UDT: Reconciling U-Nets and Diffusion Transformers with Data-Adaptive Token Reduction

Junno Yun
University of Minnesota
yun00049@umn.edu

Yaşar Utku Alçalar
University of Minnesota
alcal029@umn.edu

Mehmet Akçakaya*
University of Minnesota
akcakaya@umn.edu

Abstract

Diffusion Transformers (DiTs) have emerged as a core architecture in generative modeling due to their scalability and adaptability to multimodal tasks. DiTs comprise isotropic transformer blocks, and learn representations progressively across depth, where the denoising objective drives later layers to focus on fine-detail reconstruction. This results in degraded representation quality and an imbalanced encoder–decoder behavior. Prior approaches such as representation alignment (REPA) mitigate this by encouraging stronger early representations via training regularization. Alternatively, U-Net-style DiT architectures introduce explicit multi-scale encoder–decoder structures for improved convergence. But they build on standard U-Net wisdom via learnable operators for spatial downsampling, which are not well-suited to transformer architectures, introducing inefficiencies and compatibility issues with components such as cross-attention and representation regularization. In this work, we propose **UDT**, a U-Net diffusion transformer that combines the representation power of DiTs with the encoding–decoding benefits of U-Nets, through data-adaptive token merging for downsampling and upsampling, while preserving the DiT token dimension. Our baseline UDT architecture outperforms existing U-Net DiTs and achieves performance comparable to REPA across all model sizes. Furthermore, using architectural optimization and REPA, UDT outperforms SiT's 7.9 FID at 1400 epochs (w/o CFG) within 40 epochs ($\sim 40\times$ faster convergence) for XL model size on 256×256 ImageNet. Finally, it achieves strong image generation performance with CFG, reaching FID of 1.38 (320 epochs) with SD-VAE and 1.35 (500 epochs) with VA-VAE, providing a new backbone for DiTs with strong empirical benefits. Code is available at `https://github.com/JN-Yun/UDT`.

1 Introduction

Vision Transformers (ViTs) [12] extend the transformer architecture [51] to image data by representing 2D images as sequences of patches and applying self-attention to capture global contextual relationships. With the rapid advancement of generative modeling, transformer-based architectures have been increasingly adopted in diffusion models (DMs), leading to the development of diffusion transformers [3, 37, 41, 50, 65]. Among these, DiT [41] has played a pivotal role by introducing a highly scalable architecture that effectively leverages transformer design for diffusion modeling, while SiT [37] extends this paradigm to the rectified flow framework [33]. Their flexibility and strong performance have established diffusion transformers as powerful backbones not only for image generation but also for broader applications such as text-to-image (T2I) generation [7, 13, 52, 57].

From a representational perspective, it is well understood that diffusion transformers inherently learn semantically meaningful representations through the denoising objective [4, 8, 56]. Since diffusion

*Corresponding Author

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

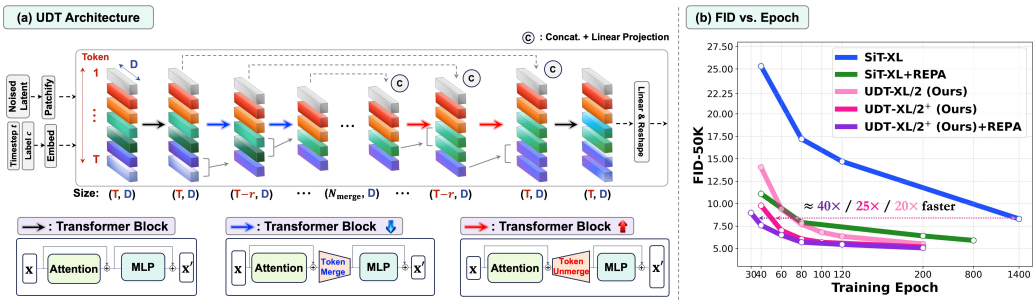


Figure 1: (a) Architecture of the proposed method **UDT**. It preserves the token hidden dimension (D) while gradually reducing and restoring the token sequence length via data-adaptive token merging and unmerging. (b) FID vs. Epoch on XL models for ImageNet 256×256 without classifier-free guidance. UDT (our baseline, marked in light pink) and UDT$^+$ (baseline + advanced techniques, marked in pink), achieve 7.7 FID at 80 epochs and 7.0 FID at 60 epochs, *without additional regularization or VAE modification*, outperforming SiT-XL's 7.9 FID at 1400 epochs. Our method with REPA achieves 7.6 FID at *just* 40 epochs, converging significantly faster than the SiT baseline and REPA.

transformers are based on the ViT architecture, which consists of a sequence of isotropic transformer blocks, they naturally learn representations progressively across depth, with semantic representation quality gradually improving from early to later layers [29]. However, diffusion models must ultimately focus on denoising and fine-detailed reconstruction, which requires recovery of higher-frequency information in later stages. As a result, representation quality tends to peak in the middle-to-late layers and then decrease toward the final layers, where the model shifts its focus from semantic encoding to denoising generation [56, 61]. This creates an imbalanced encoder–decoder structure, where the encoding stage becomes relatively deep while the effective decoding stage remains short. To address this, a line of work aims to encourage earlier formation of strong representations through training regularization, effectively moving the encoder forward and allowing later layers to devote more capacity to denoising. This direction has been explored through representation alignment with external encoders [28, 61], promoting linear separability [62], self-contrastive learning methods [53], and improving latent representations through stronger variational autoencoder (VAE) designs [28, 59, 64].

Explicit encoder–decoder architectures naturally arise in CNN-based U-Net [44] backbones that integrate spatial downsampling and upsampling with increasing channel dimensions and skip connections, improving gradient flow and preserving low- and high-frequency information. DDPM [17] adopts a CNN-based U-Net for diffusion modeling, and subsequent works [10, 22] have established U-Net variants as the standard backbone, motivating their adaptation to diffusion transformers [3, 49, 50]. U-ViT [3] incorporated skip connections into a ViT-based diffusion transformer without explicit spatial resolution changes, while later works, including U-DiT [50] and SiT↓ in UREPA [49], showed that the fast-convergence benefit primarily arises from multi-scale hierarchical modeling enabled by downsampling, with skip connections remaining important for mitigating information loss from reduced spatial resolution. However, their downsampling process relies on fixed local neighborhoods, ignoring similarity between tokens across the image, thereby weakening token-wise interactions in self-attention. It also complicates patch-wise representation alignment [61], requiring additional adaptation modules and regularization designs [49]. Moreover, architectures such as U-DiT vary channel dimensions across layers, introducing representation inconsistencies and additional engineering overhead for extensions such as T2I generation with cross-attention.

Our approach: Building on these observations, we are led to a fundamental question:

"Can we introduce an encoder-decoder architecture into diffusion transformers without disrupting their representational characteristics and self-attention dynamics?"

Our design is guided by two key principles: (1) While existing U-Net DiTs reduce the channel dimension of tokens to achieve spatial downsampling, we instead perform no dimensionality reduction in this direction and preserve the channel dimension (*i.e.*, token feature dimension) to maintain representation quality. This design also improves compatibility with downstream tasks such as T2I generation and representation alignment. (2) Rather than relying on fixed, learnable operators for spatial reduction, we advocate for a data-adaptive mechanism that accounts for token importance, enabling selective compression of similar or less informative regions while retaining critical details.

To this end, we propose **UDT**, a U-Net diffusion transformer, which introduces a transformer-specific encoder–decoder architecture that incorporates token merging (ToMe) [5] for data-adaptive and training-free downsampling and upsampling. Following the U-Net paradigm of skip connections and downsampling/upsampling (Fig. 1(a)), semantically similar tokens (*e.g.*, background regions) are progressively merged based on feature similarity, with merge indices recorded to enable exact unmerging in the decoder, while skip connections mitigate information loss induced by spatial reduction. This design adapts token merging as a transformer-compatible mechanism for resolution reduction, enabling an effective encoder–decoder structure, facilitating hierarchical representation learning while improving computational efficiency. Experiments show that UDT, out-of-the-box, substantially improves both training efficiency and generative performance on flow-based transformer models such as SiT [37], *without requiring any additional regularization or modifications to model configurations*. Our main contributions are as follows:

- We introduce a novel architecture, **U-Net Diffusion Transformer (UDT)**, which provides a new perspective on downsampling and upsampling in U-shape transformers via a data-adaptive token merging mechanism while preserving the token representation dimension.
- UDT reduces computational cost by progressively decreasing the token sequence length in an adaptive manner. Despite its lightweight architecture, our XL model achieves 7.7 FID out-of-the-box at only 80 epochs without CFG, improving on the baseline SiT model's 7.9 FID at 1400 epochs ($\sim 20\times$ faster convergence), while also outperforming even REPA methods. UDT variants with advanced techniques and REPA further improve generative performance, achieving 7.6 FID at 40 epochs ($\sim 40\times$ faster). Using CFG on 256×256 ImageNet, UDT achieves FID scores of 1.38 (at 320 epochs) with SD-VAE and 1.35 (at 500 epochs) with VA-VAE, where samples are generated by randomly sampling class labels, *i.e.* without class-balanced sampling.
- We adapt UDT for longer token sequences by leveraging early-stage token merging. By using high token reduction rates, suitable for such long sequences of patches, in the first few encoder blocks, followed by gradual merge/unmerge in later stages, efficient computation is achieved for high-resolution or long-sequence settings (*e.g.*, patch size 1 or higher resolution at 512×512) without architectural modification.
- Our improvements stem purely from architectural design, while retaining the same configuration as the DiT baseline. Thus, UDT can serve as a drop-in replacement for various DiT variants, including DiT [41] with ϵ prediction, pixel-space DiT such as JiT [30], models using improved VAE architectures such as VA-VAE [59], and T2I models such as MMDiT [13], demonstrating faster convergence and strong adaptability.

2 Related Work

2.1 Representation Learning in Diffusion Transformers

To address the characteristic of diffusion transformers where the encoding stage becomes relatively deep while the effective decoding stage remains short, a line of work has aimed to encourage the earlier formation of strong representations, effectively moving the encoder forward and allowing later layers to devote more capacity to denoising.

One prominent direction is representation alignment [21, 28, 49, 59, 61]. Among these, REPA [49, 61] improves training efficiency and generative performance by leveraging high-quality features from pretrained self-supervised visual encoders, such as DINOv2 [40], and aligning them with early intermediate representations of diffusion transformers. Subsequent variants extend alignment with external encoders into the VAE latent space [28, 59, 64]. Other approaches, such as Self-REPA (SRA) [21], avoid reliance on large-scale external encoders via internal alignment by using deeper layers, which naturally exhibit stronger representations, as teachers for earlier layers. In addition, self-supervised objectives such as self-contrastive learning [53] and methods that promote linear separability [62] improve early-layer representations without external supervision.

These studies consistently show that strengthening representations in the early stages of the model, thereby allowing subsequent layers to focus more on denoising, can significantly accelerate convergence and improve generative quality. However, these approaches *do not fundamentally improve representation learning through the diffusion transformer architecture itself*. Instead, they rely on additional external visual encoders or modifications to the VAE, which are not always applicable, or on auxiliary regularization terms, which naturally introduce computational overhead during training.

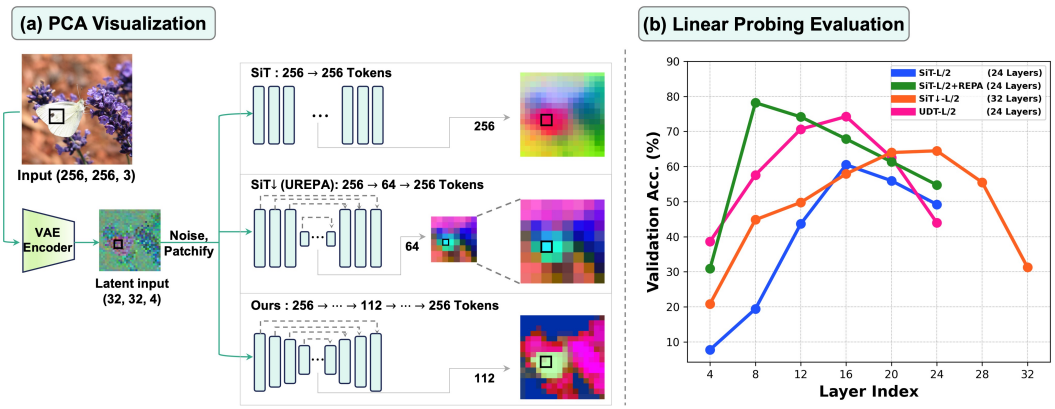


Figure 2: **Representation Analysis.** (a) PCA visualization of intermediate layers. Features are extracted from the bottleneck layer (*e.g.*, layer 18 of 24 for SiT-L/2, layer 16 of 32 for SiT↓, and layer 12 of 24 for UDT). Features with 112 tokens in our method are unmerged for visualization. (b) Linear probing evaluation on pretrained models across layers. All experiments are conducted at noise level $t = 0.1$ using Large models trained for 80 epochs on ImageNet 256×256.

2.2 U-Shape Diffusion Transformers

Motivated by the structural advantages of the U-Net [44] architecture, encoder–decoder-style designs have been explored for DiTs. U-ViT used skip connections without down/upsampling, while later works such as U-DiT [50] and SiT↓ in UREPA [49] introduce explicit resolution changes, enabling hierarchical modeling. Empirically, these works attribute improved optimization and convergence to these multi-scale representations, while skip connections facilitate preserving information lost during spatial compression. Both designs use fixed 2×2 spatial grid downsampling, where the dimensionality of tokens are first reduced using a learnable convolutional kernel. Subsequently, four spatially neighboring tokens are grouped into a single token by stacking them along the channel dimension, thereby reducing the token sequence length, *i.e.* spatial dimension. This use of fixed local neighborhoods weakens token-wise interactions in self-attention. Implementation-wise, U-DiT scales model capacity with increased feature dimensions, while SiT↓ increases number of transformer blocks. Both models further use architectural refinements, not present in original DiT/SiT. U-DiT adopts cosine similarity attention [35], rotary positional embeddings (RoPE) [48], depth-wise convolutional FFN [55], and reparameterized FFN [11], while SiT↓ employs SwiGLU [47] and RoPE [48].

Despite their benefits, existing U-Net DiTs use fixed-neighborhood down/upsampling schemes [49, 50], which introduce a fundamental tension within the transformer paradigm. These methods first compress token features and then spatially concatenate neighboring tokens along the channel dimension. While this preserves local information, it reduces four tokens into one *without* considering their global semantic similarity, thereby diminishing token-level diversity and limiting token-wise interaction capacity in self-attention to model global dependencies (Fig. 2(a), middle). For instance, a single downsampling in SiT↓ reduces 256 tokens to 64, significantly constraining self-attention. This highlights the need for downsampling that balances computational efficiency with preserving token importance and attention-critical spatial information. Finally, existing spatial downsampling schemes complicate representation alignment with pretrained ViT encoders [49], due to token dimensionality mismatch (*e.g.*, 64 vs. 256 tokens), hindering direct token-wise alignment [49].

2.3 Token Merge

Token pruning [25, 31, 39, 42, 60] and token merging [5, 38, 45] have recently emerged as a promising subfield of ViTs. These methods harness the input-agnostic nature of transformers and the widely observed redundancy of tokens in ViTs for downstream prediction. Based on this property, redundant tokens can be removed or merged at runtime, enabling faster inference with minimal performance degradation while achieving significant computational savings.

Among them, Token Merging (ToMe) [5] progressively merges tokens, reducing the number of tokens by r, thereby accelerating subsequent blocks through several key components: (1) Tokens are merged across layers based on the similarity between the keys of each token. (2) Efficient token reduction is

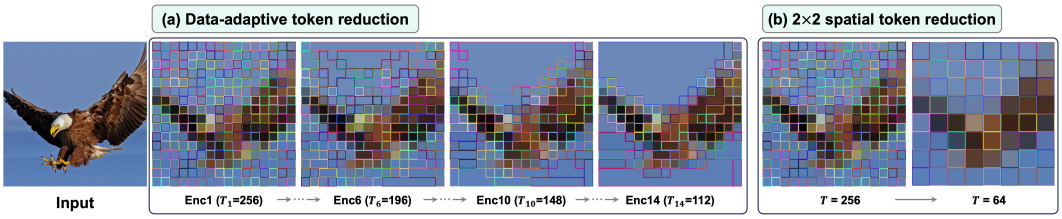


Figure 3: Visualization of token reduction: (a) Data-adaptive token reduction (ours) gradually merges semantically similar tokens (e.g., background regions), allowing fine-grained details to be retained. (b) Spatial token reduction with factor-2 downsampling, which relies on fixed local neighborhoods and ignores similarity between tokens across the image, resulting in the loss of fine details.

achieved via bipartite soft matching, which partitions tokens into two disjoint sets and connects each token in one set to its most similar counterpart in the other, forming a sparse bipartite graph. Only the top-r most similar connections are retained, and connected token pairs are merged via weighted averaging. (3) Two similar tokens with channel dimension D, $\mathbf{x}_1, \mathbf{x}_2 \in \mathbb{R}^D$, are merged into a single token using a weighted average based on token size:

$$\mathbf{x}^{\mathrm{m}}_{1,2} = \big(\mathrm{s}_1\mathbf{x}_1 + \mathrm{s}_2\mathbf{x}_2\big)/\big(\mathrm{s}_1 + \mathrm{s}_2\big), \tag{1}$$

where s_1 and s_2 denote the number of original patches represented by each token. The merged token $\mathbf{x}^{\mathrm{m}}_{1,2}$ is associated with the combined size $\mathrm{s} = \mathrm{s}_1 + \mathrm{s}_2$. (4) Lastly, since merged tokens no longer correspond to a single input patch, proportional attention is applied to correct the bias introduced by merging as: $\mathrm{softmax}(\mathbf{Q}\mathbf{K}^T/\sqrt{d} + \log \mathbf{s})$, where $\mathbf{s}$ is a row vector representing the number of original patches each token corresponds to. This formulation is equivalent to replicating each merged token s times in the attention computation.

ToMe also allows approximate recovery of the original token layout when merge indices are recorded. This unmerging step is crucial for dense prediction tasks as in DMs, which require predicting noise for every original token [6]. During unmerging, the merged token is copied back to the original token positions, while the associated sizes are restored to s_1 and s_2, respectively. Across various pretrained ViT models such as MAE [15], even with over 95% token reduction, token merging can maintain classification accuracy with only a negligible drop while enabling faster inference, while preserving meaningful semantic information [5]. We note that beyond ViTs for classification, ToMe has been applied to pretrained DMs [6] by merging tokens before self-attention for faster inference and reduced memory. But its utility has not been explored for improving training dynamics.

3 UDT: U-Net-based Diffusion Transformer with Token Merge

Building on these prior ideas, we propose a U-Net-type diffusion transformer with token merging. The proposed architecture adopts a U-Net structure and introduces transformer-friendly downsampling and upsampling via data-adaptive, parameter-free token merging and unmerging, while preserving the original DiT token representation dimension. Thus, dimensionality reduction is only constrained to the spatial dimension across tokens, without modifying token lengths themselves.

U-Net shape architecture. We first design a U-Net-shaped diffusion transformer, as illustrated in Fig. 1(a), while preserving the original DiT configurations, including the same number of blocks, attention heads, and hidden dimensions, as detailed in Appendix A. The entire network is symmetrically divided into encoder and decoder stages:

- The first encoder block and the final decoder block maintain the full token resolution and are connected through a skip connection, since diffusion models require noise prediction for every original token. This ensures that the input and output stages operate on full-resolution tokens without being affected by downsampling or upsampling (*i.e.* token merging and unmerging).
- From the second encoder block to the block before the bottleneck, token merging is progressively applied, reducing the number of tokens by r at each block until reaching the target number of merged tokens, denoted by N_{Merge} (see Appendix A for details).
- The bottleneck consists of two middle blocks operating at the lowest token resolution.
- In the decoder, token unmerging is progressively performed by restoring r tokens at each block using the recorded merge indices from the corresponding encoder stages.

Table 1: **Different Token Merging Setups.** (a) Optimal number of tokens at the bottleneck. (b) Comparison between gradual downsampling with learnable operators and token merging. (c) Effects of different components of token merging. All experiments are performed with the B/2 model on ImageNet 256×256 without CFG (80 epochs).

N_{Merge}	FID↓
96	24.3
112	**24.0**
128	24.2
144	24.3
160	25.1

(a)

Downsample Method	# Tokens	FID↓
SiT-B/2	256	33.0
Learnable Oper.	$256 \rightarrow 112$	45.5
Data-Adaptive	$256 \rightarrow 112$	**24.0**

(b)

Key-based Sim.	Weighted Avg.	Proport. Attn.	FID↓
✗	✗	✗	25.9
✓			24.5
✓	✓		24.2
✓	✓	✓	**24.0**

(c)

As shown in Tab. 1(a), performance peaks at $N_{\text{Merge}} = 112$ for our base model, with consistent behavior observed across other model sizes. An overview of the architecture, including token merging in the encoder and symmetric unmerging in the decoder, is shown in Fig. 1(a).

Down/Upsampling with data-adaptive token merge/unmerge. We examine the effectiveness of data-adaptive token merging for down/upsampling by comparing it to learnable projection operators, replacing token merge/unmerge in Fig. 1(a) with learnable projections within the same architecture. In this variant, the down/upsampling modules in Fig. 1(a) are replaced with a learnable projection function instead of token merge/unmerge. Given an input feature $\mathbf{x}^{(i)} \in \mathbb{R}^{T_i \times D}$, we apply a learnable projection $g_i(\cdot)$ that reduces the token length by a fixed amount r, *i.e.*, $g_i : \mathbb{R}^{T_i \times D} \rightarrow \mathbb{R}^{(T_i - r) \times D}$. This operation is applied progressively across layers and symmetrically used for upsampling. While this design allows gradual token reduction, it introduces token mixing, which leads to a significant degradation in performance as shown in Tab. 1(b). Similarly, comparing representations of our strategy to the downsampling scheme that includes compression in token dimensions used in prior U-shaped diffusion transformers (*e.g.*, U-DiT and SiT↓) highlights that the latter may blur fine spatial details by aggressively compressing local structures as illustrated in Fig. 2(a) (middle), showing PCA visualizations of bottleneck features. Our method, instead, enables flexible token reduction through data-adaptive token merging, where semantically similar tokens (*e.g.*, background regions) are preferentially merged (Fig. 3). As shown in Fig. 2(a) (bottom), this allows fine-grained details (*e.g.*, textures and small structures) to be retained, while redundant regions are effectively compressed. Additionally, since the token count is gradually reduced throughout the network rather than being abruptly compressed at a single stage, inference efficiency is also improved across the entire model.

Adapting general transformer improvements. A wide range of techniques have been proposed to improve transformers [11, 13, 35, 47, 48, 55], and many of these have also shown effectiveness in diffusion transformers [13, 30, 49, 50]. Building on this, our method incorporates such improvements to enhance both training efficiency and generative performance. In particular, we adopt techniques with minimal impact on parameters and FLOPs, including log-normal time sampling [13], RoPE [48], and SwiGLU [47]. For RoPE, it is applied only to the first and last blocks, since token merging/unmerging disrupts the original positional indices and restricts its use to stages where the token count remains 256. We denote UDT as the baseline model and UDT$^+$ as the version incorporating these techniques.

Extending to REPA [61]. Given its effectiveness in improving training efficiency and generative performance, compatibility with REPA is a desirable property for diffusion transformer architectures. As expected from CNN-based U-Net intuition, UREPA [49] shows that the bottleneck of a U-Net diffusion transformer is the most suitable location for representation alignment. However, fixed spatial downsampling [49, 50] introduces a mismatch with external ViT encoders, limiting faithful patch-wise alignment and requiring additional components such as upsampling MLPs and manifold losses. In contrast, our method does not require such additional engineering designs and remains compatible with the standard REPA setting. Since external encoder features share the same token resolution as the model input, we restore merged bottleneck features to the original resolution using the recorded merge indices and the unmerging operation. This allows us to recover features at the input token size from the bottleneck, where alignment is performed. Representation alignment is then achieved using an MLP projection, enabling direct patch-wise alignment without additional spatial adaptation modules or regularization loss (*e.g.*, manifold loss) [49]. We selected the last encoder layer as the target layer for REPA.

4 Experiments

Table 2: FID comparison, generated without CFG. †, +: using selected advanced techniques.

Model	#Params	GFLOPs	FID
SiT-B/2	130M	23.0	33.0
UDT-B/2	135M	17.7	24.0
UDT-B/2^{+}	135M	17.7	**20.7**
SiT↓-B/2	192M	24.7	24.2
SiT↓-B/2†	192M	24.7	20.9
UDT-M/2	180M	23.6	18.6
UDT-M/2†	180M	23.6	**15.3**
SiT-L/2	458M	80.7	18.8
UDT-L/2	479M	61.9	9.8
UDT-L/2^{+}	479M	61.9	**8.1**
SiT-XL/2	675M	118.6	17.2
SiT↓-L/2	679M	81.3	12.8
SiT↓-L/2†	679M	81.3	10.2
UDT-XL/2	707M	91.6	7.7
UDT-XL/2^{+}	707M	91.6	**6.1**
SiT↓-XL/2	946M	112.0	11.3
SiT↓-XL/2†	946M	112.0	9.2
UDT-XXL/2	935M	120.5	7.0
UDT-XXL/2^{+}	935M	120.5	**5.3**

Table 3: **System-level comparison** on ImageNet 256×256 with CFG. Lower (↓) or higher (↑) values are better.

Model	Epochs	Tokenizer	Vis. Enc.	FID↓	IS↑
Pixel diffusion					
ADM-U [10]	400	–	–	3.94	186.7
VDM++ [23]	560	–	–	**2.40**	**225.3**
Simple diffusion [20]	800	–	-	2.77	211.8
CDM [18]	2160	–	–	4.88	158.7
Latent diffusion, U-Net					
LDM-4 [43]	200	LDM-VAE	–	**3.60**	**247.7**
Latent diffusion, Transformer *without* representation learning					
DiT-XL/2 [41]	1400	SD-VAE	–	2.27	278.2
SiT-XL/2 [37]	1400	SD-VAE	–	2.06	270.3
UViT-H/2 [37]	400	SD-VAE	–	2.29	-
MaskDiT [65]	1600	SD-VAE	–	2.28	276.6
DiT + TREAD [26]	740	SD-VAE	–	1.69	292.7
UDT-XL/2 (Ours)	200	SD-VAE	–	1.57	293.0
UDT-XL/2 (Ours)	500	SD-VAE	–	**1.41**	307.4
UDT-XL/2^{+} (Ours)	200	SD-VAE	–	1.50	302.8
UDT-XL/2^{+} (Ours)	350	SD-VAE	–	1.42	**308.4**
Latent diffusion, Transformer *with* representation learning					
SiT-XL/2 + SRA [21]	800	SD-VAE	–	1.58	**311.4**
SiT-XL/2 + LSEP [62]	800	SD-VAE	–	1.46	296.8
SiT-XL/2 + REPA [61]	800	SD-VAE	DINOv2	1.42	305.7
SiT↓-XL/2† + UREPA [49]	400	SD-VAE	DINOv2	1.41	–
DDT-XL/2† + REPA [54]	400	SD-VAE	DINOv2	1.40	303.6
UDT-XL/2^{+} + REPA (Ours)	200	SD-VAE	DINOv2	1.44	296.5
UDT-XL/2^{+} + REPA (Ours)	320	SD-VAE	DINOv2	**1.38**	306.3
+ Improving VAE representation					
DiT-XL/1†, LightningDiT [59]	800	VA-VAE	–	1.35	295.3
UDT-XL/1^{+} (Ours)	500	VA-VAE	–	**1.35**	**322.0**
SiT-XL/1, REPA-E [28]	800	E2E-VAE	DINOv2	**1.26**	**314.9**

4.1 Setup

Implementation details.

We follow the experimental setup of SiT [37] and REPA [61], unless stated otherwise. All models are trained and evaluated on ImageNet [9] at 256×256, following the data preprocessing protocol of ADM [10]. We adopt the Base (B), Large (L), and X-Large (XL) models introduced in SiT [37] *without* modifying their configurations, as detailed in Appendix A. We implement our method in three variants. The baseline/out-of-the-box configuration, denoted as *UDT*, does not include any advanced techniques and is used to verify improvements that stem purely from architectural design. The enhanced version, denoted as *UDT*$^{+}$, incorporates advanced techniques that do not increase the number of parameters or GFLOPs, and serves as our final model. Finally, since our method is directly compatible with REPA, we also evaluate *UDT*$^{+}$ *+ REPA*.

Evaluation protocol. We consider two complementary evaluation settings. First, we perform architectural evaluation under the standard velocity-based training objective, comparing UDT with isotropic DiT architectures such as SiT [37] and U-Net-based transformer variants, including U-DiT [50] and SiT↓ from UREPA [49]. The goal is to elicit the improvements of our model that stem from architectural design. Second, we evaluate models augmented with REPA during training to assess how our design synergizes with advanced training strategies.

As U-DiT and SiT↓ adopt different architectural configurations (*e.g.*, channel dimensions or number of transformer blocks), we design additional models (S and M) to match their parameter scales for a fair comparison, as detailed in Appendix A. Optimization is performed using AdamW [24, 36] with learning rate 10^{-4}. All comparisons are conducted for 80 epochs, unless stated otherwise. Detailed hyperparameters and evaluation protocol are provided in Appendix A.

4.2 Results

Representation analysis. We evaluate representation quality via linear probing [1] on SiT-L/2, SiT↓-L/2, and UDT-L/2, as shown in Fig. 2(b). UDT-L/2 shows higher linear probing accuracy than SiT-L/2 and reaches its peak at the bottleneck, achieving performance comparable to SiT-

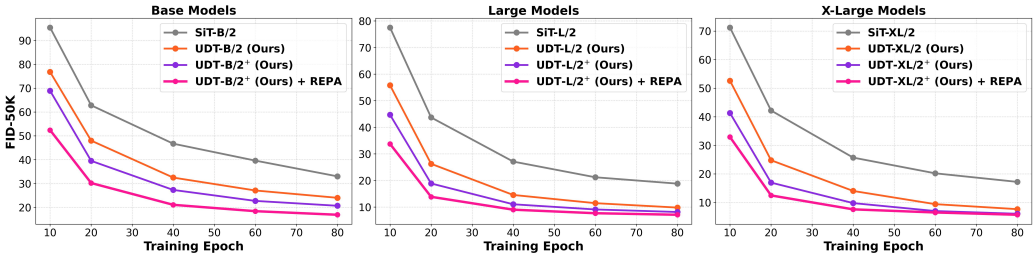


Figure 4: FID vs. Epoch on ImageNet 256×256 without CFG.

L/2+REPA at the target aligned layer with the visual encoder (8th layer). Complementary PCA visualizations in Appendix F further demonstrate that UDT preserves more clearly structured and separable components across noise levels, indicating stronger intermediate representations.

Architectural evaluation. We first assess improvements arising purely from architectural design. Tab. 2 shows that UDT consistently outperforms isotropic DiTs (SiT) across all model scales, while significantly reducing computational cost, albeit a slight increase in parameter count due to skip connections. Notably, UDT uses $\sim$75% of the GFLOPs of SiT through token merging while achieving substantial gains in generative performance. Compared to U-Net-based transformer variants, UDT also demonstrates a superior efficiency-performance trade-off. In particular, SiT$\downarrow$ has increased model depth, and when compared within the same model naming convention (*e.g.*, UDT-B/2, L/2, XL/2 vs. SiT$\downarrow$-B/2, L/2, XL/2), it has substantially higher parameter counts and computational cost. Nonetheless, it is outperformed by our UDT, which uses fewer parameters and lower GFLOPs. Furthermore, for more similar parameter scales (*e.g.*, UDT-M/2, XL/2, XXL/2 vs. SiT$\downarrow$-B/2, L/2, XL/2), the performance gap becomes even more pronounced in favor of UDT. Finally, incorporating advanced techniques without increasing computational cost (UDT$^+$) further improves performance, again outperforming comparable enhanced U-Net-based transformer variants and achieving the strongest results among models without REPA. Additional comparisons with U-DiT are in Appendix C.

Results with REPA. We next evaluate the effect of incorporating REPA for consistent model configurations. UDT and UDT$^+$ already achieve strong performance prior to alignment (Tab. 2), in several cases matching or surpassing competing models augmented with REPA (Tab. 4), indicating that the architectural improvements alone provide a competitive baseline. When combined with REPA, UDT$^+$ yields further improvements that are consistently observed across all model scales. Under similar parameter scales, UDT$^+$+REPA consistently outperforms SiT+REPA and SiT$\downarrow^\dagger$+UREPA, with this advantage extending to larger scales where it achieves the best overall performance. Notably, these results are obtained without introducing additional architectural components or auxiliary objectives, whereas UREPA relies on extra modules and manifold-based regularization.

Table 4: FID comparison of models with REPA (w/o CFG). $\dagger$, $+$: using selected advanced techniques.

Model	#Params	GFLOPs	FID
SiT-B/2 + REPA	137M	23.0	24.4
UDT-B/2$^+$ + REPA	143M	17.7	**16.8**
SiT$\downarrow$-B/2† + UREPA	200M	24.7	15.3
UDT-M/2$^+$ + REPA	180M	23.6	**12.8**
SiT-L/2 + REPA	466M	82.7	9.7
UDT-L/2$^+$ + REPA	487M	61.9	**7.1**
SiT-XL/2 + REPA	683M	118.7	7.9
SiT$\downarrow$-L/2† + UREPA	687M	81.3	5.8
UDT-XL/2$^+$ + REPA	715M	91.6	**5.7**
SiT$\downarrow$-XL/2† + UREPA	954M	109.3	5.4
UDT-XXL/2$^+$ + REPA	943M	120.5	**5.2**

System-level comparison. Tab. 3 reports results using CFG [19] with a guidance interval [27]. Our method with CFG achieves FIDs of 1.57 (UDT, out-of-the-box) and 1.50 (UDT$^+$) at only 200 epochs without explicit representation learning (middle). These results demonstrate the efficiency and strong performance of our architecture compared to the existing latent DiTs. Moreover, with continued training, UDT and UDT$^+$ *without REPA* achieve FID scores of 1.41 and 1.42 at 500 and 350 epochs, respectively, outperforming the FID achieved by REPA at 800 epochs.

Combining UDT$^+$ with REPA (bottom) further improves convergence, achieving comparable generative performance to representation learning methods in fewer epochs and ultimately reaching a SOTA FID of 1.38 at 320 epochs under SD-VAE-f8d4. Lastly, with an improved VAE (*i.e.*, VA-VAE-f16d32 [59]), our UDT$^+$ achieves an FID of 1.35 at 500 epochs, matching the FID achieved by LightningDiT after 800 epochs while attaining a higher IS. Detailed evaluations are provided in Appendix B. We also note that recent works have adopted class-balanced generation [28, 30, 54, 64],

Table 5: Ablation results for different advanced components on UDT-B/2.

Log-normal	SwiGLU	RoPE	FID
✗	✗	✗	24.0
✓			21.3
	✓		23.3
		✓	22.8
✓	✓	✓	**20.7**

Table 6: Computational comparison between patch sizes 1 and 2. Ratios (red, blue) are relative to SiT models with patch size 2.

Model	#Params	GFLOPs	Model	#Params	GFLOPs	FID
SiT-B/2	130M	23.0	UDT-B/2$^+$	135M	17.7	20.7
SiT-B/1	131M	106.4 (4.6×)	UDT-B/1$^+$	136M	49.8 (2.2×)	**16.6**
SiT-L/2	458M	80.7	UDT-L/2$^+$	479M	61.9	8.1
SiT-L/1	459M	361.0 (4.5×)	UDT-L/1$^+$	480M	116.1 (1.4×)	**7.6**
SiT-XL/2	675M	118.6	UDT-XL/2$^+$	707M	91.6	6.1
SiT-XL/1	676M	524.6 (4.4×)	UDT-XL/1$^+$	708M	158.1 (1.3×)	**5.8**

Table 7: Comparison on ImageNet 512×512 with CFG and interval.

Model	Epoch	GFLOPs	FID
DiT-XL/2	600	524.6	3.04
SiT-XL/2	600	524.6	2.62
SiT-XL/2 + REPA	200	524.6	2.10
UDT-XL/2$^+$	100	158.1	2.00
UDT-XL/2$^+$	**340**	**158.1**	**1.71**
UDT-XL/2$^+$ (FT)	**200**	**158.1**	**1.58**

which generally improves FID scores. Comparisons under this setting are provided in Appendix G. Representative qualitative samples are provided in Appendix J.

Training efficiency (FID vs. epoch). Our proposed method achieves faster convergence, measured by FID over training epochs, as shown in Fig. 1(b) and Fig. 4. Fig. 4 shows that UDT achieves the FID attained by the baseline SiT model at 80 epochs, in under 40 epochs across all model sizes. With architectural optimization, UDT$^+$, and UDT$^+$+REPA achieve the same FID in under 30 and 20 epochs, respectively. For longer training (Fig. 1(b)), UDT-XL/2 and its variants achieve 20–40× faster convergence compared to SiT-XL/2 models, while also consistently outperforming SiT-XL/2 + REPA. Further analyses of computational efficiency are provided in Appendix E.

4.3 Design Choices and Extensions

Token merge strategies. We follow the strategy of [5], computing key-based similarity, tracking the number of merged tokens, and progressively merging tokens using weighted averaging as in Eq. (1). Proportional attention is also applied to preserve consistent attention contributions after merging. Tab. 1(c) presents the ablation results with and without each component, showing that this design achieves the best performance in diffusion transformer training.

Ablation study on advanced techniques. We adopt log-normal time sampling [13] with $(\mu = 0.0, \sigma = 1.0)$, along with RoPE [48] and SwiGLU [47], and perform ablations on ImageNet 256×256 using UDT-B/2 trained for 80 epochs. As shown in Tab. 5, each component improves performance by 0.7–2.7 FID, and when combined, they synergistically yield a total improvement of 3.3 FID. Therefore, we adopt all components in our final model, denoted as UDT$^+$.

Extending to longer token sequences. Latent-space DiTs typically use a patch size of 2, balancing performance and efficiency [13, 37, 41, 49, 61]. A patch size of 1 provides finer spatial detail but significantly increases sequence length and cost [37, 41]. Our UDT naturally extends to such longer-sequence settings by applying high rates of token reduction in early encoder blocks, followed by progressive merging/unmerging. For example, with patch size 1 on 32×32 latent input (1024 tokens), early merging (e.g., 50%) quickly reduces the sequence to the standard 256-token regime, without substantial degradation due to similarity among such small patches. The model then proceeds with standard progressive reduction toward N_{Merge}, enabling efficient training and inference while retaining hierarchical U-Net modeling.

Tab. 6 (left) shows that the original SiT exhibits increased computational cost when patch size is reduced from 2 to 1, consistently increasing by ∼4.5×. In contrast, our method increases computation by only 1.3–2.2×, and this ratio decreases as the model size increases. As a result, our approach efficiently handles longer sequences (*e.g.*, 1024 instead of 256) while improving FID performance. Due to the substantial computational cost of running SiT with patch size = 1, no FID results were provided in [37], thus FID results are only reported for our model UDT$^+$.

Additional experiments on ImageNet 512×512 with a patch size of 2 further support this observation. As shown in Tab. 7, UDT$^+$, without REPA, achieves an FID of 2.00 at 100 epochs, outperforming existing DiT/SiT models as well as REPA at 200 epochs (FID 2.10). Ultimately, UDT$^+$ achieves an FID of 1.71 when trained from scratch and 1.58 when fine-tuned from a mid-training checkpoint at a resolution of 256×256, while using only 30% of the GFLOPs required by isotropic DiTs. Additional comparisons and implementation details for ImageNet 512×512 are provided in Appendix B.2.

Drop-in Replacement for Isotropic DiTs. Our method can be broadly applied as a drop-in replacement for isotropic transformer architectures to improve both training efficiency and generative

Table 9: UDT as a Drop-in Replacement for DiT Variants. Comparison on ImageNet (a,b,c) and MSCOCO (d) at 256×256. †, +: using their selected architectural optimizations. *: with CFG.

Model	Epoch	FID↓
DiT-B/2	80	43.5
UDT-DiT-B/2	80	**30.6**
DiT-L/2	80	23.3
DiT-L/2 + REPA	80	15.6
UDT-DiT-L/2	80	**13.5**
DiT-XL/2	80	19.5
DiT-XL/2 + REPA	80	12.3
UDT-DiT-XL/2	80	**11.2**

(a) ϵ-prediction (DiT [41])

Model	Epoch	FID↓	FID*↓
JiT-B/16†	50	82.6	13.6
JiT-B/16†	200	66.1	4.7
UDT-JiT-B/16+	100	64.6	6.3
UDT-JiT-B/16+	200	**58.9**	**4.6**
JiT-L/16†	50	62.2	6.9
JiT-L/16†	200	48.0	3.0
UDT-JiT-L/16+	50	47.9	5.8
UDT-JiT-L/16+	200	**37.7**	**2.8**

(b) Pixel-Level DiT (JiT [30])

Model	Tokenizer	Epoch	FID*↓	IS*↑
LightningDiT-XL/1†	VA-VAE	64	2.11	252.3
UDT-XL/1+	VA-VAE	64	**1.99**	**265.8**
LightningDiT-XL/1†	VA-VAE	800	1.35	295.3
UDT-XL/1+	VA-VAE	**500**	**1.35**	**322.0**

(c) VAE Modification (VA-VAE [59])

Model	Iter.	FID*↓
MMDiT	150K	5.5
MMUDT	150K	**4.7**

(d) T2I Model (MMDiT [13])

performance. As representative examples, we consider DiT [41] with ϵ prediction, pixel-space DiTs such as JiT [30], models using improved VAE architectures such as VA-VAE [59], as previously discussed in the system-level comparison in Section 4.2, and T2I models such as MMDiT [13]. For the first three architectures, we replace the DiT blocks with our UDT. For MMDiT, which comprises separate text and visual embedding branches, we replace the isotropic DiT-based visual branch with our UDT, denoted as MMUDT.

Tab. 9 demonstrates that replacing DiT with our UDT architecture consistently improves both training efficiency and generative performance: (a) UDT, out-of-the-box, with ϵ prediction outperforms both DiT and DiT+REPA. (b) UDT is also applicable to pixel-level DiTs, achieving improved FID scores over JiT both with and without CFG. (c) Combined with the improved VA-VAE, our method demonstrates faster convergence, achieving an FID of 1.35 in 500 epochs, matching the FID achieved by LightningDiT after 800 epochs. (d) In T2I generation on MSCOCO [32], our method further improves generative quality. These results demonstrate the *broad applicability of our method* as a drop-in replacement for existing isotropic transformer models. Detailed implementations and results are provided in Appendix D.

Efficacy on Reduced Training Data. In many real-world applications, such as medical imaging, large-scale datasets are often unavailable [2, 34, 63]. To evaluate UDT under limited data regimes, we train on only 10% of ImageNet while retaining all 1,000 classes, reducing the training set from 1.28M to 128K images. As shown in Table 8, we compare UDT against SiT-L/2 without CFG. With the limited dataset, SiT-L/2 achieves an FID of 22.9 at 500 epochs, underperforming its full-dataset counterpart (FID 18.8 at 80 epochs). In contrast, UDT-L/2 achieves FIDs of 13.7 and 10.6 at 300 and 500 epochs, respectively, while UDT$^+$-L/2 further accelerates convergence, reaching FIDs of 12.7 at 200 epochs and 10.3 at 300 epochs, surpassing SiT-L/2 trained with the full database at 80 epochs. Notably, despite achieving an even better FID, UDT$^+$-L/2 trained on the subset dataset (300 epochs) requires $\sim 2.67\times$ less wall-clock training time than SiT-L/2 trained on the full dataset (80 epochs), as it processes substantially fewer training images. These results demonstrate that UDT provides substantially improved data efficiency and faster convergence compared to isotropic DiT architectures, highlighting its effectiveness in limited-data generation scenarios.

Table 8: Comparison on Full and 10% ImageNet Subsets without CFG.

Dataset	Model	Epoch	Total Images	FID
Full set	SiT-L/2	80	102.5M	18.8
Subset (10%)	SiT-L/2	500	64.1M	22.9
	UDT-L/2	300	38.4M	13.7
		500	**64.1M**	**10.6**
	UDT-L/2+	200	25.6M	12.7
		300	**38.4M**	**10.3**

5 Conclusion

We present the U-Net Diffusion Transformer (UDT), which combines the representation power of DiTs with the architectural advantages of encoder–decoder U-Nets. Our key idea is data-adaptive token merging for transformer-specific down/upsampling, facilitating hierarchical representation learning while improving computational efficiency. UDT significantly accelerates training convergence and achieves strong generative performance out-of-the-box. Its variants further improve performance with architectural refinements and REPA, reaching an FID of 1.38 on XL models after only 320 epochs with SD-VAE, and further improves to 1.35 after 500 epochs with the stronger VA-VAE. Finally, UDT serves as a drop-in replacement for a broad range of DiT variants, including pixel-level DiTs and T2I models such as MMDiT, and works efficiently with longer token sequences, *e.g.*, at a resolution of 512×512.

References

[1] G. Alain and Y. Bengio. Understanding intermediate layers using linear classifier probes. In *Proc. Int. Conf. Learn. Represent.*, 2016.

[2] Y. U. Alçalar, M. Gülle, and M. Akçakaya. Fast MRI for all: Bridging access gaps by training without raw data. In *Proc. Adv. Neural Inf. Process. Syst.*, 2025.

[3] F. Bao, S. Nie, K. Xue, Y. Cao, C. Li, H. Su, and J. Zhu. All are worth words: A ViT backbone for diffusion models. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 22669–22679, 2023.

[4] D. Baranchuk, A. Voynov, I. Rubachev, V. Khrulkov, and A. Babenko. Label-efficient semantic segmentation with diffusion models. In *Proc. Int. Conf. Learn. Represent.*, 2022.

[5] D. Bolya, C.-Y. Fu, X. Dai, P. Zhang, C. Feichtenhofer, and J. Hoffman. Token merging: Your ViT but faster. In *Proc. Int. Conf. Learn. Represent.*, 2022.

[6] D. Bolya and J. Hoffman. Token merging for fast stable diffusion. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 4599–4603, 2023.

[7] J. Chen, C. Ge, E. Xie, Y. Wu, L. Yao, X. Ren, Z. Wang, P. Luo, H. Lu, and Z. Li. PixArt-σ: Weak-to-strong training of diffusion transformer for 4K text-to-image generation. In *Proc. Eur. Conf. Comput. Vis.*, pages 74–91. Springer, 2024.

[8] X. Chen, Z. Liu, S. Xie, and K. He. Deconstructing denoising diffusion models for self-supervised learning. In *Proc. Int. Conf. Learn. Represent.*, 2025.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 248–255, 2009.

[10] P. Dhariwal and A. Nichol. Diffusion models beat GANs on image synthesis. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 8780–8794, 2021.

[11] X. Ding, X. Zhang, N. Ma, J. Han, G. Ding, and J. Sun. Repvgg: Making vgg-style convnets great again. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 13733–13742, 2021.

[12] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. Int. Conf. Learn. Represent.*, 2021.

[13] P. Esser, S. Kulal, A. Blattmann, R. Entezari, J. Müller, H. Saini, Y. Levi, D. Lorenz, A. Sauer, F. Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Proc. Int. Conf. Mach. Learn.*, 2024.

[14] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.*, 88(2):303–338, 2010.

[15] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 16000–16009, 2022.

[16] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs trained by a two time-scale update rule converge to a local nash equilibrium. In *Proc. Adv. Neural Inf. Process. Syst.*, 2017.

[17] J. Ho, A. Jain, and P. Abbeel. Denoising diffusion probabilistic models. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 6840–6851, 2020.

[18] J. Ho, C. Saharia, W. Chan, D. J. Fleet, M. Norouzi, and T. Salimans. Cascaded diffusion models for high fidelity image generation. *J. Mach. Learn. Res.*, 23(47):1–33, 2022.

[19] J. Ho and T. Salimans. Classifier-free diffusion guidance. In *Proc. NeurIPS Workshop DGMs Appl.*, 2021.

[20] E. Hoogeboom, J. Heek, and T. Salimans. Simple diffusion: End-to-end diffusion for high resolution images. In *Proc. Int. Conf. Mach. Learn.*, pages 13213–13232, 2023.

[21] D. Jiang, M. Wang, L. Li, L. Zhang, H. Wang, W. Wei, G. Dai, Y. Zhang, and J. Wang. No other representation component is needed: Diffusion transformers can provide representation guidance by themselves. In *Proc. Int. Conf. Learn. Represent.*, 2026.

[22] T. Karras, M. Aittala, T. Aila, and S. Laine. Elucidating the design space of diffusion-based generative models. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 26565–26577, 2022.

[23] D. Kingma and R. Gao. Understanding diffusion objectives as the elbo with simple data augmentation. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 65484–65516, 2023.

[24] D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. In *Proc. Int. Conf. Learn. Represent.*, 2015.

[25] Z. Kong, P. Dong, X. Ma, X. Meng, M. Sun, W. Niu, X. Shen, G. Yuan, B. Ren, M. Qin, et al. SPViT: enabling faster vision transformers via soft token pruning. In *Proc. Eur. Conf. Comput. Vis.*, 2022.

[26] F. Krause, T. Phan, M. Gui, S. A. Baumann, V. T. Hu, and B. Ommer. TREAD: Token routing for efficient architecture-agnostic diffusion training, 2025. arXiv:2501.04765.

[27] T. Kynkäänniemi, M. Aittala, T. Karras, S. Laine, T. Aila, and J. Lehtinen. Applying guidance in a limited interval improves sample and distribution quality in diffusion models. In *Proc. Adv. Neural Inf. Process. Syst.*, 2024.

[28] X. Leng, J. Singh, Y. Hou, Z. Xing, S. Xie, and L. Zheng. REPA-E: Unlocking VAE for end-to-end tuning with latent diffusion transformers, 2025.

[29] T. Li, H. Chang, S. Mishra, H. Zhang, D. Katabi, and D. Krishnan. MAGE: Masked generative encoder to unify representation learning and image synthesis. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 2142–2152, 2023.

[30] T. Li and K. He. Back to basics: Let denoising generative models denoise. *arXiv preprint arXiv:2511.13720*, 2025.

[31] Y. Liang, C. Ge, Z. Tong, Y. Song, J. Wang, and P. Xie. Not all patches are what you need: Expediting vision transformers via token reorganizations. In *Proc. Int. Conf. Learn. Represent.*, 2022.

[32] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: Common objects in context. In *Proc. Eur. Conf. Comput. Vis.*, pages 740–755, 2014.

[33] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le. Flow matching for generative modeling. In *Proc. Int. Conf. Learn. Represent.*, 2023.

[34] G. Litjens, T. Kooi, B. E. Bejnordi, A. A. A. Setio, F. Ciompi, M. Ghafoorian, J. A. Van Der Laak, B. Van Ginneken, and C. I. Sánchez. A survey on deep learning in medical image analysis. *Med. Image Anal.*, 42:60–88, 2017.

[35] Z. Liu, Y. Lin, Y. Cao, H. Hu, Y. Wei, Z. Zhang, S. Lin, and B. Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pages 10012–10022, 2021.

[36] I. Loshchilov and F. Hutter. Decoupled weight decay regularization. In *Proc. Int. Conf. Learn. Represent.*, 2019.

[37] N. Ma, M. Goldstein, M. S. Albergo, N. M. Boffi, E. Vanden-Eijnden, and S. Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *Proc. Eur. Conf. Comput. Vis.*, pages 23–40, 2024.

[38] D. Marin, J.-H. R. Chang, A. Ranjan, A. Prabhu, M. Rastegari, and O. Tuzel. Token pooling in vision transformers. *arXiv preprint arXiv:2110.03860*, 2021.

[39] L. Meng, H. Li, B.-C. Chen, S. Lan, Z. Wu, Y.-G. Jiang, and S.-N. Lim. AdaViT: Adaptive vision transformers for efficient image recognition. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 12309–12318, 2022.

[40] M. Oquab, T. Darcet, T. Moutakanni, H. V. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. HAZIZA, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jegou, J. Mairal, P. Labatut, A. Joulin, and P. Bojanowski. DINOv2: Learning robust visual features without supervision. *Trans. Mach. Learn. Res.*, 2024.

[41] W. Peebles and S. Xie. Scalable diffusion models with transformers. In *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pages 4195–4205, 2023.

[42] Y. Rao, W. Zhao, B. Liu, J. Lu, J. Zhou, and C.-J. Hsieh. DynamicViT: Efficient vision transformers with dynamic token sparsification. In *Proc. Adv. Neural Inf. Process. Syst.*, volume 34, pages 13937–13949, 2021.

[43] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 10684–10695, 2022.

[44] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *Proc. Int. Conf. Med. Image Comput. Comput.-Assist. Intervent.*, pages 234–241, 2015.

[45] M. Ryoo, A. Piergiovanni, A. Arnab, M. Dehghani, and A. Angelova. Tokenlearner: Adaptive space-time tokenization for videos. In *Proc. Adv. Neural Inf. Process. Syst.*, volume 34, pages 12786–12797, 2021.

[46] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training GANs. In *Proc. Adv. Neural Inf. Process. Syst.*, 2016.

[47] N. Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[48] J. Su, M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[49] Y. Tian, H. Chen, M. Zheng, Y. Liang, C. Xu, and Y. Wang. U-REPA: Aligning Diffusion U-Nets to ViTs. In *Proc. Adv. Neural Inf. Process. Syst.*, 2026.

[50] Y. Tian, Z. Tu, H. Chen, J. Hu, C. Xu, and Y. Wang. U-DiTs: Downsample tokens in u-shaped diffusion transformers. In *Proc. Adv. Neural Inf. Process. Syst.*, volume 37, pages 51994–52013, 2024.

[51] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In *Proc. Adv. Neural Inf. Process. Syst.*, pages 6000–6010, 2017.

[52] J. Wang, N. Kang, L. Yao, M. Chen, C. Wu, S. Zhang, S. Xue, Y. Liu, T. Wu, X. Liu, et al. LiT: Delving into a simple linear diffusion transformer for image generation. In *ICCV*, pages 16068–16078, 2025.

[53] R. Wang and K. He. Diffuse and disperse: Image generation with representation regularization. *arXiv preprint arXiv:2506.09027*, 2025.

[54] S. Wang, Z. Tian, W. Huang, and L. Wang. DDT: Decoupled diffusion transformer. *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, 2026.

[55] Z. Wang, X. Cun, J. Bao, W. Zhou, J. Liu, and H. Li. Uformer: A general u-shaped transformer for image restoration. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 17683–17693, 2022.

[56] W. Xiang, H. Yang, D. Huang, and Y. Wang. Denoising diffusion autoencoders are unified self-supervised learners. In *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, pages 15802–15812, 2023.

[57] E. Xie, J. Chen, J. Chen, H. Cai, H. Tang, Y. Lin, Z. Zhang, M. Li, L. Zhu, Y. Lu, et al. Sana: Efficient high-resolution image synthesis with linear diffusion transformers. *arXiv preprint arXiv:2410.10629*, 2024.

[58] J. Yao, C. Wang, W. Liu, and X. Wang. FasterDiT: Towards faster diffusion transformers training without architecture modification. 37:56166–56189, 2024.

[59] J. Yao, B. Yang, and X. Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 15703–15712, 2025.

[60] H. Yin, A. Vahdat, J. M. Alvarez, A. Mallya, J. Kautz, and P. Molchanov. A-ViT: Adaptive tokens for efficient vision transformer. In *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recog.*, pages 10809–10818, 2022.

[61] S. Yu, S. Kwak, H. Jang, J. Jeong, J. Huang, J. Shin, and S. Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *Proc. Int. Conf. Learn. Represent.*, 2025.

[62] J. Yun, Y. U. Alçalar, and M. Akçakaya. No alignment needed for generation: Learning linearly separable representations in diffusion models, 2025. arXiv:2509.21565.

[63] J. Yun, Y. U. Alçalar, and M. Akçakaya. Time-embedded algorithm unrolling for computational MRI. In *Proc. Adv. Neural Inf. Process. Syst.*, 2025.

[64] B. Zheng, N. Ma, S. Tong, and S. Xie. Diffusion transformers with representation autoencoders. In *Proc. Int. Conf. Learn. Represent.*, 2026.

[65] H. Zheng, W. Nie, A. Vahdat, and A. Anandkumar. Fast training of diffusion models with masked transformers. In *Trans. Mach. Learn. Res.*, 2024.

Appendix

A Implementation Details

A.1 Model Configurations

All trainings were conducted from scratch, using 4 NVIDIA A100 GPUs for the 80-epoch setup and 8 NVIDIA A100 GPUs for higher epoch training setups.

Table 10: Hyperparameter setup for different UDT model size with patch size 2.

	UDT-S/2	**UDT-B/2**	**UDT-M/2**	**UDT-L/2**	**UDT-XL/2**	**UDT-XXL/2**
Architecture						
Input dim.	$32 \times 32 \times 4$	$32 \times 32 \times 4$	$32 \times 32 \times 4$	$32 \times 32 \times 4$	$32 \times 32 \times 4$	$32 \times 32 \times 4$
Num. layers (Enc.–Dec.)	12 (6–6)	12 (6–6)	16 (8–8)	24 (12–12)	28 (14–14)	30 (15–15)
Hidden dim.	480	768	768	1,024	1,152	1,280
Num. heads	8	12	12	16	16	16
Token Merge						
N_{Merge}	112	112	112	112	112	112
r schedule	36 (Enc 2–5)	36 (Enc 2–5)	24 (Enc 2–7)	15 (Enc 2-5), 14(Enc 6-11)	12 (Enc 2–13)	12 (Enc 2), 11(Enc 3-14)
+REPA (if used)						
Align. Depth	–	6	8	12	14	15
Weight for REPA loss	–	0.5	0.5	0.5	0.5	0.5
Optimization						
Batch size	256	256	256	256	512	512
Optimizer	AdamW	AdamW	AdamW	AdamW	AdamW	AdamW
lr	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
(β_1, β_2)	(0.9,0.999)	(0.9,0.999)	(0.9,0.999)	(0.9,0.999)	(0.9,0.999)	(0.9,0.999)
Interpolants						
α_t	$1-t$	$1-t$	$1-t$	$1-t$	$1-t$	$1-t$
σ_t	t	t	t	t	t	t
w_t	σ_t	σ_t	σ_t	σ_t	σ_t	σ_t
Training objective	v-prediction	v-prediction	v-prediction	v-prediction	v-prediction	v-prediction
Sampler	Euler-Maruyama	Euler-Maruyama	Euler-Maruyama	Euler-Maruyama	Euler-Maruyama	Euler-Maruyama
Sampling steps	250	250	250	250	250	250

Table 11: Hyperparameter setup for different UDT model size with patch size 1.

	UDT-B/1	**UDT-L/1**	**UDT-XL/1**
Architecture			
Input dim.	$32 \times 32 \times 4$	$32 \times 32 \times 4$	$32 \times 32 \times 4$
Num. layers (Enc.–Dec.)	12 (6–6)	24 (12–12)	28 (14–14)
Hidden dim.	768	1,024	1,152
Num. heads	12	16	16
Token Merge			
N_{Merge}	112	112	112
r schedule	512 (Enc 2), 256 (Enc 3), 72 (Enc 4-5)	512 (Enc 2), 256 (Enc 3), 18 (Enc 4-11)	512 (Enc 2), 256 (Enc 3), 15 (Enc 4-17), 14(Enc 8-13)
Optimization			
Training epochs	80	80	80
Batch size	256	256	512
Optimizer	AdamW	AdamW	AdamW
lr	0.0001	0.0001	0.0001
(β_1, β_2)	(0.9,0.999)	(0.9,0.999)	(0.9,0.999)
Interpolants			
α_t	$1-t$	$1-t$	$1-t$
σ_t	t	t	t
w_t	σ_t	σ_t	σ_t
Training objective	v-prediction	v-prediction	v-prediction
Sampler	Euler-Maruyama	Euler-Maruyama	Euler-Maruyama
Sampling steps	250	250	250

A.2 Evaluation Protocol

We follow the ADM evaluation protocol [10] for assessing image generation quality. We report standard metrics including Fréchet Inception Distance (FID) [16] and Inception Score (IS) [46], computed over 50K generated samples. Following SiT [37] and REPA [61], we use the SDE-based Euler–Maruyama sampler with 250 steps. Evaluation is conducted on 50K ImageNet [9] validation images resized to 256×256 or 512×512 using 4 A100-40GB GPUs with seed = 0.

A.3 Implementation of Linear Probing

Linear probing accuracy is evaluated across different depths for SiT-L/2, SiT-L/2+REPA, SiT↓-L/2 from UREPA, and UDT-L/2 (ours). Specifically, we follow the experimental setups of [56, 61] with minor modifications. For each pretrained model, we perform linear probing by attaching a batch normalization layer followed by a linear classifier, trained on the ImageNet training set for 90 epochs with a batch size of 6144. We use the Adam optimizer with a cosine decay learning rate scheduler, with an initial learning rate of 0.001. Evaluation is conducted on the 50K ImageNet validation set.

B Detailed Evaluation of UDT Variants

B.1 Experiments on ImageNet 256×256

We report detailed generation results over epochs for each UDT model with CFG on ImageNet 256×256, as shown in Tab. 12. UDT-XL/2 (out-of-the-box), UDT$^+$-XL/2, and UDT$^+$-XL/2 with REPA achieve SiT-XL/2's 2.06 FID at 1400 epochs in under 100, 80, and 50 epochs, respectively. Furthermore, all models surpass SiT-XL/2 + REPA's 1.42 FID at 800 epochs in under 410, 350, and 250 epochs, respectively. Notably, UDT-XL/2, an out-of-the-box model *without* REPA or architectural optimization, achieves an FID of 1.41 at 500 epochs.

B.2 Experiments on ImageNet 512×512

We trained UDT$^+$-XL/2 both from scratch, and also by adopting the fine-tuning strategy of DDT [54] that initializes from a mid-training checkpoint obtained at a resolution of 256×256. For the latter,

Table 12: FID of UDT variants across training epochs with CFG weights and guidance intervals on ImageNet 256×256. and indicate performance surpassing the FID achieved by SiT-XL at 1400 epochs and REPA at 800 epochs, respectively. The **best** results for each model are bolded.

Model	Epoch.	w_{cfg}	Interval	FID↓	sFID↓	IS↑	Pre.↑	Rec.↑
SiT-XL/2 (Baseline) [37]	1400	1.5	[0, 1.0]	2.06	4.50	270.3	0.82	0.59
SiT-XL/2 + REPA [61]	800	1.8	[0, 0.7]	1.42	4.70	305.7	0.80	0.64
UDT								
UDT-XL/2 (ours)	100	1.7	[0, 0.7]	1.95	4.56	266.6	0.82	0.59
UDT-XL/2 (ours)	200	1.7	[0, 0.7]	1.57	4.35	293.0	0.81	0.63
UDT-XL/2 (ours)	300	1.7	[0, 0.7]	1.48	4.33	299.3	0.80	0.63
UDT-XL/2 (ours)	400	1.7	[0, 0.7]	1.44	4.29	305.7	0.80	0.64
UDT-XL/2 (ours)	410	1.7	[0, 0.7]	1.42	4.30	306.1	0.80	0.65
UDT-XL/2 (ours)	**500**	1.7	[0, 0.7]	**1.41**	**4.28**	**307.4**	**0.80**	**0.65**
UDT$^+$								
UDT$^+$-XL/2 (ours)	80	1.7	[0, 0.7]	1.99	4.71	276.5	0.81	0.60
UDT$^+$-XL/2 (ours)	100	1.7	[0, 0.7]	1.82	4.50	287.2	0.82	0.60
UDT$^+$-XL/2 (ours)	200	1.7	[0, 0.7]	1.51	4.29	302.8	0.80	0.64
UDT$^+$-XL/2 (ours)	300	1.7	[0, 0.7]	1.49	4.37	308.1	0.80	0.64
UDT$^+$-XL/2 (ours)	320	1.7	[0, 0.7]	1.42	4.36	305.4	0.80	0.64
UDT$^+$-XL/2 (ours)	**350**	1.7	[0, 0.7]	**1.42**	**4.32**	**308.4**	**0.80**	**0.65**
UDT$^+$ + REPA								
UDT$^+$-XL/2 + REPA (ours)	50	1.7	[0, 0.7]	1.88	4.54	260.3	0.79	0.62
UDT$^+$-XL/2 + REPA (ours)	100	1.7	[0, 0.7]	1.55	4.33	283.9	0.79	0.64
UDT$^+$-XL/2 + REPA (ours)	200	1.7	[0, 0.7]	1.44	4.35	296.5	0.79	0.65
UDT$^+$-XL/2 + REPA (ours)	250	1.7	[0, 0.7]	1.42	4.47	298.2	0.79	0.65
UDT$^+$-XL/2 + REPA (ours)	280	1.7	[0, 0.7]	1.40	4.38	304.4	0.79	0.65
UDT$^+$-XL/2 + REPA (ours)	**320**	1.7	[0, 0.7]	**1.38**	**4.38**	**306.3**	**0.79**	**0.66**

Table 13: FID of UDT$^+$ across training epochs with CFG weights ($\omega_{CFG} = 2.2$ and guidance intervals [0, 0.75]) on ImageNet 512×512. indicates performance surpassing the FID achieved by REPA at 200 epochs, respectively. The **best** results for each model are bolded.

Model	Epoch.	FID↓	sFID↓	IS↑	Pre.↑	Rec.↑
DiT-XL/2 [41]	600	3.04	5.02	240.8	0.84	0.54
SiT-XL/2 [37]	600	2.62	4.18	252.2	0.84	0.57
MaskDiT [65]	800	2.50	5.10	256.3	0.84	0.57
SiT-XL/2 + REPA [61]	200	2.08	4.19	274.6	0.83	0.58
UDT$^+$						
UDT$^+$-XL/2 (ours)	100	2.00	4.86	288.0	0.80	0.62
UDT$^+$-XL/2 (ours)	150	1.85	4.81	311.0	0.79	0.65
UDT$^+$-XL/2 (ours)	200	1.79	4.83	304.6	0.79	**0.65**
UDT$^+$-XL/2 (ours)	300	1.76	4.75	**317.5**	0.80	0.62
UDT$^+$-XL/2 (ours)	**340**	**1.71**	**4.67**	316.2	**0.80**	0.64
UDT$^+$ (Fine-tuning)						
UDT$^+$-XL/2 (ours)	30	1.82	4.81	320.5	0.79	0.63
UDT$^+$-XL/2 (ours)	100	1.63	4.68	319.1	0.79	0.64
UDT$^+$-XL/2 (ours)	**200**	**1.58**	**4.58**	**327.7**	**0.79**	**0.64**

we initialized from the 250-epoch checkpoint of the UDT$^+$-XL/2 model trained at 256×256 and fine-tuned it for an additional 200 epochs.

As shown in Tab. 13, UDT$^+$-XL/2 *without* REPA achieves an FID of 2.00 after only 100 epochs of training from scratch, outperforming existing DiT-based models as well as REPA, while requiring substantially fewer training epochs. It further improves to an FID of 1.76 after 300 epochs. Moreover, by adopting the fine-tuning strategy, UDT$^+$-XL/2 achieves a SOTA FID of 1.58 after only 200 fine-tuning epochs on ImageNet 512×512.

C Comparison with U-Net DiTs

We compare two U-Net DiT variants, including U-DiT [50] and SiT↓ from UREPA [49]. Where necessary, we re-implement the baselines with and without additional techniques, marking the enhanced variants with †. SiT↓ variants are compared in Section 4.2. In this section, we provide comparison results with U-DiT [50].

Comparison with U-DiT. U-DiT is another U-Net-based transformer architecture. A direct comparison requires additional adjustments, as U-DiT fixes the number of transformer blocks to 22, performs two stages of downsampling and upsampling, and follows the standard U-Net design by increasing and decreasing channel dimensions across stages. Moreover, its model scaling differs from the conventional DiT naming scheme.

To ensure a fair comparison, we redesign our models to match the parameter count and GFLOPs of U-DiT-S and U-DiT-B, resulting in UDT-S/2 and UDT-M/2, respectively, as detailed in Appendix A. For U-DiT-L, we perform a direct comparison with UDT-XL/2. All our counterparts maintain fewer parameters than U-DiT.

Table 14: Comparison with U-DiTs on ImageNet 256×256 (80 epochs) w/o CFG.

Model	#Params	GFLOPs	FID
U-DiT-S	52M	5.9	41.0
U-DiT-S†	59M	6.0	**31.5**
UDT-S/2	53M	6.6	**39.3**
UDT-S/2$^+$	53M	6.6	34.6
U-DiT-B	204M	22.0	20.9
U-DiT-B†	231M	22.2	16.7
UDT-M/2	180M	23.6	**18.6**
UDT-M/2$^+$	180M	23.6	**15.3**
U-DiT-L	810M	84.5	12.0
U-DiT-L†	916M	85.0	10.1
UDT-XL/2	707M	91.6	**7.7**
UDT-XL/2$^+$	707M	91.6	**6.1**

As illustrated in Tab. 14, the results show that our method outperforms U-DiT in all cases except for U-DiT-S with advanced techniques (vs. UDT-S/2), and the performance gap becomes more pronounced as the model size increases. We note that U-DiT uses full tokens at the input and output stages (*i.e.*, 1024 tokens for 32×32 latent input) without patchification, and applies two stages of $2\times$ spatial downsampling, resulting in a sharp reduction in token length ($1024 \rightarrow 256 \rightarrow 64$). Despite operating with patch size 2, our method achieves better performance than U-DiT, even though U-DiT processes full tokens in the latent space. Furthermore, we note that our method can be also be used with patch size 1, yielding additional improvements, as discussed in Section 4.3.

D Drop-in Replacement for Isotropic DiT Variants

UDT serves as a drop-in replacement for a wide range of isotropic DiT variants, including DiT [41] with ϵ prediction, pixel-space DiTs such as JiT [30], models using modified VAE architectures such as VA-VAE [59], and text-to-image models such as MMDiT [13]. Unless otherwise stated, we follow the original training and sampling protocols of each baseline and replace only the backbone architecture with UDT. This enables a fair comparison and isolates the performance gains attributable to the proposed architecture.

D.1 Applying UDT to ϵ-prediction DiT

In the main text, UDT models are trained with velocity prediction following the protocol of SiT [37]. UDT can be extended to different objective functions such as ϵ-prediction, as in DiT [41]. Accordingly, we replace the isotropic DiT architecture with our proposed UDT and conduct experiments with both ϵ and variance prediction settings, denoting this variant as UDT-DiT for clarity.

As shown in Tab. 15, UDT-DiT significantly improves generative performance across model scales compared to DiT. Notably, even the out-of-the-box UDT-DiT models outperform DiT+REPA. The UDT-DiT$^+$ variant further improves performance. For the $^+$ model, we additionally apply SwiGLU and partial RoPE.

Table 15: UDT-DiT (ϵ-prediction) on ImageNet 256×256 (80 epochs) w/o CFG.

Model	#Params	GFLOPs	FID
DiT-B/2	130M	23.0	43.5
UDT-DiT-B/2	135M	17.7	**30.6**
UDT-DiT-B/2$^+$	135M	17.7	**29.1**
DiT-L/2	458M	80.7	23.3
DiT-L/2 + REPA	458M	80.7	15.6
UDT-DiT-L/2	479M	61.9	**13.5**
UDT-DiT-L/2$^+$	479M	61.9	**12.4**
DiT-XL/2	675M	118.6	19.5
DiT-XL/2 + REPA	675M	118.6	12.3
UDT-DiT-XL/2	707M	91.6	**11.2**
UDT-DiT-XL/2$^+$	707M	91.6	**9.6**

D.2 Applying UDT to Pixel-Level DiT (JiT)

Our UDT can also be applied to pixel-level DiTs. JiT [30] revisits pixel-space diffusion and shows that plain ViT architectures with larger patch sizes can be trained end-to-end on raw images without a latent tokenizer. Since ViT-based models share the same transformer blocks as DiT/SiT [37, 41], differing mainly in the embedding of noisy inputs, they can be naturally replaced with our U-shaped UDT. We use the official JiT implementation for training and evaluation.

JiT improves performance by introducing a bottleneck embedding and adopting several architectural enhancements, including in-context class conditioning (appending 32 class tokens), SwiGLU, RMSNorm, RoPE, QK-Norm, and lognormal sampling. We integrate our UDT$^+$ into JiT, denoted as UDT$^+$-JiT, adopting all architectural enhancements except RMSNorm. For in-context class conditioning, JiT appends 32 additional class tokens from a specific layer onward (*e.g.*, the 4th layer in the Base model and the 8th layer in the Large model) and maintains them until the final layer. In contrast, UDT applies in-context class tokens only around the bottleneck stages, specifically to 3 layers in the Base model and 6 layers in the Large model. This design reduces computational overhead while applying conditioning to layers with enhanced representations.

We train both the Base and Large models from scratch for 200 epochs, using a global batch size of 512 and a learning rate of 1.5×10^{-4} for the Base model, and a global batch size of 256 with a learning rate of 1×10^{-4} for the Large model. All other training configurations follow those of JiT. For sampling, we also follow the JiT's Euler 50-step sampling protocol with CFG and a CFG interval. We set $\omega_{\text{CFG}} = 3.0$ and 2.5 for the Base and Large models, respectively, with a guidance interval of $[0.1, 1.0]$. Note that JiT uses the opposite noise schedule convention from ours, where 0 corresponds to pure noise and 1 corresponds to a clean sample.

D.3 Improving UDT with a Stronger VAE (VA-VAE)

LightningDiT [59] utilizes a vision foundation model-aligned variational autoencoder (VA-VAE). Specifically, it follows a two-stage training pipeline: first, SD-VAE-f16d32 is fine-tuned using REPA with pre-trained ViTs such as DINOv2, resulting in VA-VAE-f16d32, which achieves strong performance in both generation and reconstruction. This improved VAE encoder generates latent features $\mathbf{z} \in \mathbb{R}^{16\times16\times32}$ for 256×256 image resolution. Using these latent features, a DiT model

with a patch size of 1 is trained with additional architectural optimizations and a velocity direction loss [58], further improving training effectiveness.

We replace the DiT backbone in LightningDiT-XL/1 with UDT$^+$-XL/1, adopting the same architectural optimizations and training configuration, except that we use QK-Norm instead of RMSNorm for training stability and set the global batch size to 512 and the learning rate to 1.5×10^{-4}, instead of the batch size of 1024 and learning rate of 2×10^{-4} used in LightningDiT. For sampling, we follow the LightningDiT protocol, specifically using Euler sampling with 250 steps and a timestep shift (shift factor = 0.3). We set $\omega_{\text{CFG}} = 2.8$ and the guidance interval to $[0.3, 1.0]$. Note that LightningDiT uses the opposite noise schedule convention from ours, where 0 corresponds to pure noise and 1 corresponds to a clean sample.

Table 16: UDT$^+$ with VA-VAE on ImageNet 256×256. *: results with CFG.

Model	Epoch	FID↓	IS↑	FID*↓	IS*↑
LightningDiT-XL/1	64	5.14	130.2	2.11	252.3
UDT-XL/1$^+$	64	4.90	145.8	1.99	265.8
UDT-XL/1$^+$	200	3.21	182.6	1.54	313.6
UDT-XL/1$^+$	**500**	2.45	**205.7**	**1.35**	**322.0**
LightningDiT-XL/1	800	**2.17**	205.6	1.35	295.3

As shown in Table 16, UDT demonstrates faster convergence, achieving an FID of 1.99 after only 64 epochs, surpassing LightningDiT, which reaches an FID of 2.11 with CFG. UDT further achieves an FID of 1.35 at 500 epochs, matching LightningDiT's 800-epoch performance, while obtaining a higher IS score.

D.4 Applying UDT to T2I model

We conduct T2I experiments on the MSCOCO dataset [32], following the protocol in [3, 61]. We replace the DiT backbone in MMDiT with UDT, denoted as MMUDT. Specifically, we set both MMDiT and MMUDT to 24 layers with a hidden dimension of 768, following REPA [61]. The former consists of an isotropic DiT backbone, while the latter uses our proposed UDT. We train MMDiT and MMUDT from scratch for 150K iterations with a batch size of 256 on the MSCOCO 256×256. We use the token merging configuration of the Large model for MMUDT, as described in Appendix A. For sampling, we use an SDE-based sampler with 250 steps and follow prior work by setting $\text{CFG} = 2$ [3, 61].

E Analysis of Computational Efficiency

E.1 Training Efficiency: Wall-Clock Time vs. FID

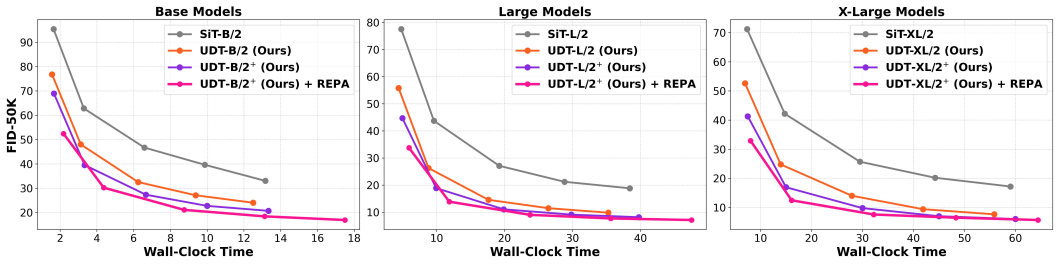


Figure 5: FID vs. Wall-Clock Time (hours) on ImageNet 256×256 without CFG.

We analyze the training efficiency of our model and its variants using Wall-Clock Time (WCT) vs. FID, compared to SiT with measurements taken at $\{10, 20, 40, 60, 80\}$ epochs. UDT achieves faster convergence and reaches the final performance at 80 epochs more efficiently than SiT, thanks to reduced computational cost enabled by token merging-based data-adaptive downsampling, while also showing faster FID convergence.

UDT† slightly slows down training due to additional architectural components (e.g., SwiGLU and RoPE), but it remains comparable to SiT in terms of WCT, while achieving better FID vs. time trade-offs. Finally, UDT†+REPA introduces additional computational overhead from extracting features from a visual encoder and applying a regularization loss (*i.e.*, patch-level alignment), resulting in

longer WCT to reach 80 epochs. Nonetheless, it achieves the fastest FID convergence WCT due to the benefits of representation alignment with strong ViT representations.

E.2 Analysis of the Computational Overhead of Token Merging/Unmerging.

Table 17: Comparison of computational efficiency among SiT-L/2, SiT↓-L/2, and UDT-L/2.

Method	Att. Time (ms)	MLP (ms)	down/upsampling (ms)	skip connection (ms)	Total Time (ms)	GPU Memory (GB)
Training						
SiT-L/2	207.34	368.85	–	–	1836.89	35.66
SiT↓-L/2	185.10	333.24	3.50	35.13	1783.41	36.13
UDT-L/2	148.42	260.91	9.58	29.67	1486.18	29.97
Inference						
SiT-L/2	207.27	368.83	–	–	596.68	4.18
SiT↓-L/2	184.66	333.08	3.49	35.12	576.69	6.33
UDT-L/2	148.34	260.85	9.25	29.65	476.67	4.50

While UDT introduces additional operations for token merging and unmerging, the overhead is negligible compared to the computational savings from reducing the number of tokens processed by the transformer blocks. To quantitatively analyze the overhead, we measured the training and inference time of the Large models on 4 NVIDIA A100-40GB GPUs with a global batch size of 256 and averaged the results over 30 iterations.

As shown in Tab. 17, both SiT↓ and UDT introduce additional costs during both training and inference due to the downsampling/upsampling operations. Specifically, SiT↓ uses fixed 2×2 spatial token downsampling and upsampling, while UDT performs adaptive token merging and unmerging. In addition, both U-shaped architectures introduce overhead from skip connections with linear projections. However, these additional costs are outweighed by the computational savings from reducing the number of tokens processed by the transformer blocks. Compared to SiT, token reduction decreases the computation of the attention and MLP blocks by 89% for SiT↓ and 70% for UDT.

Consequently, after accounting for both the overhead of the downsampling/upsampling operations and the computational savings in the attention and MLP blocks for training, UDT-L/2 requires only 81% of the per-batch training time and 84% of the GPU memory of SiT-L/2, whereas SiT↓ exhibits a similar training time but slightly higher GPU memory consumption. The reported training time includes both the forward and backward passes. For inference, UDT-L/2 requires only 80% of the per-batch inference time while using a comparable amount of GPU memory to SiT-L/2, whereas SiT↓ exhibits a similar inference time but requires $1.5\times$ more GPU memory.

A key difference between SiT↓ and UDT is that SiT↓ increases the number of transformer blocks from 24 to 32 to match the GLOPs of SiT-L/2. This results in a significant increase in the number of parameters (from 458M to 679M) and prevents it from achieving meaningful computational savings in terms of training/inference time and GPU memory. In contrast, UDT introduces only additional parameters from the skip connections (479M parameters in total) while achieving faster training and inference with reduced GPU memory usage during training.

These results demonstrate that UDT achieves improved computational efficiency in terms of both wall-clock time and GPU memory usage while maintaining strong generative performance.

F Representation Analysis

F.1 PCA Visualization

We visualize the PCA of intermediate features from each trained XL/2 model as shown in Fig. 6. The SiT-XL/2 exhibits noisy representations in the early layers at $t = 0.3$, and it shows little semantic structure until the mid-layers under higher noise levels (e.g., $t = 0.7$). SiT-XL/2+REPA, which targets the 8th layer for representation learning, produces clearer semantic structures at both $t = 0.3$ and $t = 0.7$ compared to SiT-XL/2.

SiT↓-XL/2, a U-shape diffusion transformer, shows improved representations over the SiT-XL/2 in the early layers, however, its later representations become noisy, and spatial reduction (to $H/2, W/2$)

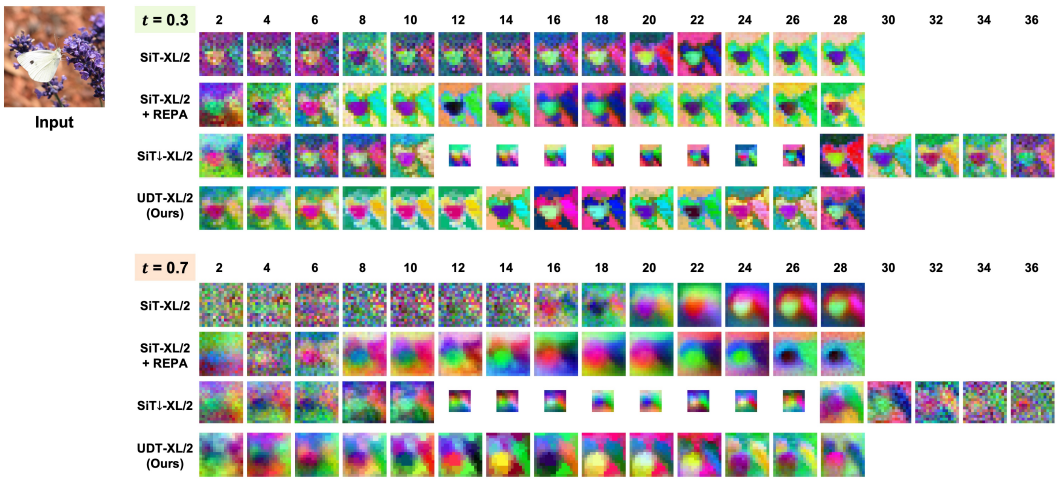


Figure 6: PCA visualization of intermediate features at $t \in \{0.3, 0.7\}$ for XL/2 models without advanced techniques, trained for 80 epochs (zoom in for a more detailed view). SiT↓-XL/2 has 36 layers in total, while the others have 28 layers.

leads to blurred fine-grained details. In contrast, our UDT-XL/2 produces sharper and more clearly separated representations across all depths, consistently outperforming other models under both low- and high-noise conditions.

F.2 Downstream Evaluation of UDT Representations on Semantic Segmentation

To evaluate whether UDT representations transfer well to dense prediction tasks, we conducted semantic segmentation experiments on the PASCAL VOC 2012 dataset [14] (train: 1,464 images; val: 1,449 images). We followed the DINOv2 [40] protocol for linear semantic segmentation. Specifically, each pretrained Large backbone (80 epochs) was frozen. Input images were resized to 256×256, encoded into the latent space, noised at $t = 0.1$, and passed through the frozen backbone to obtain patch-token representations. Only a linear segmentation head was trained on top of these extracted features. The segmentation head predicts a 16×16 logit map, which is bilinearly upsampled to 256×256 for loss computation and evaluation.

Table 18: Comparison of linear semantic segmentation performance on PASCAL data.

Method	Layer (Total)	mIoU	pixAcc
SiT-L/2	16 (24)	38.30	84.81%
SiT↓-L/2	24 (32)	41.02	85.92%
UDT-L/2	16 (24)	**55.00**	**89.78**%

For each model, we extracted features from the layer that achieved the highest linear classification probing accuracy, as shown in Fig. 2(b). For UDT, we extracted features from the 16th layer and applied token unmerging to recover the original 16×16 token grid. For SiT and UREPA, we extracted features from the 16th and 24th layers, respectively, which already have a 16×16 token grid. For training, we used a global batch size of 256, a learning rate of 5×10^{-3}, the AdamW optimizer, and trained the linear head for 100 epochs. We evaluate performance using mIoU and pixel accuracy.

As shown in Tab. 18, the representations from UDT-L/2 achieve the best segmentation performance in terms of both mIoU and pixel accuracy. These dense prediction results are consistent with the linear probing results on the classification task in the initial submission, further demonstrating that our proposed method effectively learns transferable and semantically meaningful representations.

G Evaluation with class-balanced 50K sampling.

Figure 7: Evaluation with class-balanced 50K sampling on ImageNet 256×256. Gray: results with auto-guidance instead of CFG.

Model	Epochs	Vis. Enc.	FID↓
SiT-XL/2 [37]	1400	–	1.95
SiT-XL/2 + REPA [61]	800	DINOv2	1.29
DDT-XL/2† + REPA [54]	400	DINOv2	1.26
UDT-XL/2^{+} + REPA (Ours)	**320**	DINOv2	**1.26**
with improved VAE			
SiT-XL/1, REPA-E [28]	800	DINOv2	1.15
DiT^{DH}-XL/158, RAE [28]	800	DINOv2	1.13

Recent studies have adopted class-balanced generation [28, 30, 64], which generally results in improved FID scores. To ensure a fair comparison, we evaluate our method under the same setting and compare it with the reimplemented results reported in RAE [64]. We use CFG weights ($\omega_{\text{CFG}} = 1.71$) and guidance interval $[0, 0.69]$.

H Additional Analyses.

H.1 Effects of Each Component (Token Merging and Skip Connections).

Table 19: Ablation study of token merging/unmerging and skip connections.

Method	Epoch	FID
SiT-B/2	80	33.0
(a) SiT-B/2 + Token Merge/Unmerge	80	293.0
(b) SiT-B/2 + Skip Connection	80	31.5
(c) SiT-B/2 + Token Merge/Unmerge + Skip Connection (UDT-B/2)	80	24.0

To better isolate each component's contribution, we conducted additional experiments on an isotropic DiT backbone (SiT-B/2): (a) adding token merging/unmerging for downsampling/upsampling, and (b) adding skip connections without any spatial token reduction.

As shown in Tab. 19, token merging reduces the spatial resolution and inevitably discards some high-frequency information. Without skip connections, the model cannot effectively recover this information, resulting in severely degraded generative performance. This highlights the necessity of skip connections when introducing token merging. On the other hand, adding skip connections alone to the isotropic SiT results in a U-shaped architecture similar to U-ViT. Note this leads to a simpler architecture than prior U-shaped DiTs (e.g., U-ViT, U-DiT, and SiT↓), thus the benefit from skip connections alone ends up being limited.

Finally, (c), our full UDT-B/2, combines token merging/unmerging with skip connections. The two components work synergistically, enabling efficient hierarchical token processing while preserving high-frequency information through skip connections, resulting in the best generative performance.

H.2 Impact of Different Downsampling Methods.

Existing U-shaped DiTs, such as U-DiT and SiT↓, already investigate fixed spatial downsampling strategies (e.g., convolution-based downsampling and patch merging) within the DiT architecture. Our work directly replaces these fixed downsampling operations with adaptive token merging, while keeping the overall encoder–decoder design comparable. Therefore, the comparisons with U-DiT and SiT↓ can also be viewed as a comparison between fixed spatial downsampling and adaptive token merging under similar architectural settings.

Table 20: Comparison of downsampling strategies for UDT-B/2 based on FID.

Method	Downsampling	Epoch	FID
SiT-B/2	-	80	33.0
UDT-B/2	Avg. pooling	80	29.2
UDT-B/2	Strided Conv.	80	32.3
UDT-B/2	Token Merge	80	24.0

Furthermore, we conducted additional experiments with several representative downsampling strategies. We considered average pooling and strided convolution as alternatives to token merging. Specifically, average pooling used a 2×2 kernel with stride 2, while strided convolution used a 3×3 kernel with stride 2. Both methods reduced the spatial token resolution by $2\times$ while preserving the token embedding dimension. We evaluated these approaches on UDT-B/2 by replacing token merging

with each alternative. Among the 12 transformer blocks, spatial token reduction was applied only to the middle 4 blocks, while the first and last 4 blocks maintained full token resolution with skip connections. As shown in Tab. 20, both downsampling strategies improve the generative performance over SiT-B/2, with average pooling outperforming strided convolution. However, our proposed token merging approach consistently achieves the best FID, demonstrating that adaptive token reduction is more effective than fixed local downsampling methods.

I Limitations & Impact

Limitations. The scope of this paper does not cover advanced generative settings such as video generation and 2K higher-resolution images. Further investigation is warranted to evaluate the applicability and scalability of our framework in these scenarios.

Broader Impact and Safeguards. As this work focuses on AI-generated content (AIGC), there is a possibility that the outputs may include inappropriate material. It is therefore important to remain mindful of the potential negative societal implications.

J Qualitative Results



Figure 8: Uncurated ImageNet samples at resolutions of 512×512 (left) and 256×256 (right), generated using UDT^{+}–XL/2 and UDT^{+}–XL/2 + REPA, respectively, with CFG ($\omega_{\text{CFG}} = 4.0$).

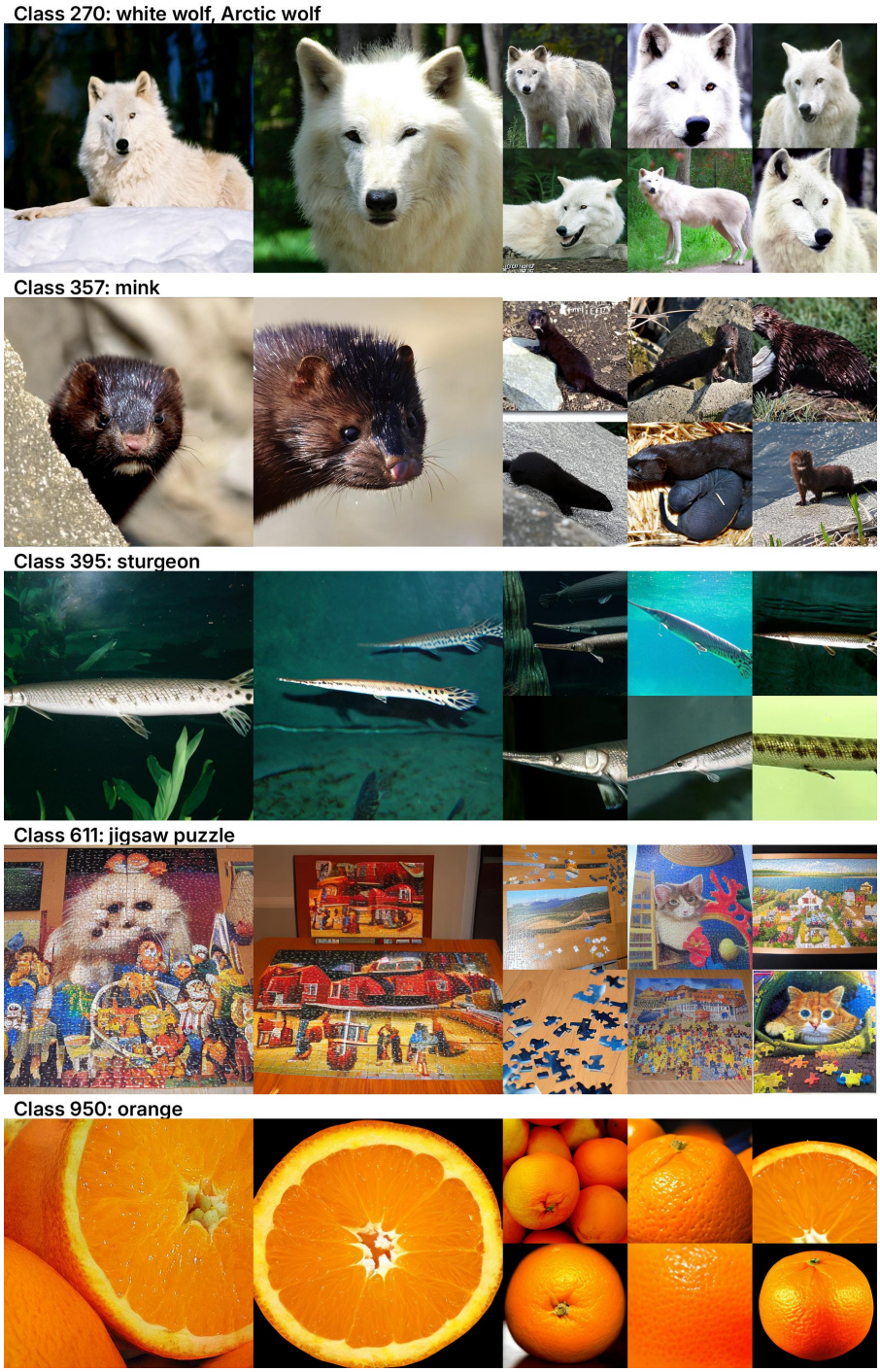


Figure 9: Uncurated ImageNet samples at resolutions of 512×512 (left) and 256×256 (right), generated using UDT^{+}–XL/2 and UDT^{+}–XL/2 + REPA, respectively, with CFG ($\omega_{\text{CFG}} = 4.0$).

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 we made in abstract and introduction accurately reflect the paper's contributions and scope.

   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 paper has discussed the limitations of the work in Appendix I.

   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 include theoretical results.

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: We have provided a full description of the experimental setup used in the paper, with detailed explanations included in Section 4 and Appendix A.

   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: [Yes]

Justification: Data for our studies are openly available and details to reproduce the main experimental results are provided in Section 4 and Appendix A.

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: This paper has specified all the training and evaluation details in Section 4 and Appendix A.

   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: [N/A]

   Justification: This is not relevant to this paper.

   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 has indicated sufficient information on the computer resources in Appendix A.

   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: This research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics.

   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: [Yes]

    Justification: The societal impacts are included in Appendix I.

    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: [Yes]

    Justification: This paper has described safeguards in Appendix I.

    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 codes and datasets used in this paper have been properly cited.

    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: This 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: This paper does not involve crowdsourcing nor research with human subjects.

    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: This paper does not involve crowdsourcing nor research with human subjects.

    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: This work does not involve the use of LLMs in any core method or experimental component.

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.