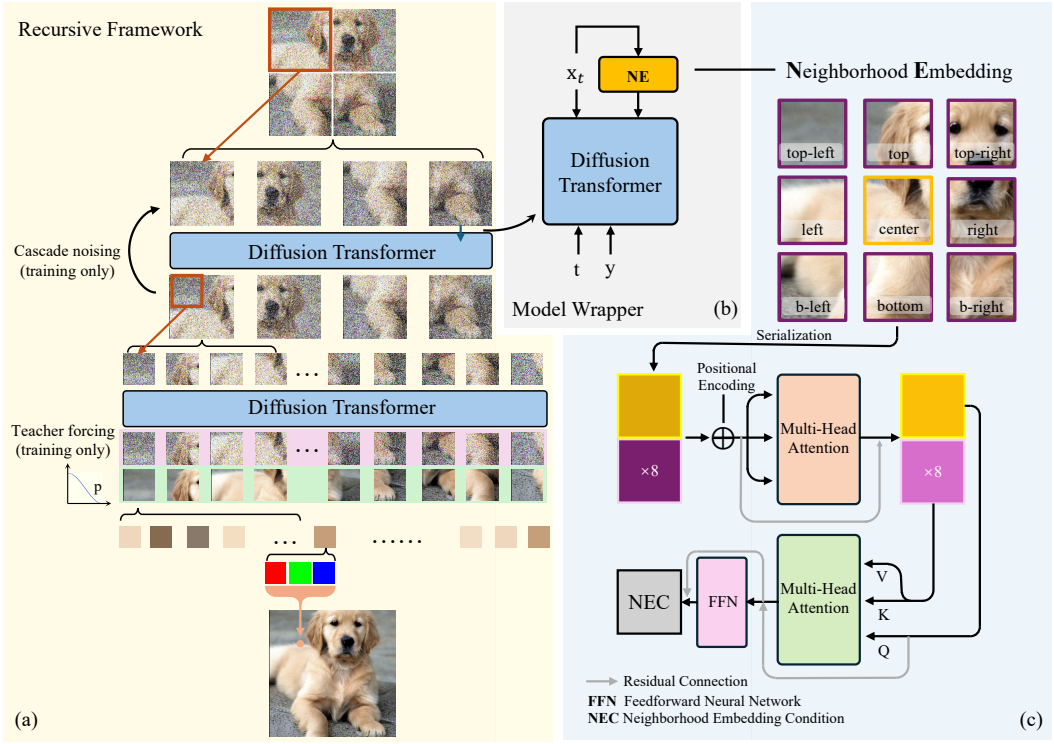

Figure 2: The diagram illustrates the structural framework of RePiD. (a) Depicts the specific recursive framework, which divides the diffusion process into several stages, wherein each stage segments the output from the previous stage and processes each segment independently. During the training process, the reverse phase of adding noise plays a crucial role within the recursive framework. (b) Shows the operational mode of the core diffusion model, where the model receives the current image, time t, condition y, and the neighborhood embedding information surrounding the current patch as inputs. (c) Demonstrates the specific implementation of neighborhood embedding, which is primarily divided into a self-attention mechanism in the first stage and a cross-attention mechanism in the second stage.

denoising into parallel block-wise processes, and restores cross-block communication through Neighborhood Embedding.

3 Methodology

3.1 Preliminary

3.1.1 Diffusion Models

The diffusion model introduces perturbations to the true data distribution $p(x_0)$ by incrementally adding Gaussian noise[22] ϵ, ultimately transforming the distribution into a simple standard Gaussian distribution.[17, 42] This discrete perturbation process adheres to the distribution $N(x_t|\alpha_t x_0, \sigma_t^2 I)$ at each time step t, where $\alpha_t, \sigma_t > 0$. This relationship can also be expressed through the following equation:

$$x_t = \alpha_t x_0 + \sigma_t \epsilon \tag{1}$$

In the reverse process, we predict the noise to be removed, denoted as $\hat{\epsilon}_t$, using the time index t, the noised image x_t, and the condition $\tau_\theta(y)$. This prediction is formulated as follows:

$$\hat{\epsilon}_t = \epsilon_\theta(x_t, t, \tau_\theta(y)) \tag{2}$$



Figure 3: The diagram illustrates several qualitative results of our model in Class-to-Image task using dataset ImageNet 256×256

The optimization objective of the diffusion process can be articulated as:

$$L_{diff} = \mathbb{E}_{t,x_0,\epsilon,c}[||\epsilon_t - \hat{\epsilon}_t||^2] \tag{3}$$

3.1.2 Diffusion Transformer

The Diffusion Transformer (DiT)[35] has been introduced into diffusion models, replacing the traditionally dominant UNet architecture. Empirical evidence demonstrates that, after sufficient training iterations, the performance of the Diffusion Transformer surpasses that of conventional UNet methods[10, 47, 48], even in the absence of long-range residual connections. The SiT[30] further substantiates the effectiveness of the Transformer architecture in linear flow diffusion settings.

$$x_t = x_t + \text{AdaLN}(y, t, \text{Attention}(x_t)) \tag{4}$$

$$x_t = x_t + \text{AdaLN}(y, t, \text{FFN}(x_t)) \tag{5}$$

Given a noisy image representation x_t, as indicated in Equation 1, along with the conditional input y and the time step t, the initial step involves segmenting x_t into non-overlapping image patches and converting these into a one-dimensional sequence. These noisy image patches are subsequently processed through stacked Self-Attention and Feed-Forward Network (FFN) modules, wherein the categorical labels are integrated through AdaLN modulation. After the blocks described in Equations 4 and 5, a linear projection decodes the features into noise estimates for the image patches. Throughout this paper, RePiD adopts the ϵ-prediction objective; a comparison with x- and v-prediction is given in Appendix A.7.

3.2 Hierarchical Recursive Diffusion Framework

As illustrated in Figure 2-(a), we partition the diffusion process into several stages, with the i-th stage segmenting the output of the previous stage into k_i patches. Consequently, the i-th stage will handle $K_i = \prod_{j \leq i} k_j$ patches of the original image, ensuring that these patches do not overlap. Arranging these patches sequentially reproduces the original image.

From a holistic perspective, let us assume a total of $N = K_n$ random variables exist. We define $Q_i = \frac{K_n}{K_i}$. During the first stage, these random variables are divided into k_1 subsets, each containing Q_1 random variables. We decompose $p(x_1, \ldots, x_{K_n})$ as follows: $p(x_1, \ldots, x_{K_n}) = \prod_{i=1}^{k_1} p(x_{(i-1)\cdot Q_1+1}, \ldots, x_{i\cdot Q_1} | x_1, \ldots, x_{(i-1)\cdot Q_1})$ Each conditional distribution $p(\cdot|\cdot)$, which consists of Q_1 random variables, is modeled by the Diffusion Transformer at this stage, and this

procedure continues recursively. By invoking such a divide-and-conquer approach, we aim to model the distribution of a total of K_n random variables from a probabilistic standpoint[9, 29].

In the context of image generation tasks, the random variables here refer to individual pixels, and we can systematically reduce noise from the original image through recursive staging. Conversely, during the training process, we are required to train in reverse order, commencing with the stage closest to the output image, that is, the last stage. Inter-stage dependencies necessitate the addition of noise to the images to satisfy the denoising objectives of each stage. However, as training across stages occurs synchronously, we employ a teacher forcing strategy[25, 51] to minimize inter-stage interference. Specifically, with probability p, the original image is used to train the current stage, while with probability $1 - p$, the cumulatively noised inter-stage state is employed for that stage's training. The probability follows a cosine decay, $p_{\mathrm{TF}}(e) = 0.5\,[1 + \cos(\pi e/E)]$, where e is the current epoch and E the total number of epochs (see Appendix A.6 for an ablation). Teacher forcing is used only during training. At inference, the block-wise ϵ estimates are assembled at every denoising step to update the global noisy image, which is directly repartitioned at each stage boundary.

The computational load of this process primarily hinges on the control parameters k_i. Assuming an image resolution of $W \times H$, the dominant computational effort associated with matrix operations of the attention mechanism can be represented as:

$$\sum_{i=1}^{n} K_i \left(\frac{W \cdot H}{K_i} \right)^2 = (W \cdot H)^2 \sum_{i=1}^{n} \frac{1}{K_i} \tag{6}$$

compared with $\mathcal{O}((W \cdot H)^2)$ for full-image attention. Here k_i denotes the number of children per parent block, and $K_i = \prod_{j=1}^{i} k_j$ the cumulative number of blocks. For a 256×256 image, we use recursive branching factors $\{k_1, k_2, k_3\} = \{256, 16, 16\}$. The three stages contain 256, 4,096, and 65,536 target blocks of size 16×16, 4×4, and 1×1, respectively, giving a ratio of $\sum_i 1/K_i = 273/65{,}536 \approx 1/240$. Equation 6 is an abstract scaling comparison of quadratic attention rather than an implementation-level FLOPs count; measured GFLOPs are reported in Section 4.

For each stage, as demonstrated in Figure 2-(b), we provide the current patch x_t, the time step t, and the condition y. To ensure smoothness among patches and enhance their interconnections, this process also involves neighborhood embedding of the patch x_t, which similarly acts as a condition. Blocks are not concatenated into one image-wide sequence: each forward evaluation of the stage-specific DiT processes a single target block, while the target block and its eight same-scale neighbors enter through the Neighborhood Embedding condition. Target blocks are processed in parallel batches. Each stage has its own DiT and Neighborhood Embedder, and reflection padding provides neighbors for blocks at the image border.

3.3 Neighborhood Embedding

As illustrated in Figure 2-(c), given a central block and its eight spatial neighbors, we construct a 3×3 neighbor grid denoted as $\mathcal{N} = \{p_i\}_{i=0}^{8}$, where p_4 represents the central block. Each block $p_i \in \mathbb{R}^{C \times H \times W}$ is flattened and embedded into a D-dimensional feature space. Learnable positional encodings are subsequently added to preserve spatial structural information, resulting in $\mathbf{E} = [\mathbf{E}_0, \mathbf{E}_1, \ldots, \mathbf{E}_8]^T \in \mathbb{R}^{9 \times D}$, where $\mathbf{E}_i = \mathbf{W}_e \mathbf{x}_i + \mathbf{p}_i$. The NeighborEmbedder employs a two-stage attention mechanism: initially, self-attention allows all neighbor blocks to interact with one another and establish a global contextual relationship; thereafter, the central block aggregates information from all enhanced neighbor blocks through cross-attention.

During the self-attention stage, all neighbor blocks interact via the query , key, and value mechanism:

$$\mathbf{Q}^{(1)} = \mathbf{E}\mathbf{W}_q^{(1)}, \quad \mathbf{K}^{(1)} = \mathbf{E}\mathbf{W}_k^{(1)}, \quad \mathbf{V}^{(1)} = \mathbf{E}\mathbf{W}_v^{(1)} \tag{7}$$

$$\mathbf{A}^{(1)} = \mathrm{softmax}\left(\frac{\mathbf{Q}^{(1)} (\mathbf{K}^{(1)})^T}{\sqrt{D}} \right), \quad \mathbf{E}^{(1)} = \mathbf{A}^{(1)} \mathbf{V}^{(1)} + \mathbf{E} \tag{8}$$

Table 1: Experimental results on Class-to-Image generation on ImageNet-256. The best results are shown in **bold**, and the second-best results are underlined. Recent pixel models use official results (DeCo at 50 steps) with latencies normalized to an RTX 4090; settings are not strictly matched (see Appendix A.4).

Model	Params	FID↓	IS↑	Precision↑	Recall↑	GFLOPs↓	Time/s↓
Latent Diffusion Models							
LDM-4-G [38]	86M+400M	3.60	247.7	**0.87**	0.48	**17.8**	3.46
DiT-XL/2 [35]	86M+675M	2.27	278.2	0.83	0.57	118.6	5.53
SiT-XL/2 [30]	86M+675M	2.06	277.5	0.83	0.59	118.6	5.03
Pixel Models							
ADM-G [8]	554M	4.59	186.7	0.82	0.52	156.9	13.24
RIN [20]	320M	3.42	182.0	–	–	64.8	4.64
RDM [45]	553M+553M	1.99	260.4	0.81	0.58	211.2	16.93
FractalMAR-H [29]	844M	6.15	**348.9**	0.81	0.46	283.5	22.31
PixelFlow-XL/4 [4]	677M	1.98	282.1	0.81	0.60	146.3	12.53
PixelDiT-XL/16 [54]	797M	**1.61**	292.7	–	**0.64**	155.5	23.90
JiT-H/16 [27]	953M	1.86	303.4	–	0.62	182.0	31.20
DeCo-XL/16 [31]	682M	1.69	304.0	–	0.63	130.5	23.20
PixelGen-XL/16 [32]	676M	1.83	293.6	–	0.63	129.0	17.80
RePiD (ours)	280M	1.87	341.6	0.83	0.63	43.3	**3.43**

Here, $\mathbf{A}_{ij}^{(1)}$ denotes the attention weight of neighbor block i concerning neighbor block j. In the cross-attention stage, the central block extracts information from all enhanced neighbor blocks:

$$\mathbf{q}_c = \mathbf{E}_4^{(1)}\mathbf{W}_q^{(2)}, \quad \mathbf{K}^{(2)} = \mathbf{E}^{(1)}\mathbf{W}_k^{(2)}, \quad \mathbf{V}^{(2)} = \mathbf{E}^{(1)}\mathbf{W}_v^{(2)} \tag{9}$$

$$\boldsymbol{\alpha} = \text{softmax}\left(\frac{\mathbf{q}_c(\mathbf{K}^{(2)})^T}{\sqrt{D}}\right) \in \mathbb{R}^{1\times 9}, \quad \mathbf{E}_c = \boldsymbol{\alpha}\mathbf{V}^{(2)} + \mathbf{E}_4^{(1)} \tag{10}$$

The final output is obtained through a feed-forward network yielding an enhanced representation of the central block, denoted as $\mathbf{E}_c^{out} = \text{FFN}(\mathbf{E}_c)$.

This two-stage design enables the model to simultaneously leverage spatial structural information and semantic relevance. Initially, allowing the neighbor blocks to interact with one another facilitates the establishment of global contextual relationships, resulting in richer representations. Subsequently, the central block aggregates information from all enhanced neighbor blocks, thereby achieving information fusion from local to global scales, which in turn improves the modeling of the complex relationships between image blocks and their spatial neighbors.

4 Experiments & Discussion

4.1 Dataset & Metrics

In the Class-to-Image task, we tested and evaluated our method on the ImageNet-256×256[7] dataset, recording metrics such as FID[15], Inception Score (IS)[40], Precision, and Recall[23]. For the Image-to-Image task, we evaluate RePiD on two unpaired translation datasets: CelebA-HQ[21] and AFHQ[6]. CelebA-HQ contains high-quality human face images and is used to evaluate face-domain translation, while AFHQ consists of animal face images from three domains, including cats, dogs, and wildlife. These two datasets cover both human-centered and animal-centered translation scenarios, allowing us to examine whether RePiD can be adapted to different visual domains. We report FID[15], Peak Signal-to-Noise Ratio (PSNR), and Structural Similarity Index Measure (SSIM)[50] for evaluation. All images are resized to 256×256 pixels.

4.2 Implementation Details & Settings

In both tasks, the main structure of the model is divided into three stages, with target block sizes of 16, 4, and 1, respectively. The patch sizes for the Diffusion Transformer are 4, 2, and 1, while the

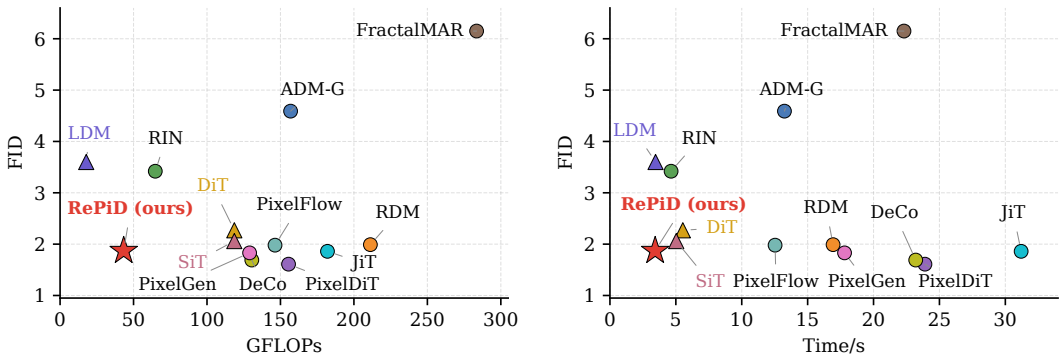

Figure 4: Quality–efficiency trade-off on ImageNet-256. Triangles denote latent-space models, circles denote pixel-space models, and red stars denote RePiD. RePiD achieves a favorable operating point with competitive FID, low GFLOPs, and short sampling time

time sequences are allocated as 400, 300, and 300, resulting in a total time sequence length of 1000. For the Class-to-Image Generation task, during the inference phase, we utilized the Euler-50 solver, whose 50 steps are allocated as 20/15/15 across the three stages, with the optimal classifier-free guidance (CFG)[16] value set at 3.4 and the interval-guidance[24] range set to [0.1, 1]. In the task of unpaired Image-to-Image Translation, we have incorporated an additional recurrent generator structure with the aim of providing pseudo-conditional inputs to our generative model. Specifically, the core of this recurrent generator is based on GAN[11, 56], and its output is utilized as conditional input for the RePiD model to facilitate a pairing effect from unpaired data. The recurrent generator and RePiD are trained synchronously, employing the same teacher forcing strategy as RePiD itself for multi-stage learning.

During training, we employed AdamW with an initial learning rate of 3.0e-5, accompanied by a step size of 50 and a gamma value of 0.8 for learning rate decay, on a single NVIDIA RTX 4090 with 24GB of memory. Training the 400K-step models takes 91.2 GPU-hours on ImageNet-256 and 146.4 GPU-hours on ImageNet-512; the full training configuration and cost are reported in Appendix A.3. Note that the GFLOPs and sampling times reported below measure inference efficiency rather than training cost.

4.3 Class-to-Image Generation

In the Class-to-Image Generation task, we compare RePiD with strong latent-based diffusion models, including Latent Diffusion Model (LDM)[38], DiT[35], and SiT[30], as well as pixel-based models including ADM[8], RIN[20], RDM[45], FractalMAR-H[29], and PixelFlow[4]. The qualitative results in Figure 3 show that RePiD produces visually coherent samples with fine-grained details. As reported in Table 1, RePiD achieves an FID of 1.87 and an IS of 341.6 on ImageNet-256, while requiring only 43.3 GFLOPs and 3.43s sampling time. Compared with PixelFlow, RePiD reduces the reported GFLOPs from 146.3 to 43.3 and the sampling time from 12.53s to 3.43s, while maintaining comparable generation quality. We further compare with recent end-to-end pixel diffusion models, namely PixelDiT[54], JiT[27], DeCo[31], and PixelGen[32]. PixelDiT and DeCo achieve lower FID (1.61 and 1.69), while JiT and PixelGen reach FID close to RePiD at 50 steps (1.86 and 1.83). In comparison, RePiD uses 2.4–3.4$\times$ fewer parameters and 3.0–4.2$\times$ fewer GFLOPs than these methods and obtains the highest IS among them (341.6), with comparable recall. These results suggest that recursive patch-level denoising offers a practical quality–efficiency trade-off for direct pixel-space generation. Figure 4 further visualizes the quality–efficiency trade-off by plotting FID against GFLOPs and sampling time. RePiD lies in the low-cost region in both views, suggesting a favorable operating point for efficient pixel-space generation.

4.4 Image-to-Image Translation

We further evaluate RePiD on unpaired image-to-image translation as a downstream case study. We compare with representative unpaired translation methods, including CycleGAN [56], CUT [34], ILVR [5], QS-Attn [19], EGSDE [55], and Injecting Diffusion [26]. As shown in Table 2, RePiD

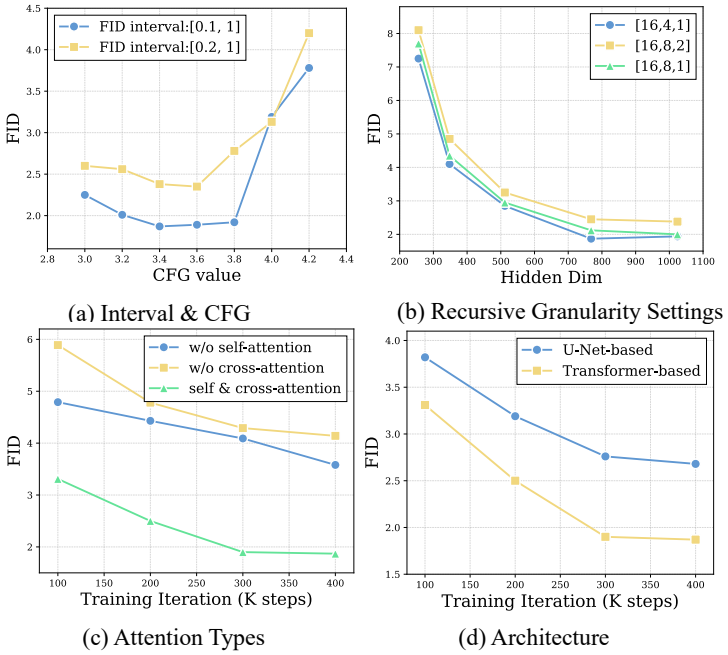

Figure 5: The chart presents the results of ablation experiments focusing on four different components. (a) Illustrates the variation in result quality associated with different interval and CFG values; (b) demonstrates the impact of varying recursive granularity and hidden dimension on result quality; (c) shows the results of ablation studies on the types of attention utilized within Neighborhood Embedding; (d) depicts the influence of using different diffusion model architectures on result quality.

obtains competitive FID and SSIM on both CelebA-HQ and AFHQ, suggesting that recursive patch-level denoising can be adapted to pseudo-conditional generation settings. We note that PSNR and SSIM can sometimes favor methods that preserve the input image without fully translating it to the target domain; therefore, we mainly use this experiment to demonstrate the transferability of RePiD rather than to make a broad claim of dominance in unpaired translation. Because the recurrent generator supplies the pseudo-conditions, we regard this setting as a system-level case study. Compared with the intermediate output of the recurrent generator from the same checkpoint, the final RePiD output improves FID from 31.23 to 26.30 on CelebA-HQ and from 103.64 to 59.37 on AFHQ, with higher PSNR and SSIM (Appendix A.8). Since the two components are trained jointly and may co-adapt, this gain indicates refinement by RePiD but does not isolate its causal contribution.

Table 2: Experimental results on Image-to-Image translation. The best results are shown in **bold**, and the second-best results are underlined.

	CelebA-HQ			AFHQ		
Method	**FID↓**	**PSNR↑**	**SSIM↑**	**FID↓**	**PSNR↑**	**SSIM↑**
CycleGAN [56]	36.75	21.54	0.70	85.90	18.13	0.58
CUT [34]	31.94	19.87	0.74	76.21	17.48	0.60
QS-Attn [19]	32.56	20.68	0.60	72.80	18.26	0.39
ILVR [5]	46.12	18.59	0.51	74.37	17.77	0.36
EGSDE [55]	30.61	20.35	0.64	65.82	19.31	0.41
SDEdit [33]	49.43	20.03	0.67	74.17	19.19	0.42
Injecting Diffusion [26]	28.12	**21.65**	0.67	63.76	20.10	0.48
RePiD (ours)	**26.30**	20.98	**0.77**	**59.37**	**21.32**	**0.63**

Table 3: Controlled ablation of stage design and neighborhood aggregation on ImageNet-256 (200K iterations, 50-step Euler sampler).

Stages	Neighborhood aggregation	FID↓	IS↑	Recall↑
Single	3×3 convolution	5.30	175.3	0.42
Single	Neighborhood Attention	3.58	203.2	0.54
Multi	3×3 convolution	3.32	243.6	0.59
Multi	Neighborhood Attention (RePiD)	**2.50**	**301.9**	**0.61**

4.5 Ablation Study

We conduct ablation studies on ImageNet-256 to examine the main design choices of RePiD, as summarized in Figure 5. First, Figure 5(a) shows that a moderate CFG value with interval guidance gives the best sample quality, while overly large guidance degrades FID. Second, Figure 5(b) studies the recursive granularity and hidden dimension. The results suggest that an intermediate recursive configuration provides a better balance between representation capacity and optimization difficulty. Third, Figure 5(c) verifies the importance of neighborhood embedding: combining self-attention among neighboring patches with cross-attention to the center patch achieves the best FID, supporting the need for explicit cross-patch context modeling. Finally, Figure 5(d) compares U-Net and Transformer backbones under the same recursive framework. The Transformer backbone obtains better final performance, indicating that token-level interaction is beneficial for recursively partitioned pixel groups.

Recursion and neighborhood attention. To isolate the two core components, we conduct a controlled 2×2 ablation that varies only the stage design and the neighborhood operator (Table 3). The single-stage variant uses 16×16 blocks for all 50 Euler steps, whereas the multi-stage variant uses $16 \times 16 \rightarrow 4 \times 4 \rightarrow 1 \times 1$ blocks with 20/15/15 steps. The convolutional baseline replaces Neighborhood Attention with a 3×3 convolution over the same nine block embeddings. Multi-stage recursion consistently improves over a single stage, and multi-stage with convolution even outperforms single-stage with attention, isolating the benefit of recursion. Attention, which adapts its aggregation to content and noise level, further improves over convolution in both settings, and their combination performs best.

Cross-patch consistency. Since Neighborhood Embedding is local, distant blocks communicate hop by hop. Evaluating $\hat{x}_0$ at each stage boundary, FID/sFID decrease monotonically from 6.32/18.96 to 1.87/4.55, and a Boundary Score for grid artifacts deviates from real-image statistics by only 3.31%/1.85% (Appendix A.5). A 5×5 neighborhood improves FID from 2.50 to 2.21 at $2.88\times$ GFLOPs, so we keep 3×3 for efficiency.

5 Conclusion

In this paper, we presented RePiD, a recursive pixel-space diffusion framework for efficient image generation. Instead of repeatedly performing denoising over full-resolution images, RePiD decomposes the generation process into hierarchical patch-level denoising steps, where each stage operates on lower-dimensional pixel groups. To mitigate the loss of spatial context caused by patch-wise processing, we introduced a neighborhood embedding module that conditions each patch on its surrounding regions. Experiments on class-to-image generation and unpaired image-to-image translation demonstrate that RePiD offers a favorable quality–efficiency trade-off compared with representative pixel-space diffusion baselines.

RePiD still has several limitations. Recursive patch-level denoising may introduce cross-patch consistency challenges, especially when modeling scenes with strong global structures; local neighborhood propagation mitigates, rather than eliminates, this limitation. In addition, although the ImageNet-512 results in the supplementary material suggest that RePiD can be extended to higher-resolution generation, there remains room to further improve high-resolution fidelity. Scalability to large-scale text-to-image generation also remains unverified. Future work will investigate adaptive stage allocation, stronger cross-patch context modeling, and text-to-image evaluation on GenEval and DPG-Bench.

Acknowledgments and Disclosure of Funding

This work was supported by the National Natural Science Foundation of China (62301026, 62576359). The authors declare no competing interests.

References

[1] Yuval Atzmon, Maciej Bala, Yogesh Balaji, Tiffany Cai, Yin Cui, Jiaojiao Fan, Yunhao Ge, Siddharth Gururani, Jacob Huffman, Ronald Isaac, et al. Edify image: High-quality image generation with pixel space laplacian diffusion models. *arXiv preprint arXiv:2411.07126*, 2024.

[2] Fan Bao, Shen Nie, Kaiwen Xue, Yue Cao, Chongxuan Li, Hang Su, and Jun Zhu. All are worth words: A vit backbone for diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 22669–22679, 2023.

[3] 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. *arXiv preprint arXiv:2410.10733*, 2024.

[4] Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025.

[5] Jooyoung Choi, Sungwon Kim, Yonghyun Jeong, Youngjune Gwon, and Sungroh Yoon. Ilvr: Conditioning method for denoising diffusion probabilistic models. *arXiv preprint arXiv:2108.02938*, 2021.

[6] Yunjey Choi, Youngjung Uh, Jaejun Yoo, and Jung-Woo Ha. Stargan v2: Diverse image synthesis for multiple domains. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8188–8197, 2020.

[7] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[8] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. *Advances in neural information processing systems*, 34:8780–8794, 2021.

[9] Dhruba Ghosh, Hannaneh Hajishirzi, and Ludwig Schmidt. Geneval: An object-focused framework for evaluating text-to-image alignment. *Advances in Neural Information Processing Systems*, 36:52132–52152, 2023.

[10] Lixue Gong, Xiaoxia Hou, Fanshi Li, Liang Li, Xiaochen Lian, Fei Liu, Liyang Liu, Wei Liu, Wei Lu, Yichun Shi, et al. Seedream 2.0: A native chinese-english bilingual image generation foundation model. *arXiv preprint arXiv:2503.07703*, 2025.

[11] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11): 139–144, 2020.

[12] Jiatao Gu, Shuangfei Zhai, Yizhe Zhang, Joshua M Susskind, and Navdeep Jaitly. Matryoshka diffusion models. In *The Twelfth International Conference on Learning Representations*, 2023.

[13] Changhee Han, Kohei Murao, Tomoyuki Noguchi, Yusuke Kawata, Fumiya Uchiyama, Leonardo Rundo, Hideki Nakayama, and Shin'ichi Satoh. Learning more with less: Conditional pggan-based data augmentation for brain metastases detection using highly-rough annotation on mr images. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 119–127, 2019.

[14] Jian Han, Jinlai Liu, Yi Jiang, Bin Yan, Yuqi Zhang, Zehuan Yuan, Bingyue Peng, and Xiaobing Liu. Infinity: Scaling bitwise autoregressive modeling for high-resolution image synthesis. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15733–15744, 2025.

[15] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

[16] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

[17] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

[18] Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23 (47):1–33, 2022.

[19] Xueqi Hu, Xinyue Zhou, Qiusheng Huang, Zhengyi Shi, Li Sun, and Qingli Li. Qs-attn: Query-selected attention for contrastive learning in i2i translation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 18291–18300, 2022.

[20] Allan Jabri, David Fleet, and Ting Chen. Scalable adaptive computation for iterative generation. *arXiv preprint arXiv:2212.11972*, 2022.

[21] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*, 2017.

[22] Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. *Advances in neural information processing systems*, 35:26565–26577, 2022.

[23] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved precision and recall metric for assessing generative models. *Advances in neural information processing systems*, 32, 2019.

[24] Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. *Advances in Neural Information Processing Systems*, 37:122458–122483, 2024.

[25] Alex M Lamb, Anirudh Goyal ALIAS PARTH GOYAL, Ying Zhang, Saizheng Zhang, Aaron C Courville, and Yoshua Bengio. Professor forcing: A new algorithm for training recurrent networks. *Advances in neural information processing systems*, 29, 2016.

[26] Luying Li and Lizhuang Ma. Injecting-diffusion: Inject domain-independent contents into diffusion models for unpaired image-to-image translation. In *2023 IEEE International Conference on Multimedia and Expo (ICME)*, pages 282–287. IEEE, 2023.

[27] Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[28] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems*, 37:56424–56445, 2024.

[29] Tianhong Li, Qinyi Sun, Lijie Fan, and Kaiming He. Fractal generative models. *arXiv preprint arXiv:2502.17437*, 2025.

[30] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *European Conference on Computer Vision*, pages 23–40. Springer, 2024.

[31] Zehong Ma, Longhui Wei, Shuai Wang, Shiliang Zhang, and Qi Tian. Deco: Frequency-decoupled pixel diffusion for end-to-end image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2026.

[32] Zehong Ma, Ruihan Xu, and Shiliang Zhang. Pixelgen: Improving pixel diffusion with perceptual supervision. *arXiv preprint arXiv:2602.02493*, 2026.

[33] Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. Sdedit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

[34] Taesung Park, Alexei A Efros, Richard Zhang, and Jun-Yan Zhu. Contrastive learning for unpaired image-to-image translation. In *European conference on computer vision*, pages 319–345. Springer, 2020.

[35] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023.

[36] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[37] Ali Razavi, Aaron Van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2. *Advances in neural information processing systems*, 32, 2019.

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10684–10695, 2022.

[39] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. *Advances in neural information processing systems*, 35:36479–36494, 2022.

[40] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

[41] A Sauer, K Schwarz, and A StyleGAN-XL Geiger. scaling stylegan to large diverse datasets. In *Proceedings of the SIGGRAPH Conference. ACM*, pages 1–10.

[42] Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

[43] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.

[44] Haotian Tang, Yecheng Wu, Shang Yang, Enze Xie, Junsong Chen, Junyu Chen, Zhuoyang Zhang, Han Cai, Yao Lu, and Song Han. Hart: Efficient visual generation with hybrid autoregressive transformer. *arXiv preprint arXiv:2410.10812*, 2024.

[45] Jiayan Teng, Wendi Zheng, Ming Ding, Wenyi Hong, Jianqiao Wangni, Zhuoyi Yang, and Jie Tang. Relay diffusion: Unifying diffusion process across resolutions for image synthesis. *arXiv preprint arXiv:2309.03350*, 2023.

[46] Ren Togo, Takahiro Ogawa, and Miki Haseyama. Synthetic gastritis image generation via loss function-based conditional pggan. *IEEE access*, 7:87448–87457, 2019.

[47] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[48] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[49] Michael Tschannen, André Susano Pinto, and Alexander Kolesnikov. Jetformer: An autoregressive generative model of raw images and text. *arXiv preprint arXiv:2411.19722*, 2024.

[50] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

[51] Ronald J Williams and David Zipser. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280, 1989.

[52] Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. *arXiv preprint arXiv:2408.12528*, 2024.

[53] Jingfeng Yao, Bin Yang, and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 15703–15712, 2025.

[54] Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. *arXiv preprint arXiv:2511.20645*, 2025.

[55] Min Zhao, Fan Bao, Chongxuan Li, and Jun Zhu. Egsde: Unpaired image-to-image translation via energy-guided stochastic differential equations. *Advances in Neural Information Processing Systems*, 35: 3609–3623, 2022.

[56] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 2223–2232, 2017.

A APPENDIX

A.1 Implementation Details

This section provides additional implementation details for RePiD. In the main paper, RePiD is implemented as a three-stage recursive pixel-space diffusion framework, where the stage resolutions are set to 16, 4, and 1, and the patch sizes of the Diffusion Transformer are set to 4, 2, and 1, respectively. The diffusion timesteps are allocated as 400, 300, and 300 across the three stages, resulting in a total of 1000 timesteps. During inference, we use the Euler-50 solver with classifier-free guidance (CFG) set to 3.4 and interval guidance range $[0.1, 1]$ for class-to-image generation.

For optimization, we use AdamW with an initial learning rate of 3.0×10^{-5}. The learning rate is decayed with a step size of 50 and a decay factor of 0.8. Unless otherwise specified, all experiments are conducted on NVIDIA RTX 4090 GPUs with 24GB memory. For Image-to-Image translation, we use an additional recurrent generator to provide pseudo-conditional inputs for RePiD. The recurrent generator and RePiD are trained jointly, following the same teacher-forcing strategy used in the recursive diffusion framework.

For evaluation, FID and IS are computed using 50K generated samples following the standard ImageNet evaluation protocol. GFLOPs are measured for the denoising backbone under the same input resolution and sampling configuration used in the corresponding experiment. Sampling time is reported under the same hardware environment to ensure a consistent efficiency comparison across methods.

A.2 Additional Results on ImageNet-512

We further evaluate RePiD on ImageNet-512 to examine the scalability of recursive patch-level denoising. Since ImageNet-512 contains a substantially larger pixel space than ImageNet-256, this setting provides a more challenging testbed for direct pixel-space generation. The compared baselines follow the model families discussed in the main paper, including latent-space diffusion models and pixel-space diffusion models.

Table 4 reports the quantitative results. RePiD achieves an FID of 3.16, an Inception Score of 287.13, a precision of 0.86, and a recall of 0.60. While SiT-XL/2 obtains the best FID, RePiD achieves the highest Inception Score and recall among the compared methods. This indicates that RePiD maintains strong semantic coverage and sample diversity at higher resolution.

In terms of efficiency, RePiD uses 167.6 GFLOPs and 10.32s sampling time under the reported setting, which is lower than the compared baselines. Therefore, the ImageNet-512 results further support the main observation that recursive patch-level denoising provides a favorable quality–efficiency trade-off for direct pixel-space generation. At the same time, the remaining FID gap to latent-space baselines suggests that improving high-resolution fidelity and cross-patch consistency is an important direction for future work.

Table 4: Additional class-to-image generation results on ImageNet-512. The best results are shown in **bold**, and the second-best results are underlined.

Model	Params	FID↓	IS↑	Precision↑	Recall↑	GFLOPs↓	Time/s↓
Latent Diffusion Models							
DiT-XL/2 [35]	675M+86M	<u>3.04</u>	240.82	0.84	0.54	469.5	20.41
SiT-XL/2 [30]	675M+86M	**2.62**	<u>252.21</u>	0.84	<u>0.57</u>	454.3	22.59
Pixel Diffusion Models							
ADM-G [8]	559M	7.72	172.71	**0.87**	0.42	635.2	32.56
RIN [20]	320M	3.95	210.00	–	–	<u>254.5</u>	<u>16.53</u>
RePiD (ours)	280M	3.16	**287.13**	<u>0.86</u>	**0.60**	**167.6**	**10.32**

A.3 Training Configuration and Cost

Table 5 reports the training configuration and cost of RePiD. Both models are trained on a single RTX 4090. For ImageNet-256, we use AdamW with a learning rate of 3×10^{-5}, StepLR with a

step size of 50 and a decay factor of 0.8, target block sizes $[16, 4, 1]$, DiT patch sizes $[4, 2, 1]$, and diffusion intervals $[400, 300, 300]$. The intervals partition the 1,000-step training timeline and map to 20/15/15 steps of the 50-step Euler sampler. When a stage contains more than 2,048 target blocks, we randomly sample 2,048 blocks per update.

Table 5: Training configuration and cost of RePiD.

Setting	**ImageNet-256**	**ImageNet-512**
Hardware	1× RTX 4090 (24GB)	1× RTX 4090 (24GB)
Global image batch size	4	2
Gradient accumulation	1	1
Precision	FP16 mixed	FP16 mixed
Activation checkpointing	Disabled	Disabled
Maximum target blocks per step	2,048	2,048
Training steps	400K	400K
Peak GPU memory	22GB	23GB
Wall-clock time	3.8 days	6.1 days
Total GPU-hours	91.2	146.4

A.4 Detailed Comparison with Recent Pixel Diffusion Models

Table 6 lists the sampling settings of the recent pixel diffusion models in Table 1, including both 50-step and best operating points when available. We count one multiply-add as one operation, and values reported with two FLOPs per multiply-add are divided by two. Latencies of other methods are normalized to an RTX 4090 by compute and number of solver evaluations. Training and guidance settings are not strictly matched.

Table 6: Detailed comparison with recent pixel diffusion models on ImageNet-256.

Method	**Setting**	**Sampler**	**Steps**	**Evals.**	**Params**	**GFLOPs↓**	**Lat.↓**	**FID↓**	**IS↑**
RePiD (ours)	50-step/official	Euler	50	50	280M	43.3	3.43s	1.87	341.6
PixelDiT-XL/16	official best	FlowDPMSolver	100	100	797M	155.5	23.9s	1.61	292.7
JiT-H/16	50-step/official	Heun	50	100	953M	182.0	31.2s	1.86	303.4
DeCo-XL/16	50-step	Heun	50	100	682M	130.5	23.2s	1.69	304.0
DeCo-XL/16	official best	Euler	250	250	682M	130.5	48.2s	1.62	301.0
PixelGen-XL/16	50-step/official	Heun	50	100	676M	129.0	17.8s	1.83	293.6

These methods differ from RePiD in how they organize pixel-space computation. PixelDiT models patch- and pixel-level features within each step, whereas RePiD changes block granularity across the trajectory. JiT uses full-image global attention and x-prediction, whereas RePiD uses block-wise ϵ-prediction with local conditioning; hence our single-stage ablation is not equivalent to JiT. DeCo separates low- and high-frequency modeling within each step, and PixelGen combines x-prediction with LPIPS and DINO supervision, whereas RePiD uses ϵ-prediction without perceptual losses. Overall, RePiD targets a favorable inference-time quality–compute trade-off, remaining competitive in FID with 280M parameters and 43.3 GFLOPs.

A.5 Cross-Patch Consistency Analysis

Boundary Score. We measure grid artifacts with adjacent-pixel L1 RGB differences. At scale s, D_B^s denotes the mean difference over pixel pairs straddling a recursive block boundary, and D_N^s the mean over matched same-orientation non-boundary pairs. We compute $R_s = D_B^s / D_N^s$ and define $B_s = 100 \times |R_s^{gen} / R_s^{real} - 1|$. Horizontal and vertical pairs are averaged over all grid lines and 50K images. B_s diagnoses grid artifacts rather than all forms of semantic inconsistency.

Stage-wise evaluation. Using the 400K model, we recover $\hat{x}_0$ from the noisy state and the predicted ϵ at each stage boundary by the standard conversion; these are diagnostics rather than independent sampling settings. The cumulative boundaries are at 20, 35, and 50 steps. As shown in Table 7, FID and sFID decrease monotonically, suggesting progressive refinement without obvious error amplification. With repeated 3×3 exchange, a block's theoretical dependency field grows to

$(2r+1)\times(2r+1)$ after r evaluations, so the 20 coarse evaluations already connect the entire 16×16 grid, although less directly than full-image attention.

Table 7: Stage-wise evaluation of the predicted clean image on ImageNet-256 (400K model).

Evaluated output	**Steps**	**FID↓**	**sFID↓**	B_{16} (%)↓	B_4 (%)↓
After 16×16-block stage	20	6.32	18.96	1.36	0.20
After 4×4-block stage	35	4.74	9.23	1.78	0.53
After pixel stage	50	1.87	4.55	3.31	1.85

Neighborhood size. We enlarge only the neighborhood from 3×3 to 5×5 (9 to 25 tokens), keeping the model dimensions, attention layers, 200K training budget, and 50-step Euler sampler fixed. As shown in Table 8, a wider neighborhood improves all metrics but increases GFLOPs by $2.88\times$. We retain 3×3 as the efficiency-oriented default. The 200K results here are not directly comparable to the 400K results in Table 7.

Table 8: Effect of neighborhood size on ImageNet-256 (200K iterations).

Neighborhood	**FID↓**	**sFID↓**	B_{16} (%)↓	B_4 (%)↓	**GFLOPs/eval.↓**
3×3 (default)	2.50	5.12	3.22	1.93	43.3
5×5	2.21	4.92	2.94	1.13	124.6

A.6 Teacher-Forcing Schedule

Table 9 compares teacher-forcing schedules at 200K iterations. Always using the clean source causes a train–inference mismatch, while never using it hinders early optimization. The default cosine decay balances both.

Table 9: Ablation of the teacher-forcing schedule on ImageNet-256 (200K iterations).

Schedule	**Start/end prob.**	**FID↓**	**sFID↓**	**Recall↑**
Always teacher forcing	$1\to1$	7.63	25.36	0.39
No teacher forcing	$0\to0$	5.32	19.57	0.48
Cosine decay (default)	$1\to0$	**2.50**	**5.12**	**0.61**

A.7 Prediction Objective

We compare ϵ-, x-, and v-prediction under the same timestep sampling, loss weighting, optimizer, 200K training budget, guidance setting, and 50-step Euler schedule, without target-specific tuning. During sampling, x- and v-parameterized outputs are converted to noise estimates before applying the same Euler update. As shown in Table 10, x-prediction slightly lowers FID but also lowers IS and recall, so the choice is metric-dependent. v-prediction underperforms under this fixed recipe; we do not interpret this as a general disadvantage of v-parameterization.

A.8 Attribution in Image-to-Image Translation

In unpaired translation, the recurrent generator supplies pseudo-conditions to RePiD. Table 11 compares the intermediate output of the recurrent generator with the final RePiD output from the same checkpoint. RePiD improves every metric on both datasets. Because the two components are trained jointly and may co-adapt, this comparison indicates refinement by RePiD but does not establish isolated causality.

A.9 Detailed Ablation Results

This section provides the detailed numerical results for the ablation studies summarized in the main paper. We examine four design factors: classifier-free guidance and interval guidance, recursive

Table 10: Comparison of prediction targets on ImageNet-256 (200K iterations).

Prediction target	**FID↓**	**IS↑**	**Recall↑**
ϵ-prediction (default)	2.50	**301.9**	**0.61**
x-prediction	**2.43**	293.8	0.53
v-prediction	3.54	214.3	0.49

Table 11: Intermediate recurrent-generator output versus final RePiD output.

	CelebA-HQ			**AFHQ**		
Evaluated output	**FID↓**	**PSNR↑**	**SSIM↑**	**FID↓**	**PSNR↑**	**SSIM↑**
Recurrent generator output	31.23	18.32	0.69	103.64	17.45	0.46
RePiD final output	**26.30**	**20.98**	**0.77**	**59.37**	**21.32**	**0.63**

granularity and hidden dimension, attention design in neighborhood embedding, and backbone architecture. All ablations are conducted on ImageNet-256, and FID is used as the evaluation metric.

Table 12: Detailed ablation results on ImageNet-256. The best result in each block is shown in **bold**.

(a) CFG and interval guidance

CFG value	**Interval [0.1, 1]**	**Interval [0.2, 1]**
3.0	2.25	2.60
3.2	2.01	2.56
3.4	**1.87**	2.38
3.6	1.89	2.35
3.8	1.92	2.78
4.0	3.19	3.13
4.2	3.78	4.20

(b) Recursive granularity and hidden dimension

Hidden dim	**Config. [16,4,1]**	**Config. [16,8,2]**	**Config. [16,8,1]**
256	7.25	8.10	7.70
348	4.10	4.85	4.35
512	2.85	3.25	2.95
768	**1.87**	2.45	2.12
1024	1.94	2.38	2.00

(c) Neighborhood embedding and backbone architecture

Setting	**100K**	**200K**	**300K**	**400K**
Neighborhood embedding				
w/o self-attention	4.79	4.43	4.09	3.58
w/o cross-attention	5.89	4.78	4.29	4.14
self- + cross-attention	3.31	2.50	1.90	**1.87**
Backbone architecture				
U-Net-based	3.82	3.19	2.76	2.68
Transformer-based	3.31	2.50	1.90	**1.87**

A.10 Limitations

RePiD has several limitations. First, recursive patch-level denoising may introduce cross-patch consistency challenges, especially for scenes with strong global structures or long-range spatial dependencies. The neighborhood embedding module mitigates, rather than eliminates, this issue: distant blocks communicate only through multi-hop local propagation, and a larger neighborhood

improves consistency at a higher computational cost (Appendix A.5). Stronger global context modeling may still be needed.

Second, the ImageNet-512 results indicate that RePiD can scale to higher-resolution generation, but high-resolution fidelity still leaves room for improvement compared with strong latent-space baselines. Better stage allocation and more adaptive neighborhood aggregation may help close this gap.

Third, our experiments are restricted to class-conditional ImageNet generation and an image-to-image case study. Although RePiD could in principle accept text-derived conditioning, its scalability to large-scale text-to-image generation remains unverified, and we leave evaluation on benchmarks such as GenEval and DPG-Bench to future work.

Finally, RePiD's efficiency depends on the recursive partitioning strategy and stage configuration. Although the current design provides a favorable quality–efficiency trade-off, adaptive stage allocation may further improve the balance between sample quality, spatial consistency, and computational cost.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims in the abstract and introduction are consistent with the proposed recursive pixel-space diffusion framework and the reported experimental results.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The supplementary material discusses limitations including cross-patch consistency, high-resolution fidelity, pseudo-conditional translation, and stage configuration.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not present formal theoretical results such as theorems or lemmas requiring proofs.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The paper and appendix describe the model architecture, recursive stage configuration, training and sampling settings, and evaluation protocol in sufficient detail to reproduce the main results; all datasets used are publicly available.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: The code is not included with this version of the paper. All datasets used are publicly available, and the paper and appendix provide the implementation and experimental details needed to reproduce the main results.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: The paper and supplementary material describe the datasets, metrics, model settings, training details, sampling protocol, and evaluation settings.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: Error bars or multiple-seed statistics are not reported due to the high computational cost of training large-scale generative models.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The paper reports hardware information and computational costs, including GFLOPs and sampling time for the main comparisons.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses public datasets and standard research benchmarks without private data collection, human-subject experiments, or high-risk deployment.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [No]

    Justification: The current paper focuses on method development and does not include a separate broader-impact discussion.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The work is not a deployed user-facing system and does not introduce application-specific safety safeguards.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: The experiments use publicly available datasets following their standard research usage.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: The paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The work does not involve crowdsourcing, new annotation work, or new human-subject data collection.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The work does not involve human-subject experiments requiring IRB approval.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: Large language models are not used as a core component of the proposed method.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.