DuoDiff: Accelerating Diffusion Models with a Dual-Backbone Approach

Daniel Gallo Fernández[*]
University of Amsterdam
daniel.gallo.fernandez@student.uva.nl

Răzvan-Andrei Matişan[*]
University of Amsterdam
razvan.matisan@student.uva.nl

Alejandro Monroy Muñoz[*]
University of Amsterdam
alejandro.monroy.munoz@student.uva.nl

Ana-Maria Vasilcoiu[*]
University of Amsterdam
ana-maria.vasilcoiu@student.uva.nl

Janusz Partyka
University of Amsterdam
janusz.partyka@student.uva.nl

Tin Hadži Veljković
University of Amsterdam
t.hadziveljkovic@uva.nl

Metod Jazbec
University of Amsterdam
m.jazbec@uva.nl

Abstract

Diffusion models have achieved unprecedented performance in image generation, yet they suffer from slow inference due to their iterative sampling process. To address this, early-exiting has recently been proposed, where the depth of the denoising network is made adaptive based on the (estimated) difficulty of each sampling step. Here, we discover an interesting "phase transition" in the sampling process of current adaptive diffusion models: the denoising network consistently exits early during the initial sampling steps, until it suddenly switches to utilizing the full network. Based on this, we propose accelerating generation by employing a shallower denoising network in the initial sampling steps and a deeper network in the later steps. We demonstrate empirically that our dual-backbone approach, *DuoDiff*, outperforms existing early-exit diffusion methods in both inference speed and generation quality. Importantly, DuoDiff is easy to implement and complementary to existing approaches for accelerating diffusion.

1 Introduction

Diffusion models [21] have recently demonstrated impressive performance in generative tasks across various modalities, including images [6, 3], videos [7, 8], audio [12], and molecules [9]. However, generating new samples with diffusion can be slow, as numerous sequential calls to the denoising network are required [25]. To improve sampling efficiency [26], some of the most promising approaches focus on reducing the number of sampling steps (e.g., DDIM [22] and distillation-based methods [19, 15]) or modifying the sampling space (e.g., latent diffusion [18]).

Complementary to these efforts to accelerate diffusion, early-exiting [24] has been proposed in AdaDiff [23]. Unlike the aforementioned static methods, AdaDiff is an adaptive approach in which

[*] Alphabetical order. Equal contribution

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

the utilized depth of the denoising network can vary between sampling steps. Specifically, the difficulty of each sampling step t (where t decreases from the total number of steps T to 0) is estimated by computing the uncertainty of the denoising network at each layer. If the uncertainty is low enough, the forward pass terminates at that layer (i.e., the model *exits early*), thereby reducing computation for that step.

In this work, we leverage the adaptive nature of early-exit models to study the dynamics of the generative process in diffusion models. Interestingly, we find that early in the process (i.e., for large t), only a few layers of the denoising network are active, whereas later in the process (i.e., when t approaches 0), the full network is utilized (Figure 1). This suggests that the generation process in diffusion models begins with an easier phase, followed by a more challenging one. Motivated by these findings, we propose eliminating dynamic early-exit at every sampling step and instead introduce a (static) dual-backbone design, DuoDiff. DuoDiff consists of two denoising networks: a shallower one employed during the initial, easier phase of the generation process, and a deeper one used in the subsequent, more difficult stage (Figure 3).

We experimentally demonstrate that DuoDiff outperforms existing early-exit diffusion models in both sampling latency and image generation quality across a range of standard datasets (e.g., ImageNet 256×256). Furthermore, compared to early-exit counterparts [23, 16], DuoDiff is better suited for batch inference, as it does not require per-sample computational paths. Additionally, we show that DuoDiff can be effectively combined with other popular efficiency-enhancing methods [22, 18].

2 Background

Diffusion models generate high-quality samples by progressively adding noise to data and learning to reverse this process. The *forward process*, which adds noise to the original data, is defined as

$$\boldsymbol{x_t} = \sqrt{\bar{\alpha}_t}\boldsymbol{x_0} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon},\ \boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I}), \tag{1}$$

where $\boldsymbol{x_0} \sim q_0(\boldsymbol{x})$ is a data sample, $t \in \{T-1, \dots, 0\}$, and $\bar{\alpha}_t$ is a noise function that decreases with t (see Figure 2). Learning the generative model then corresponds to the *reverse process*, which entails fitting the denoising network $f(\boldsymbol{x}_t, t)$ using the (simplified) regression objective [6]:

$$\mathcal{L} = \mathbb{E}_{t,\boldsymbol{x_0},\boldsymbol{\epsilon}}||f(\sqrt{\bar{\alpha}_t}\boldsymbol{x_0} + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) - \boldsymbol{\epsilon}||^2\ . \tag{2}$$

After training, new samples are generated by first sampling $\boldsymbol{x_T} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$, and then iteratively applying the denoising network f according to the transition rules from DDPM [6] or DDIM [22].

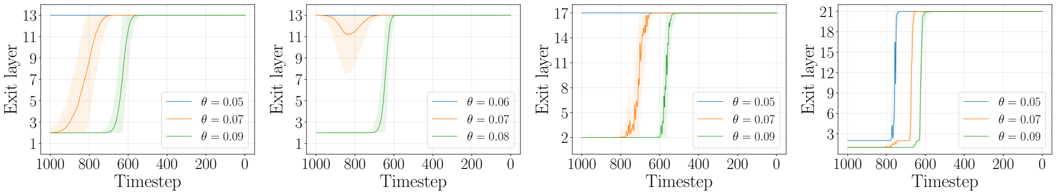
Early-exiting is a popular paradigm for making inference more efficient by allowing the model's depth to adapt based on the difficulty of the given input [24]. It has been successfully applied across various domains, including computer vision [10, 11] and language modeling [4, 20]. For diffusion models, early-exiting has been previously explored in AdaDiff [23]. To enable dynamic inference in AdaDiff, intermediate output heads are attached to the original backbone model (U-ViT [1]) before each layer $i = 0, \dots, N-1$. Furthermore, an uncertainty estimate, $u_{i,t} \in [0, 1]$, is defined at every sampling step t and at each layer i. The early (noise) prediction is returned once the uncertainty at a given layer falls below a predefined threshold $\theta \in [0, 1]$:

$$f(\boldsymbol{x_t}, t; \theta) := \begin{cases} g_0(L_{0,t}) & \text{if } u_{0,t} \leq \theta, \\ \vdots & \vdots \\ g_{N-1}(L_{N-1,t}) & \text{if } u_{N-1,t} \leq \theta, \\ g_N(L_{N,t}) & \text{otherwise.} \end{cases} \tag{3}$$

where $L_{i,t}$ denotes the activations before layer i, and g_i denotes the i-th output head. For more details, refer to Appendix A.

3 Methods

Early-Exit Trends in Diffusion Models. We begin by leveraging the adaptivity of AdaDiff [23] to study the dynamics of the generative process in diffusion models. Specifically, in Figure 1, we visualize the average exit layer across test samples for each sampling step t. Interestingly, we observe that early-exit occurs exclusively at the beginning of the reverse process. For example, on ImageNet 64×64 with threshold $\theta = 0.09$, the average exit layer equals 2 until $t \approx 600$, after which the full



(a) CIFAR-10 (b) CelebA (c) ImageNet (64 × 64) (d) ImageNet (256 × 256)

Figure 1: **Early-exit trends in AdaDiff [23].** The plots show the average exit layer across 5,120 images for different datasets and various exiting thresholds θ. We observe that early-exiting in the denoising network occurs only at the start of the generation process (for t close to T), followed by a sudden switch to using the full denoising network for the remaining generation steps. The pattern is consistent across different datasets and resembles a step function.

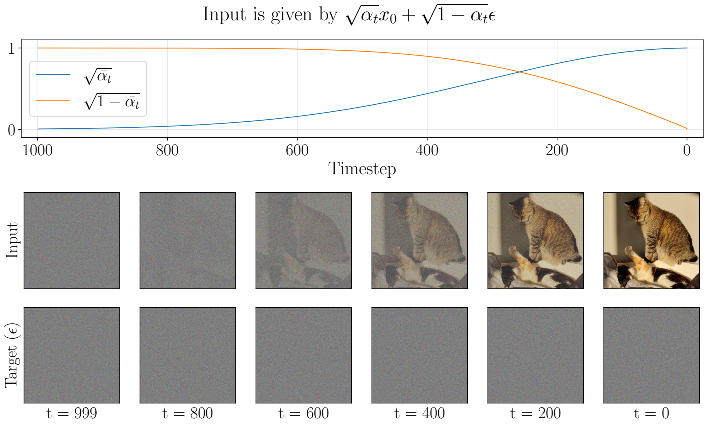


Figure 2: **Denoising objective.** Given a noisy image and a timestep, the model must predict the added noise. As we can observe, this task is easier for high values of t, in which the expected output is very similar to the input.

model is utilized. This suggests that, based on AdaDiff's exit trends, the diffusion generative process can be roughly divided into two stages, with the first one being 'easier' than the second.

Although such behaviour is surprising at first (and was overlooked in the original AdaDiff paper [23]), we demonstrate that it can be explained by taking a closer look at the training of diffusion models. To this end, observe how as t grows, $\boldsymbol{x_t}$ becomes increasingly dominated by noise (Eq. 1). Consequently, the input and expected output of the denoising network f begin to resemble each other more closely, making the task easier, as the network primarily needs to learn an identity-like behavior [2]. See Figure 2 for a more visual explanation. This is also reflected at test time, with the denoising network consistently early-exiting for larger t, indicating an easier task.

DuoDiff. Building upon the early-exit trends reported above, we propose DuoDiff, a novel diffusion framework designed to accelerate inference by employing a dual-backbone architecture. During the initial timesteps of the reverse diffusion process, where the input is largely dominated by noise and the task is simpler, DuoDiff utilizes a shallow three-layer backbone, as the early-exit layer for most samples in these timesteps is usually lower than 3 (Figure 1). As the diffusion process progresses and the input becomes more structured, DuoDiff switches to the full backbone for the remaining, more complex timesteps. We denote by t_s the number of steps during which the shallow model is active. Both the shallow and the complete backbones are trained from scratch on the same dataset using the same diffusion training objective. In addition, both backbones are trained for all values of t such that one can freely choose t_s after training. Figure 3 provides a detailed illustration of DuoDiff's design.

[2] This is further supported by the training loss being larger for smaller values of t, e.g., see Figure 2 in [17].

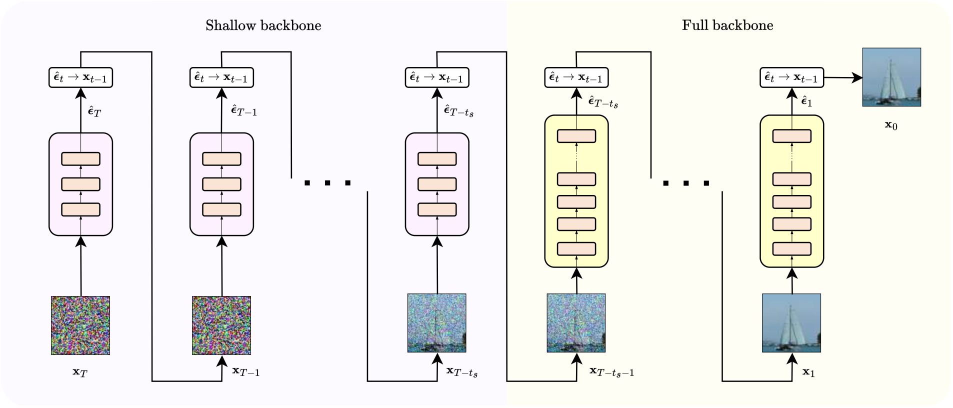

Shallow backbone
Full backbone
$\hat{\boldsymbol{\epsilon}}_t \to \mathbf{x}_{t-1}$
$\hat{\boldsymbol{\epsilon}}_T$
$\hat{\boldsymbol{\epsilon}}_{T-1}$
$\hat{\boldsymbol{\epsilon}}_{T-t_s}$
$\hat{\boldsymbol{\epsilon}}_1$
$\mathbf{x}_0$
$\mathbf{x}_T$
$\mathbf{x}_{T-1}$
$\mathbf{x}_{T-t_s}$
$\mathbf{x}_{T-t_s-1}$
$\mathbf{x}_1$

Figure 3: **DuoDiff framework.** DuoDiff employs a shallow three-layer U-ViT backbone for the first t_s timesteps to reduce computational overhead, before switching to a full backbone for the remaining denoising steps, ensuring both efficiency and image quality. Both backbones are trained on the same dataset using the same diffusion objective.

Unlike AdaDiff, which relies on dynamic early-exit mechanisms based on per-sample uncertainty levels (Eq. 3), DuoDiff simplifies this process by using a fixed transition point between the two backbones. While this sacrifices the adaptiveness of early-exiting (i.e., varying compute based on sample's difficulty), we believe this is well justified here as we observe very little variability in exiting patterns between different samples (as indicated by small standard deviation bars in Figure 1). Moreover, the static approach eliminates the batching inefficiencies caused by AdaDiff's varying exit points for different samples (see Appendix A.5), making batch inference more efficient and easier to implement.

4 Experiments

In order to illustrate the capabilities of DuoDiff, we compare it to AdaDiff on three widely used datasets: CIFAR-10 32×32 [13] and CelebA 64×64 [14] for unconditional generation and ImageNet [2] for class-conditional generation. For ImageNet, we evaluate the models on two resolutions: 64×64 and 256×256, enabling us to assess DuoDiff's scalability across varying image sizes. For ImageNet 256×256, we train our diffusion models in latent space. We utilize the U-ViT [1] architecture as the base model. In all experiments, DuoDiff employs a shallow three-layer backbone, while the full model varies in size depending on the dataset (see Tables 2 and 3).

We evaluate the quality of the generated images using the FID score [5] and measure the performance by recording the inference time per sample. Additionally, we test DuoDiff with both DDPM and DDIM samplers and provide evidence that DuoDiff works seamlessly with latent space diffusion. All metrics are computed over 5,120 images, processed in batches of 128. For AdaDiff, computing the inference time using batch sampling is challenging. For more details on batching, see Appendix A.5.

Appendix C presents the hyperparameters and further implementation details. We also make publicly available our code on GitHub[3] which contains both the DuoDiff and AdaDiff implementations together with experiments, configuration files, and demo notebooks.

Performance and Image Quality on AdaDiff. In this study, we compare the performance of AdaDiff and DuoDiff, demonstrating that DuoDiff surpasses AdaDiff in both image quality and sampling efficiency. Figure 4 illustrates the FID scores and inference time across ImageNet 64×64 and ImageNet 256×256. For a tabular view of all the results, please refer to Appendix B.

DuoDiff demonstrates superior performance over both the baseline and AdaDiff in terms of inference time. This outcome is expected, as DuoDiff leverages a shallow U-ViT for the first timesteps, while AdaDiff incurs additional overhead from its uncertainty-based early-exit mechanism.

[3] `https://github.com/razvanmatisan/duodiff`

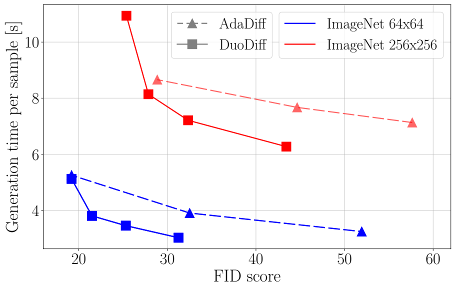


Figure 4: **Comparison of AdaDiff and DuoDiff.** Comparison of AdaDiff and DuoDiff. The plot shows FID score and generation time per sample (lower is better for both) across two datasets (ImageNet 64×64 and 256×256). Each point represents a different parameter configuration, including the base model, which can be seen as a special case of DuoDiff ($t_s = 0$). We can see how DuoDiff consistently outperforms AdaDiff in both performance and inference time.

Table 1: **Compatibility with DDIM.** Image quality (FID score) and inference speed using DuoDiff with DDIM sampling in a latent space. We observe how DuoDiff successfully increases the sampling speed without a significant impact in image quality.

Dataset	**Base model**	**Inference method**	**FID score ↓**	**Inference Time [s] ↓**
ImageNet (256×256)	U-ViT-L/2	DDIM ($\eta = 0, n\_steps = 50$)	27.82	0.55
		+ DuoDiff ($t_s = 150$)	29.17	0.47
		+ DuoDiff ($t_s = 200$)	30.06	0.46
		+ DuoDiff ($t_s = 300$)	34.36	0.41

However, the decrease in sampling time for both methods is accompanied by a decline in FID scores. For AdaDiff, this decline is more pronounced, with a clear trade-off between faster inference and lower image quality as θ increases. In contrast, while DuoDiff also experiences a reduction in FID scores as the t_s value increases, this decline is significantly less severe compared to AdaDiff, with the image quality remaining more stable and closer to the baseline. For example, on ImageNet 256×256 and with a computational budget of $7s$ per sample, AdaDiff achieves a FID score of 57, whereas DuoDiff achieves a FID score of 32 – an improvement of roughly 40%. Refer to Table 2 for more details and quantitative results.

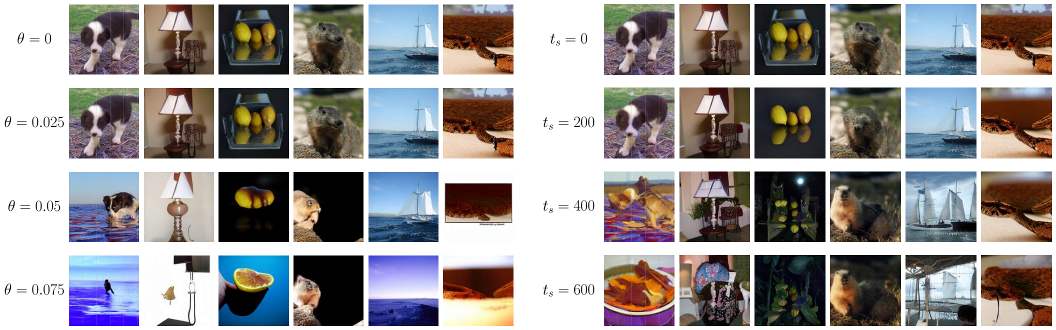
Moreover, Table 1 demonstrates that DuoDiff can be used alongside other techniques such as DDIM [22] and latent diffusion [18].

Hyperparameters Effect on Performance. A general trend can be observed for both AdaDiff and DuoDiff: as the threshold hyperparameters (θ and t_s, respectively) increase, image quality progressively degrades, while inference time decreases. This relationship is illustrated qualitatively in Figure 5 and quantified in Table 2. We leave for future work the incorporation of more principled mechanisms for threshold selection.

5 Conclusion & Future Work

In this paper, we have introduced DuoDiff, a dual-backbone alternative to adaptive diffusion models motivated by the consistency of the early-exit trends. We show that DuoDiff substantially decreases per-sample inference time while maintaining image quality. DuoDiff is also compatible with other diffusion techniques, including latent space diffusion and DDIM sampling, providing an efficient solution to address the slow inference speed of diffusion models.

Future research will focus on exploring different DuoDiff configurations, such as increasing the number of layers in the shallow transformer in order to increase t_s. Additionally, a promising direction involves investigating early-exit trends across different diffusion parametrizations, such as predicting the original image rather than the added noise.



(a) AdaDiff threshold (θ)

(b) DuoDiff switch time (t_s)

Figure 5: **Qualitative hyperparameter analysis.** Comparison of image generation results for AdaDiff (left) and DuoDiff (right) on the ImageNet dataset (256×256) using different values for their respective hyperparameters (θ in AdaDiff and t_s in DuoDiff). We observe how higher values of θ and t_s diminish the quality of the generated images.

References

[1] Fan Bao et al. “All are worth words: A vit backbone for diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2023, pp. 22669–22679.

[2] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009, pp. 248–255. DOI: `10.1109/CVPR.2009.5206848`.

[3] Prafulla Dhariwal and Alexander Nichol. “Diffusion models beat gans on image synthesis”. In: *Advances in neural information processing systems* 34 (2021), pp. 8780–8794.

[4] Maha Elbayad et al. “Depth-adaptive transformer”. In: *arXiv preprint arXiv:1910.10073* (2019).

[5] Martin Heusel et al. “Gans trained by a two time-scale update rule converge to a local nash equilibrium”. In: *NeurIPS* (2017).

[6] Jonathan Ho, Ajay Jain, and Pieter Abbeel. “Denoising diffusion probabilistic models”. In: *Advances in neural information processing systems* 33 (2020), pp. 6840–6851.

[7] Jonathan Ho et al. “Imagen video: High definition video generation with diffusion models”. In: *arXiv preprint arXiv:2210.02303* (2022).

[8] Jonathan Ho et al. “Video diffusion models”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 8633–8646.

[9] Emiel Hoogeboom et al. “Equivariant diffusion for molecule generation in 3d”. In: *International conference on machine learning*. PMLR. 2022, pp. 8867–8887.

[10] Gao Huang et al. “Multi-scale dense networks for resource efficient image classification”. In: *arXiv preprint arXiv:1703.09844* (2017).

[11] Metod Jazbec et al. “Towards anytime classification in early-exit architectures by enforcing conditional monotonicity”. In: *Advances in Neural Information Processing Systems* 36 (2024).

[12] Zhifeng Kong et al. “Diffwave: A versatile diffusion model for audio synthesis”. In: *arXiv preprint arXiv:2009.09761* (2020).

[13] Alex Krizhevsky and Geoffrey Hinton. *Learning multiple layers of features from tiny images*. Tech. rep. 0. Toronto, Ontario: University of Toronto, 2009. URL: `https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf`.

[14] Ziwei Liu et al. “Deep Learning Face Attributes in the Wild”. In: *ICCV*. 2015.

[15] Eric Luhman and Troy Luhman. “Knowledge distillation in iterative generative models for improved sampling speed”. In: *arXiv preprint arXiv:2101.02388* (2021).

[16] Taehong Moon et al. “Early exiting for accelerated inference in diffusion models”. In: *ICML 2023 Workshop on Structured Probabilistic Inference $\{\backslash\&\}$ Generative Modeling*. 2023.

[17] Alex Nichol and Prafulla Dhariwal. “Improved Denoising Diffusion Probabilistic Models”. In: *CoRR* (2021).

[18] Robin Rombach et al. “High-resolution image synthesis with latent diffusion models”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 10684–10695.

[19] Tim Salimans and Jonathan Ho. “Progressive distillation for fast sampling of diffusion models”. In: *arXiv preprint arXiv:2202.00512* (2022).

[20] Tal Schuster et al. “Confident adaptive language modeling”. In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 17456–17472.

[21] Jascha Sohl-Dickstein et al. “Deep unsupervised learning using nonequilibrium thermodynamics”. In: *International conference on machine learning*. PMLR. 2015, pp. 2256–2265.

[22] Jiaming Song, Chenlin Meng, and Stefano Ermon. “Denoising diffusion implicit models”. In: *arXiv preprint arXiv:2010.02502* (2020).

[23] Shengkun Tang et al. *AdaDiff: Accelerating Diffusion Models through Step-Wise Adaptive Computation*. 2024. arXiv: 2309.17074.

[24] Surat Teerapittayanon, Bradley McDanel, and Hsiang-Tsung Kung. “Branchynet: Fast inference via early exiting from deep neural networks”. In: *2016 23rd international conference on pattern recognition (ICPR)*. IEEE. 2016, pp. 2464–2469.

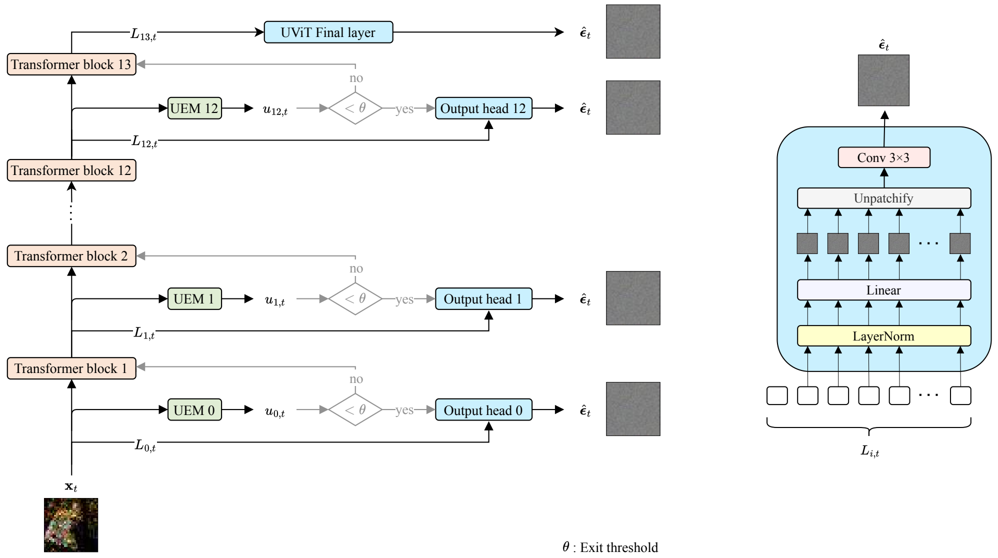
[25] Jakub M. Tomczak. *Deep Generative Modeling*. English. Germany: Springer, Feb. 2022. ISBN: 978-3-030-93157-5. DOI: 10.1007/978-3-030-93158-2.

[26] Anwaar Ulhaq, Naveed Akhtar, and Ganna Pogrebna. “Efficient diffusion models for vision: A survey”. In: *arXiv preprint arXiv:2210.09292* (2022).

A AdaDiff

A.1 Architecture

AdaDiff implements a dynamic early-exit strategy, where Uncertainty Estimation Modules (UEMs) are used to determine whether computation can be halted at each layer of the model. This process is illustrated in Figure 6, which displays the AdaDiff architecture built on top of a 13-layer U-ViT transformer, and the architectural design of the output heads.



(a) Architecture

(b) Output head

Figure 6: **AdaDiff architecture.** AdaDiff architecture integrated in a U-ViT transformer with 13 layers. An Uncertainty Estimation Module is included before each transformer block to check whether early-exiting can be applied. In the affirmative case, an output head computes the predicted noise from the output of the previous transformer block. U-ViT skip connections are omitted for simplicity.

A.2 Timestep-Aware Uncertainty Estimation Module (UEM)

For the implementation of the uncertainty estimation networks, they propose a timestep-aware UEM in the form of a fully-connected layer:

$$u_{i,t} = f\left(\mathbf{w}_t^T \left[L_{i,t}\ \ , timesteps\ \right] + \mathbf{b}_t\right) \tag{4}$$

where $\mathbf{w}_t$, $\mathbf{b}_t$, f, and $timesteps$ are the weight matrix, weight bias, activation function, and timestep embeddings, respectively. The pseudo-uncertainty ground truth is constructed as follows:

$$\hat{u}_{i,t} = F\left(\left|\mathbf{g}_i\left(L_{i,t}\right) - \boldsymbol{\epsilon}\right|\right) \tag{5}$$

where $\mathbf{g}_i$ is the output head, $\boldsymbol{\epsilon}$ is the ground truth noise value and F is a function to keep the output smaller than one (the authors use $F = \tanh$). The implementation of the output layer, shown in Figure 6b, is inspired on the final layer of the U-ViT architecture.This brings forth the loss function of this module, designed as the MSE loss of the estimated and pseudo-uncertainty ground truth:

$$\mathcal{L}_u^t = \sum_{i=0}^{N-1} \left\|u_{i,t} - \hat{u}_{i,t}\right\|^2 . \tag{6}$$

During inference, early-exiting is then achieved by comparing the estimated uncertainty of the output prediction from each layer with a predefined threshold. Figure 7 provides a visual representation of the UEM.

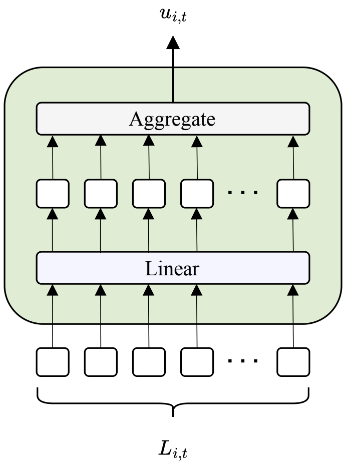


Figure 7: **Uncertainty Estimation Module.** We design the UEM as an multilayer perceptron, as specified by the AdaDiff's authors.

A.3 Uncertainty-Aware Layer-wise Loss

The authors also propose an uncertainty-aware layer-wise loss. They draw inspiration from previous work, with one important modification, a weighting term to give more importance to the output layers where the uncertainty is lower (i.e., early-exiting will happen).

$$\mathcal{L}_{UAL}^{t}=\sum_{i=0}^{N-1}\left(1-u_{i,t}\right)\times\left\|\mathbf{g}_{i}\left(L_{i,t}\right)-\boldsymbol{\epsilon}_{t}\right\|^{2}. \tag{7}$$

A.4 Training Strategy

AdaDiff utilizes a joint training strategy to balance the effect between uncertainty estimation loss and uncertainty-aware layer-wise loss, added to the orignal diffusion loss:

$$\mathcal{L}_{\text{all}}=\mathcal{L}_{\text{simple}}^{t}(\boldsymbol{\theta})+\lambda\mathcal{L}_{u}^{t}+\beta\mathcal{L}_{UAL}^{t} \tag{8}$$

In their experiments, the authors chose $\lambda=1$ and $\beta=1$, which we keep the same throughout our study.

A.5 Batching Issues

Implementing early-exiting is problematic when the batch size is larger than one, as some samples are "ready" to exit early while others are not. A possible implementation would be to use a *shrinking batch size*: start with a fixed batch size (e.g., 128) and as it goes through the transformer, take out the samples that are ready.

To simplify, we *simulated* early-exiting: we make all samples go through the entire transformer, and keep the intermediate activations. Then, we compute where each sample would have exited, and replace the output with the corresponding intermediate activations. Thus, the output is the same as if we had used the *shrinking batch size* implementation.

We also record the exit-layer per sample in order to approximate the inference time. Since all layers and uncertainty modules are identical, we linearly interpolate the total running time using the average exit layer. Note that the *shrinking batch size* implementation would likely result in longer inference times, as it would need to use batch sizes that are not powers of two. Thus, we are underestimating the inference time for AdaDiff.

B Quantitative results

Table 2 shows the FID score and inference time obtained for all experiments.

Table 2: **Quantitative image generation results.** Image generation quality and speed results for the different datasets. *Linearly interpolating with mean exit layer. In practice, it is hard to apply early-exiting with a large batch size. More details regarding the computation of inference time can be found in Appendix A.5

Dataset	Base model	Inference method	FID score ↓	Inference time [s] ↓
CIFAR10 (32 × 32)	U-ViT-S/2	DDPM	17.89	1.88
		+ AdaDiff ($\theta = 0.05$)	17.89	1.93
		+ AdaDiff ($\theta = 0.07$)	17.55	1.63*
		+ AdaDiff ($\theta = 0.09$)	24.60	1.32*
		+ DuoDiff ($t_s = 300$)	17.81	1.45
		+ DuoDiff ($t_s = 400$)	17.95	1.30
		+ DuoDiff ($t_s = 500$)	18.67	1.16
CelebA (64 × 64)	U-ViT-S/4	DDPM	9.98	1.88
		+ AdaDiff ($\theta = 0.06$)	9.75	1.96*
		+ AdaDiff ($\theta = 0.07$)	9.99	1.92*
		+ AdaDiff ($\theta = 0.08$)	31.41	1.36*
		+ DuoDiff ($t_s = 300$)	10.08	1.45
		+ DuoDiff ($t_s = 400$)	10.61	1.30
		+ DuoDiff ($t_s = 500$)	12.18	1.16
ImageNet (64 × 64)	U-ViT-M/4	DDPM	19.19	5.12
		+ AdaDiff ($\theta = 0.05$)	19.19	5.25
		+ AdaDiff ($\theta = 0.07$)	32.52	3.90*
		+ AdaDiff ($\theta = 0.09$)	51.94	3.24*
		+ DuoDiff ($t_s = 300$)	21.49	3.86
		+ DuoDiff ($t_s = 400$)	25.31	3.45
		+ DuoDiff ($t_s = 500$)	31.26	3.02
ImageNet (256 × 256)	U-ViT-L/2	DDPM	25.38	10.94
		+ AdaDiff ($\theta = 0.05$)	28.86	8.66*
		+ AdaDiff ($\theta = 0.07$)	44.65	7.67*
		+ AdaDiff ($\theta = 0.09$)	57.64	7.13*
		+ DuoDiff ($t_s = 300$)	27.86	8.14
		+ DuoDiff ($t_s = 400$)	32.34	7.21
		+ DuoDiff ($t_s = 500$)	43.43	6.27
ImageNet (256 × 256)	U-ViT-L/2	DDIM ($\eta = 0, n\_steps = 50$)	27.82	0.55
		+ DuoDiff ($t_s = 150$)	29.17	0.47
		+ DuoDiff ($t_s = 200$)	30.06	0.46
		+ DuoDiff ($t_s = 300$)	34.36	0.41

C Model specifications

Inspired by the authors of U-ViT, we use the 13-layer configuration for CIFAR-10 (U-ViT-S/2) and CelebA (U-ViT-S/4), as well as 17-layer (U-ViT-M/4) and 21-layer (U-ViT-L/2) configurations for the 64×64 and 256×256 ImageNet datasets, respectively. We train everything on a single 40 GB Nvidia A100 GPU except for the full-models for ImageNet, for which we used the weights made public by the authors [1].

The training loss and strategy for AdaDiff are presented in Appendix A. In our experiments, we keep the backbone frozen and train just the output heads and UEMs, as it yielded better performance.

For DuoDiff, we train two U-ViT backbones: a shallow U-ViT with just three layers, and a large one (its size depends on the dataset they were trained, as described previously in this section). The two backbones are trained independently and for all values of t. This is important so we can freely decide t_s after training, and to ensure a smooth transition between models. For the ImageNet 256×256 dataset, we perform diffusion in latent space rather than directly in pixel space due to the large size of the images, which significantly reduces computational overhead. We use a pre-trained autoencoder to map images into latent space, which remains frozen during training. Additionally, we experiment with both DDPM and DDIM samplers, evaluating DuoDiff's performance in terms of inference speed and image quality with each approach.

In Tables 3 and 4, we present a comprehensive list of the hyperparameters that we used in our experiments.

Table 3: **U-ViT configurations.** Hyperparameters of the U-ViT backbones. We used a different backbone depending on the dataset and image resolutions used, similar to the official implementation of U-ViT [1]. *For DuoDiff, the shallow backbone will have the same model specifications except for the number of layers, which is 3.

	CIFAR-10	**CelebA**	**ImageNet** (64×64)	**ImageNet** (256×256)
Image size	32	64	64	32
Patch size	2	4	4	2
Input channels	3	3	3	4
Embedding dimension	512	512	768	1,024
Number of layers*	13	13	17	21
Number of heads	8	8	12	16
Number of classes	-	-	1,000	1,000
Latent space diffusion	No	No	No	Yes

Table 4: **Training hyperparameters.** Training hyperparameters for the baseline, AdaDiff, and DuoDiff.

Parameter	**Training value**
Training iterations	
U-ViT (CIFAR10 and CelebA)	500,000
U-ViT (ImageNet)	300,000
AdaDiff (output heads and UEMs)	100,000
Batch size	128
Optimizer	AdamW
Learning rate	2e-4
Weight decay	3e-2
β_1	0.99
β_2	0.999
Warmup steps	1,500