Revisiting Non-Autoregressive Transformers for Efficient Image Synthesis

Zanlin Ni[1*] Yulin Wang[1*] Renping Zhou[1] Jiayi Guo[1]
Jinyi Hu[1] Zhiyuan Liu[1] Shiji Song[1] Yuan Yao[2†] Gao Huang[1†]
[1]Tsinghua University [2]National University of Singapore

Abstract

The field of image synthesis is currently flourishing due to the advancements in diffusion models. While diffusion models have been successful, their computational intensity has prompted the pursuit of more efficient alternatives. As a representative work, non-autoregressive Transformers (NATs) have been recognized for their rapid generation. However, a major drawback of these models is their inferior performance compared to diffusion models. In this paper, we aim to re-evaluate the full potential of NATs by revisiting the design of their training and inference strategies. Specifically, we identify the complexities in properly configuring these strategies and indicate the possible sub-optimality in existing heuristic-driven designs. Recognizing this, we propose to go beyond existing methods by directly solving the optimal strategies in an automatic framework. The resulting method, named AutoNAT, advances the performance boundaries of NATs notably, and is able to perform comparably with the latest diffusion models at a significantly reduced inference cost. The effectiveness of AutoNAT is validated on four benchmark datasets, i.e., ImageNet-256 & 512, MS-COCO, and CC3M. Our code is available at https://github.com/LeapLabTHU/ImprovedNAT.

1. Introduction

In recent years, the field of image synthesis has witnessed a surge in popularity due to the success of highly capable diffusion models [9, 19, 36, 38, 45]. However, the iterative generation process of diffusion models tends to be computationally intensive [10, 29, 30, 45, 49], which leads to significant practical latency and energy consumption. As a consequence, a number of research efforts have been put into exploring more efficient alternatives.

Within this context, non-autoregressive Transformers (NATs) [6, 7, 25, 27, 34] have emerged as a representative work. These models offer benefits akin to those of diffusion models, such as high scalability [7] and sample diversity [6], while being notably faster due to their parallel decoding mechanism in Vector Quantized (VQ) space [10, 46]. As depicted in Figure 2, NATs start generation from a fully masked canvas, and swiftly progress by concurrently decoding multiple tokens at each step. Nevertheless, a major drawback of NATs is their inferior performance compared to diffusion models. For example, MaskGIT [6], a popular NAT model, can synthesize an image on ImageNet [40] in only 8 steps, but the generation quality measured in FID is only 6.18, which is far behind latest diffusion models [1, 33] (approximately 2).

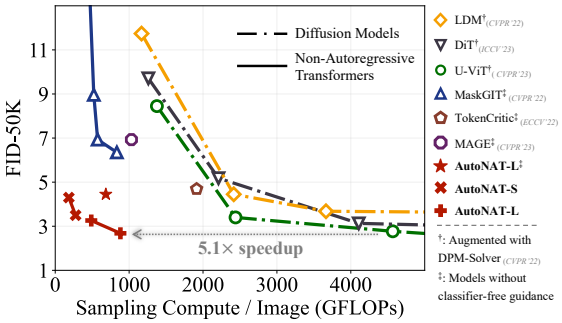


Figure 1. **FID-50K *vs*. computational cost on ImageNet-256**. For fair comparisons, diffusion models are equipped with DPM-Solver [29, 30] for efficient synthesis.

In this paper, we re-examine the performance limit of NATs. Our findings demonstrate that the inferior generation quality compared to diffusion models may not be their inherent limitation. Instead, it is largely caused by the sub-optimal, heuristic-driven strategies in the training and generation process of NATs. To be specific, different from diffusion models, the parallel decoding mechanism in NATs introduces intricate design challenges, posing questions like 1) how many tokens should be decoded at each step; 2) which tokens should be decoded; and 3) how to sample tokens from the VQ codebook? Configuring these aspects appropriately necessitates a specialized "*generation strategy*" comprising multiple scheduling functions. Moreover, a "*training strategy*" needs to be carefully designed to equip

*Equal contribution.
†Corresponding authors.

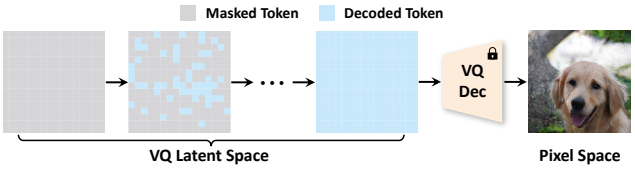


Figure 2. **The generation process of non-autoregressive Transformers** starts from an entirely masked canvas and parallelly decodes multiple tokens at each step. The generated tokens are then mapped to the pixel space with a pre-trained VQ-decoder [10].

the model with the ability to handle the varying input distributions encountered during generation. Developing proper generation and training strategies that maximally unleash the potential of NATs is a critical but under-explored open problem. The common practice in the literature predominantly designs these strategies with heuristic-driven rules (see: Tab. 1) [6, 7]. This demands extensive expert knowledge and labor-intensive efforts, yet it can still result in sub-optimal configurations.

Contrary to existing practices, we introduce a heuristic-free, automatic approach, termed AutoNAT. The major insight behind our method is to formulate the task of designing effective training and generation strategies into a unified optimization problem, as shown in Table 1. This allows for a more comprehensive exploration of NATs' full potential without being constrained by the limited prior knowledge. However, a notable challenge arises when optimizing the training strategy, as it involves a time-consuming model training process. This requirement leads to a bottleneck in the optimization procedure, hindering the swift advancement of other optimization variables. To address this challenge, we introduce a tailored alternating optimization algorithm, where the training and generation strategies are optimized alternatively according to their own characteristics. This mechanism allows us to design specialized solutions conditioned on each individual sub-problem, resulting in a more efficient and focused optimization process.

The effectiveness of AutoNAT is extensively validated on four benchmark datasets, *i.e.*, ImageNet-256 and 512 [40], MS-COCO [28], and CC3M [44]. The results show that AutoNAT outperforms previous NATs by large margins. Furthermore, compared to the diffusion models equipped with fast samplers, AutoNAT achieves at about $5\times$ inference speedup without sacrificing performance.

2. Related Work

Image generation models have historically been dominated by Generative Adversarial Networks (GANs) [5, 13, 21]. Despite their success, GANs are challenging to train, prone to mode collapse, and heavily dependent on precise hyperparameter tuning and regularization [4, 5, 20, 32]. Consequently, likelihood-based models such as diffusion [9, 35, 36, 38, 41] and autoregressive models [36, 49, 50] have emerged. These models are gaining prominence due to their straightforward training processes, scalability, and capabilities to generate diverse, high-resolution images. However, their iterative refinement approach for sampling, while powerful, demands substantial computational resources, presenting significant hurdles for real-time applications and deployment on edge devices with limited computational capacity or low latency is imperative.

Strategy	Configurations	Heuristic Design (existing works)	AutoNAT (ours)
Generation Strategy	Re-masking Ratio $r(t)$	$r(t) = \cos(\frac{\pi t}{2T})$	Jointly Optimized: $r^*(t), \tau_1^*(t), \tau_2^*(t), s^*(t), p^*(r)$
	Sampling Temp. $\tau_1(t)$	$\tau_1(t) = 1.0$	
	Re-masking Temp. $\tau_2(t)$	$\tau_2(t) = \frac{\lambda(T-t+1)}{T}$	
	Guidance Scale $s(t)$	$s(t) = \frac{kt}{T}$	
Training Strategy	Mask Ratio Dist. $p(r)$	$p(r) = \frac{2}{\pi\sqrt{1-r^2}}$	

FID on ImageNet 256×256 (model size = 46M)				
Generation Steps	$T=4$	$T=8$	$T=4$	$T=8$
TFLOPs / Image	0.18	0.28	0.18	0.28
FID-50K	8.40	5.73	**4.30** (↓4.10)	**3.52** (↓2.21)

Table 1. **Comparisons between existing works and AutoNAT**. AutoNAT tackles the strategy design of NATs by directly solving for the optimal solution, outperforming the heuristic-based counterparts by large margins. Here, T denotes generation steps. The generation strategy is controlled by the scheduling functions $r(t)$, $\tau_1(t)$, $\tau_2(t)$, $s(t)$, while the training strategy is parameterized by a mask ratio distribution $p(r)$ (see Section 3.1 for details).

Efficient image synthesis techniques have seen numerous advancements recently. There are primarily two approaches. The first involves developing new models that inherently support efficient sampling, such as non-autoregressive Transformers, which we discuss further in the next paragraph. The second approach focuses on enhancing existing models, particularly diffusion models, given their widespread use. For example, DDIM [45] and DPM-Solver [29, 30] have expedited diffusion sampling through intricate mathematical analysis, though they face challenges in preserving image quality with fewer than 10 sampling steps. Other techniques, such as DDSS [47] and AutoDiffusion [26], have adopted an optimization-based approach to boost generation efficiency. Our research aligns with these efforts in its embrace of optimization, but it uniquely focuses on non-autoregressive Transformers for their inherent efficiency advantages. Furthermore, our approach examines the interplay between training and generation, enhancing both in a unified manner. More recently, distillation-based methods [31, 43, 48] have further reduced the sampling steps of diffusion models by transferring knowledge from a large pre-trained diffusion teacher to a few-step student generator. Notably, this distillation approach to transferring the knowledge of a large diffusion teacher into a fewer-step student generator does not impose

architectural constraints on the student generator. Consequently, this technique is conceptually orthogonal to non-autoregressive Transformers and, by extension, remains compatible with the methodologies employed in AutoNAT.

Non-autoregressive Transformers (NATs) originate from machine translation [12, 14], known for their rapid inference capabilities. Recently, these models have been applied to image synthesis, offering high-quality images with a minimal number of inference steps [6, 7, 25, 27, 34]. As a pioneering work, MaskGIT [6] demonstrates highly-competitive fidelity and diversity on the ImageNet [40] with only 8 sampling steps. It has been further extended for text-to-image generation and scaled up to 3B parameters in Muse [7] and yields remarkable performance. Separately, Token-critic [25] and MAGE [27] build upon NATs: Token-critic [25] enhances MaskGIT's performance by introducing an auxiliary model for guided sampling, while MAGE [27] proposes leveraging NATs to unify representation learning with image synthesis. More recently, MAGVIT-v2 [51] further advances the field with an enhanced tokenizer design. Given that AutoNAT focuses on refining the training and generation strategies, which are the core components of all NATs, it inherently maintains compatibility with these approaches.

3. On the Limitations of Non-Autoregressive Transformers (NATs)

3.1. Preliminaries of NATs

In this subsection, we start by giving an overview of the non-autoregressive Transformers [6, 7, 27] in image generation, laying the basis for our proposed method. Non-autoregressive Transformers typically work in conjunction with a pre-trained VQ-autoencoder [10, 37, 46] to generate images. The VQ-autoencoder is responsible for the conversion between images and latent visual tokens, while the non-autoregressive Transformer learns to generate visual tokens in the latent VQ space. As the VQ autoencoder is usually pre-trained and remains unchanged throughout the generation process, we mainly outline the training and generation strategies of non-autoregressive Transformers.

Training strategy. The training of non-autoregressive Transformers is based on the masked token modeling objective [2, 8, 16]. Specifically, we denote the visual tokens obtained by the VQ-encoder as $\boldsymbol{V} = [V_i]_{i=1:N}$, where N is the sequence length. Each visual token v_i corresponds to a specific index of the VQ-encoder's codebook. During training, a variable number of tokens, specifically $\lceil r \cdot N \rceil$, are randomly selected and replaced with `[MASK]` token, where r is a mask ratio sampled from a predefined distribution $p(r)$ within the $[0, 1]$ range. The training objective is to predict the original tokens based on the surrounding unmasked ones, optimizing a cross-entropy loss function.

Generation strategy. During inference, non-autoregressive Transformers generate the latent visual tokens in a multi-step manner. The model starts from an all-`[MASK]` token sequence $\boldsymbol{V}^{(0)}$. At t^{th} step, the model predicts $\boldsymbol{V}^{(t)}$ from $\boldsymbol{V}^{(t-1)}$ by first **parallely decoding** all tokens and then **re-masking** less reliable predictions, as described below.

1. **Parallel decoding.** Given visual tokens $\boldsymbol{V}^{(t-1)}$, the model first parallely decodes all of the `[MASK]` tokens to form an initial guess $\hat{\boldsymbol{V}}^{(t)}$:

$$\hat{V}_i^{(t)} \begin{cases} \sim \hat{p}_{\tau_1(t)}(V_i|\boldsymbol{V}^{(t-1)}), & \text{if } V_i^{(t-1)} = \texttt{[MASK]}; \\ = V_i^{(t-1)}, & \text{otherwise.} \end{cases}$$

   Here, $\tau_1(\cdot)$ is the sampling temperature scheduling function, and $\hat{p}_{\tau_1(t)}(V_i|\boldsymbol{V}^{(t-1)})$ represents the model's predicted probability distribution at position i, scaled by a temperature $\tau_1(t)$. Meanwhile, confidence scores $\boldsymbol{C}^{(t)}$ are defined for all tokens:

$$C_i^{(t)} = \begin{cases} \log \hat{p}(V_i = \hat{V}_i^{(t)}|\boldsymbol{V}^{(t-1)}), & \text{if } V_i^{(t-1)} = \texttt{[MASK]}; \\ +\infty, & \text{otherwise.} \end{cases}$$

   where $\hat{p}(V_i = \hat{V}_i^{(t)}|\boldsymbol{V}^{(t-1)})$ is the predicted probability for the selected token $\hat{V}_i^{(t)}$ at position i.

2. **Re-masking.** From the initial guess $\hat{\boldsymbol{V}}^{(t)}$, the model then obtains $\boldsymbol{V}^{(t)}$ by re-masking the $\lceil r(t) \cdot N \rceil$ least confident predictions:

$$V_i^{(t)} = \begin{cases} \hat{V}_i^{(t)}, & \text{if } i \in \mathcal{I}; \\ \texttt{[MASK]}, & \text{if } i \notin \mathcal{I}. \end{cases}$$

   Here, $r(\cdot) \in [0, 1]$ is the re-masking scheduling function, which regulates the proportion of tokens to be re-masked at each step. The set $\mathcal{I}$ comprises indices of the $N - \lceil r(t) \cdot N \rceil$ most confident predictions and are sampled without replacement from $\text{Softmax}(\boldsymbol{C}^{(t)}/\tau_2(t))$[1], where $\tau_2(\cdot)$ is the re-masking temperature scheduling function.

The model iterates the process for T steps to decode all `[MASK]` tokens, yielding the final sequence $\boldsymbol{V}^{(T)}$. The sequence is then fed into the VQ-decoder to obtain the image.

3.2. Limitations of NATs

Compared to other likelihood-based generative models, one of the predominant advantages of NATs is their superior efficiency [6, 7]. Once they are appropriately deployed, decent-quality images can be generated with a few sampling steps. However, it is usually non-trivial for practitioners to utilize these models properly, since their performance tends to heavily depend on multiple scheduling functions that need to be carefully configured. As discussed above, a generation process typically involves three scheduling functions to control the mask ratio $r(t)$, the sampling temperature $\tau_1(t)$, and the re-masking temperature $\tau_2(t)$, respec-

[1] In practice, this sampling procedure is implemented via Gumbel-Top-k trick [22].

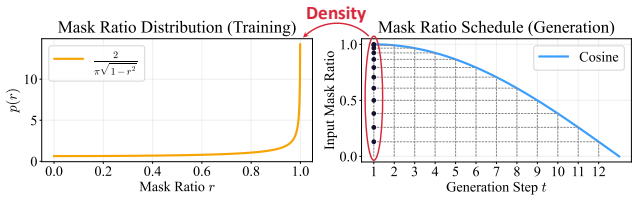


Figure 3. **The heuristic design of $p(r)$ in existing works**: the density of $p(r)$ reflects the frequency of mask ratios encountered during generation. We take $T = 12$ for example. Notably, as shown in Table 2, such a heuristic design is sub-optimal.

Training Distribution $p(r)$	Original: $\frac{2}{\pi\sqrt{1-r^2}}$	Fixed: $r \equiv 0.8$
FID-50K	8.40	**8.31**

Table 2. **Heuristically-designed $p(r)$ in existing works *vs*. fixed mask ratio** for training NATs. A simple fixed mask ratio produces even better results.

tively. When classifier-free guidance [7, 9] is adopted, a guidance scale scheduling function $s(t)$ is further introduced [7] to progressively adjust the guidance strength. Existing works typically design these functions with heuristic rules (see Table 1), necessitating expert knowledge and manual effort. Meanwhile, these heuristic rules may fall short of capturing the optimal dynamics of the generation process, leading to sub-optimal designs (see Table 6a).

Moreover, this intricate generation process requires the support of a well-designed training strategy. More precisely, it is essential to design a proper $p(r)$ for mask ratio sampling during training, such that NATs can effectively process diverse input distributions with varying proportions of mask tokens during generation. In most existing works [6, 7, 34], $p(r)$ is configured to mimic the variation of mask ratios in the generation process, as illustrated in Figure 3. However, this common design may not be proper. A counterintuitive finding in Table 2 reveals that even using a *single, fixed* mask ratio can produce *better* results than the heuristic design. A possible explanation is that the capabilities model learned from decoding at one mask ratio are highly transferable for decoding other ratios of mask tokens. Consequently, the strict, proportional allocation of $p(r)$'s density, based on the encountered frequencies of mask ratios during generation, could be sub-optimal. This finding suggests the need for a more systematic approach to better capture the relationship between training and inference.

4. Method

As aforementioned, the heuristic-driven design of training and generation strategies for non-autoregressive generative models is both labor-intensive and sub-optimal. To address this issue, we propose an optimization-based approach to derive these optimal configurations with minimal human effort. In this section, we elaborate on our AutoNAT method.

4.1. A Unified Optimization Framework

Mathematical formulation. Our objective is to determine the most appropriate configurations for the training and generation procedures of non-autoregressive Transformers. From the lens of optimization, this corresponds to identifying the optimal generation scheduling functions $r^*(t)$, $\tau_1^*(t)$, $\tau_2^*(t)$, $s^*(t)$ and mask ratio distribution $p^*(r)$ that maximize the performance of the model according to a chosen metric. Notably, since a generation process of T steps involves T values of each scheduling function, we can circumvent the difficulty of directly optimizing the functions by optimizing four groups of hyperparameters $\boldsymbol{r} = [r(t)]_{t=1:T}$, $\boldsymbol{\tau_1} = [\tau_1(t)]_{t=1:T}$, $\boldsymbol{\tau_2} = [\tau_2(t)]_{t=1:T}$, $\boldsymbol{s} = [s(t)]_{t=1:T}$ instead. Formally, this optimization problem can be defined as:

$$\boldsymbol{r}^*, \boldsymbol{\tau_1}^*, \boldsymbol{\tau_2}^*, \boldsymbol{s}^*, p^*(r) = \underset{\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}, p(r)}{\arg\max} F\left(\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}, \boldsymbol{\theta}^*_{p(r)}\right),$$

$$\text{s.t.} \quad \boldsymbol{r} \in [0,1]^T, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s} \in \mathbb{R}_+^T,$$

$$p(r) \geq 0 \quad \forall r \in [0,1], \int_0^1 p(r)dr = 1.$$

Herein, $\boldsymbol{\theta}^*_{p(r)}$ denotes the model parameters trained under the mask ratio distribution $p(r)$. The function F measures the generation quality, and can be instantiated with metrics like Fréchet inception distance (FID) [17], Inception score (IS) [42], *etc*. Note that for metrics like FID, which are considered better when smaller, we maximize their negative values. In addition, the constraints on $p(r)$ confirm that it's a valid probability distribution.

Alternating optimization. Before solving the optimization problem, we note an important distinction between $p(r)$ and other variables: $p(r)$ is nested within model parameter term $\boldsymbol{\theta}^*_{p(r)}$. Consider an arbitrary optimization procedure, where we obtain an intermediate candidate for $p(r)$. It is usually essential to train the model with $p(r)$ and evaluate how good $p(r)$ is. In contrast, evaluating $\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}$ only necessitates inferring the generative model, which is much cheaper than $p(r)$. As a consequence, directly solving all variables simultaneously would be inefficient as the slow evaluation of $p(r)$ hinders the optimization of other variables, which could otherwise proceed at a much faster pace.

Motivated by the imbalanced nature of the variables, we divide our optimization problem into two sub-problems: generation strategy optimization and training strategy optimization, and propose to solve them with an alternating algorithm. The first sub-problem focuses on obtaining the optimal configuration for the hyperparameters controlling the generation procedure $\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}$, given a model trained under $p(r)$, while the second sub-problem optimizes $p(r)$ with the given generation configurations. These two sub-problems are solved alternatively until the generation quality of the model converges. This iterative approach allows the rapid adjustment of $\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}$, while steadily steering

$p(r)$ towards optimality, yielding a superior efficiency for solving the overall optimization problem. The empirical evidence can be found in Table 6c.

Algorithm 1 Alternating Algorithm for Solving the Training & Inference Configurations of NATs

Initialize $p(r), \boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}$
Set convergence threshold ϵ, $F_{\text{prev}} \leftarrow \infty$
repeat
  `# Alternating optimization`
  $\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s} \leftarrow \text{OptimizeGenerationStrategy}(p(r))$
  $p(r) \leftarrow \text{OptimizeTrainingStrategy}(\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s})$
  `# Evaluate the strategies`
  $\boldsymbol{\theta}^*_{p(r)} \leftarrow \text{TrainModel}(p(r))$
  $F_{\text{new}} \leftarrow \text{Evaluate}(\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}, \boldsymbol{\theta}^*_{p(r)})$
  $\Delta F \leftarrow |F_{\text{new}} - F_{\text{prev}}|$
  $F_{\text{prev}} \leftarrow F_{\text{new}}$
until $\Delta F \leq \epsilon$
return $\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}, p(r)$

We summarize our overall optimization process in Algorithm 1. In the following, we elaborate on the optimization of the generation strategy and training strategy respectively.

4.2. Generation Strategy Optimization

In this sub-problem, we are interested in searching for the optimal generation strategy variables $\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}$ given a fixed $p(r)$, which can be formulated as:

$$\boldsymbol{r}^*, \boldsymbol{\tau_1}^*, \boldsymbol{\tau_2}^*, \boldsymbol{s}^* = \arg\max_{\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}} F(\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}, \boldsymbol{\theta}^*_{p(r)}),$$
$$\text{s.t.} \quad \boldsymbol{r} \in [0,1]^T, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s} \in \mathbb{R}_+^T.$$

Inspired by the success of gradient-based optimization in deep learning [39], we find this sub-problem can be effectively solved via simple gradient descent. In specific, although F is not differentiable with respect to $\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}$ due to the parallel decoding process (see Section 3.1), it is feasible to approximate the gradients corresponding to each variable by leveraging a finite difference method. Let $\boldsymbol{\xi} = [\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}]$ be the concatenation of variables. The gradient of F with respect to $\boldsymbol{\xi}$ can be approximated by

$$\frac{\partial F(\boldsymbol{\xi}, \boldsymbol{\theta}^*_{p(r)})}{\partial \xi_i} \approx \frac{F(\boldsymbol{\xi} + \epsilon \boldsymbol{e}_i, \boldsymbol{\theta}^*_{p(r)}) - F(\boldsymbol{\xi}, \boldsymbol{\theta}^*_{p(r)})}{\epsilon},$$

where i ranges over the dimensions of $\boldsymbol{\xi}$, $\boldsymbol{e}_i$ is the unit vector in the direction of ξ_i, and ϵ is a small positive number. We denote the estimated gradients as $\hat{\nabla}_{\boldsymbol{\xi}} F(\boldsymbol{\xi}, \boldsymbol{\theta}^*_{p(r)})$. Then the hyperparameters can be updated by gradient descent:

$$\boldsymbol{\xi} \leftarrow \boldsymbol{\xi} - \eta \hat{\nabla}_{\boldsymbol{\xi}} F(\boldsymbol{\xi}, \boldsymbol{\theta}^*_{p(r)}),$$

where η denotes the learning rate.

4.3. Training Strategy Optimization

In this sub-problem, we focus on searching for the optimal mask ratio distribution $p^*(r)$ with a given generation strategy $\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}$. Notably, since we usually need to train the model to evaluate any candidate values of $p^*(r)$, optimizing $p(r)$ in a general probability distribution space is computationally expensive. As a result, we propose to restrict $p(r)$ to a specific family of probability density functions. In this work, we adopt the Beta distribution as the family of $p(r)$:

$$p(r; \alpha, \beta) = \frac{r^{\alpha-1}(1-r)^{\beta-1}}{B(\alpha, \beta)},$$

where $B(\alpha, \beta)$ is the Beta function and $\alpha, \beta > 0$. Thus, we can simplify our optimization problem as:

$$\alpha^*, \beta^* = \arg\max_{\alpha, \beta} F(\boldsymbol{r}, \boldsymbol{\tau_1}, \boldsymbol{\tau_2}, \boldsymbol{s}, \boldsymbol{\theta}^*_{p(r;\alpha,\beta)}),$$
$$\text{s.t.} \quad \alpha, \beta > 0.$$

With such an assumption, we find it is feasible to effectively solve $p^*(r)$ with a simple greedy search algorithm [3]. Specifically, we begin by performing a line search to optimize one parameter while keeping the other fixed. Once we find an optimal value for the first parameter, we switch to the second parameter and conduct a line search again to find its optimal value. This optimization continues until there is no further improvement in the performance metric.

5. Experiments

Implementation details. Following [6, 7, 27], we employ a pretrained VQGAN [10] with a codebook of size 1024 for the conversion between images and visual tokens. The architecture of our models follows U-ViT [1], a type of Transformer adapted for image generation tasks. We consider two model configurations: a small model (13 layers, 512 embedding dimensions, denoted as AutoNAT-S) and a large model (25 layers, 768 embedding dimensions, denoted as AutoNAT-L). For ImageNet-512, we adopt a patch size of 2 to accommodate the increased token count. In the implementation of AutoNAT, we first adopt the small model and perform alternating optimization on ImageNet-256 ($T = 4$) for three iterations to obtain our basic training and generation strategy, where we utilize FID [17] as the default evaluation metric F. When applying the basic strategy to different datasets and network architectures, we conduct a single additional search for only the generation strategy. This implementation technique slightly improves the performance of AutoNAT by fine-tuning it conditioned on the specific scenario (see Table 6d for ablation studies). Notably, adjusting the generation strategy is more efficient since it only needs to infer the model.

5.1. Main Results

Class-conditional generation on ImageNet. In Table 3, we present a comparison of our method against state-of-the-art generative models on ImageNet 256x256 and ImageNet 512x512. The number of generation steps, model pa-

Method	Type	ImageNet-256					ImageNet-512				
		#Params	Steps	TFLOPs↓	FID↓	IS↑	#Params	Steps	TFLOPs↓	FID↓	IS↑
VQVAE-2‡ [37] *(NeurIPS'19)*	AR	13.5B	5120	-	31.1	∼ 45	-	-	-	-	-
VQGAN‡ [10] *(CVPR'21)*	AR	1.4B	256	-	15.78	78.3	227M	1024	-	26.52	66.8
ADM-G [9] *(NeurIPS'21)*	Diff.	554M	250	334.0	4.59	186.7	559M	250	579.0	7.72	172.7
ADM-G, ADM-U [9] *(NeurIPS'21)*	Diff.	608M	250	239.5	3.94	215.8	731M	250	719.0	3.85	221.7
LDM [38] *(CVPR'22)*	Diff.	400M	250	52.3	3.60	247.7	-	-	-	-	-
VQ-Diffusion‡ [15] *(CVPR'22)*	Diff.	554M	100	12.4	11.89	-	-	-	-	-	-
Draft-and-revise [24] *(NeurIPS'22)*	NAT	1.4B	72	-	3.41	224.6	-	-	-	-	-
Efficient likelihood-based image generation models (†: augmented with DPM-Solver [30] *(NeurIPS'22)*)											
ADM-G† [9] *(NeurIPS'21)*	Diff.	554M	4	5.3	22.35	-	559M	4	9.3	42.85	-
			16	21.4	5.28	214.8		16	37.1	8.8	157.2
LDM† [38] *(CVPR'22)*	Diff.	400M	4	1.2	11.74	-	-	-	-	-	-
			16	3.7	3.68	202.7		-	-	-	-
U-ViT-H† [1] *(CVPR'23)*	Diff.	501M	4	1.4	8.45	-	501M	4	2.3	8.29	-
			16	4.6	2.77	259.5		16	5.5	4.04	252.6
DiT-XL† [33] *(ICCV'23)*	Diff.	675M	4	1.3	9.71	-	675M	4	5.4	11.72	-
			16	4.1	3.13	256.1		16	18.0	3.84	211.7
MaskGIT‡ [6] *(CVPR'22)*	NAT	227M	8	0.6	6.18	182.1	227M	12	3.3	7.32	156.0
Token-Critic‡ [25] *(ECCV'22)*	NAT	422M	36	1.9	4.69	174.5	422M	36	7.6	6.80	182.1
MaskGIT [6] *(CVPR'22)*	NAT	227M	8	-	4.02	-	227M	12	-	4.46	-
Token-Critic [25] *(ECCV'22)*	NAT	422M	36	-	3.75	-	422M	36	-	4.03	-
MAGE‡ [27] *(CVPR'23)*	NAT	230M	20	1.0	6.93	-	-	-	-	-	-
AutoNAT-L‡	NAT	194M	12	0.7	4.45	193.3	199M	12	1.0	6.36	185.0
AutoNAT-S	NAT	46M	4	**0.2**	4.30	249.7	48M	4	**0.5**	6.59	252.9
			8	0.3	3.52	253.4		8	0.6	5.06	254.5
AutoNAT-L	NAT	194M	4	0.5	3.26	271.3	199M	4	0.8	5.37	278.7
			8	0.9	**2.68**	278.8		8	1.2	**3.74**	286.2

Table 3. **Class-conditional image generation on ImageNet-256 and ImageNet-512**. TFLOPs represents the number of floating-point operations required for generating a single image. We calculate FID-50K following [9, 33]. For DPM-Solver [29] augmented diffusion models (marked with †), we follow instructions in [29] to tune all solver configurations as well as classifier-free guidance scale, and report results with the lowest FID. ‡: methods without classifier-based or classifier-free guidance [9, 18]. Rows in gray use compute-intensive classifier-based rejection sampling. Diff: diffusion, AR: autoregressive, NAT: non-autoregressive Transformers.

rameters and the total computational cost during generation (measured in TFLOPs) are also reported to give a comprehensive evaluation of the efficiency-effectiveness tradeoff. Moreover, in Table 3, the models specialized in efficient image synthesis, *e.g.*, recently proposed non-autoregressive Transformers and diffusion models with advanced samplers [29, 30], are grouped together for a direct comparison with our method.

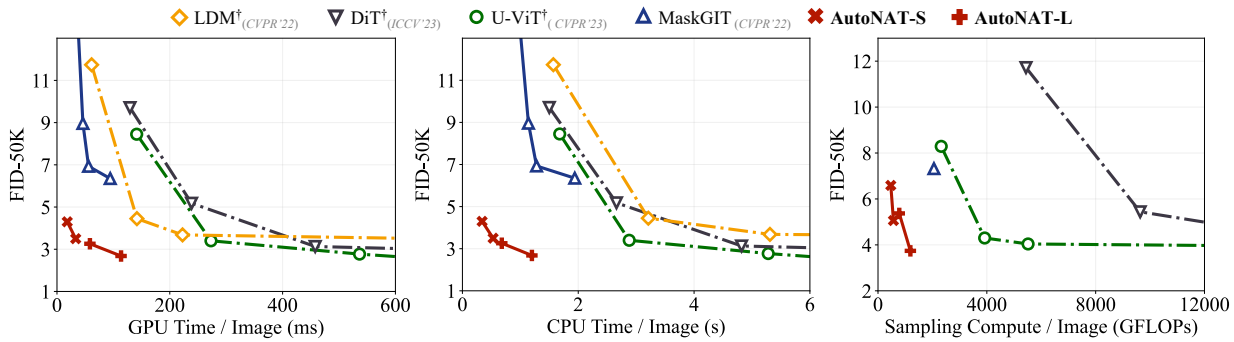
The results in Table 3 show that AutoNAT-S, although having notably fewer parameters than other baselines, yields competitive performance on ImageNet-256. For example, AutoNAT-S requires only 0.2 TFLOPs and 4 synthesis steps to achieve an FID of 4.30. With a slightly increased computational budget of 0.3 TFLOPs, the FID obtained by AutoNAT-S is further improved to 3.52, surpassing most of the baselines. Furthermore, the larger AutoNAT continues this trend, attaining an FID of 2.68 with 8 steps. On ImageNet-512, our best model achieves an FID of 3.74, exceeding other state-of-the-art models while utilizing significantly fewer computational resources.

In addition, we present more comprehensive comparisons of the trade-off between generation quality and computational cost in Figs. 1 and 4. Importantly, both the theoretical GFLOPs and the practical GPU/CPU latency required for generating an image are reported. The results demonstrate that our method consistently outperforms other baselines in terms of both generation quality and computational cost. Compared to the strongest diffusion models with fast samplers, AutoNAT offers approximately a $5\times$ inference speedup without compromising performance. Qualitative results of our method are presented in Figure 5.

Method	Type	#Params	Steps	TFLOPs↓	FID↓
VQ-Diffusion [15]	Diff.	370M	100	-	13.86
Frido [11]	Diff.	512M	200	-	8.97
U-Net† [1]	Diff.	53M	50	-	7.32
U-ViT† [1]	Diff.	58M	4	0.4	11.88
U-ViT† [1]	Diff.	58M	50	2.2	5.48
AutoNAT-S	NAT	45M	8	**0.3**	**5.36**

Table 4. **Text-to-image generation on MS-COCO**; all models are trained and evaluated on MS-COCO. We report FID-30K following [1]. †: augmented with DPM-Solver [29, 30].

Text-to-image generation on MS-COCO. We further evaluate the effectiveness of AutoNAT in the text-to-image



(a) FID *vs.* GPU Time on **ImageNet-256** (b) FID *vs.* CPU Time on **ImageNet-256** (c) FID *vs.* Compute on **ImageNet-512**

Figure 4. **Sampling efficiency** on ImageNet-256 and ImageNet-512. LDM is not included in ImageNet-512 results as it is only trained on ImageNet-256. GPU time is measured on an A100 GPU with batch size 50. CPU time is measured on Xeon 8358 CPU with batch size 1. †: DPM-Solver [29] augmented diffusion models.

Method	Type	#Params	Steps	TFLOPs↓	FID↓
VQGAN [10]	AR	600M	256	-	28.86
LDM-4 [38]	Diff.	645M	50	-	17.01
RQ-Transformer [23]	AR	654M	64	-	12.33
Draft-and-revise [24]	NAT	654M	72	-	9.65
Muse [7]	NAT	500M	8	2.8	7.67
			16	5.4	7.01
AutoNAT-Muse [7]	NAT	500M	8	**2.8**	**6.90**

Table 5. **Text-to-image generation on CC3M**; all models are trained and evaluated on CC3M. We report FID-30K following [7].

generation scenario, using the MS-COCO [28] benchmark. As summarized in Table 4, AutoNAT-S is able to outperform other baselines with a minimal 0.3 TFLOPs compute and achieves a leading FID score of 5.36, indicating its superior efficiency and effectiveness. When compared to the latest diffusion models equipped with a fast sampler [1], AutoNAT-S surpasses its 50-step sampling results while requiring 7× less computational cost, and outperforming it by large margins with similar computational resources.

Text-to-image generation on CC3M. In Table 5, we validate the effectiveness of AutoNAT for text-to-image generation on the larger CC3M dataset [44]. Here its efficiency is mainly evaluated on top of the recently proposed non-autoregressive Transformer model: Muse [7]. We apply AutoNAT to the pre-trained Muse model, and only optimize the generation strategy. As indicated in Table 5, combining our optimized strategy with the Muse model yields an FID of 6.90 on CC3M, surpassing other baselines notably. Compared to the vanilla Muse model, AutoNAT outperforms it with half of the computational cost and achieves significantly better FID (6.90 *vs.* 7.67) when utilizing the same computational resources.

5.2. Analytical Results

In this section, we provide more analytical results of our method. Unless otherwise specified, we adopt AutoNAT-S trained on ImageNet-256 in all experiments.

Contribution of training and generation strategies. Our method optimizes both training and generation strategies by default, and we conduct a more fine-grained ablation study in Table 6a to assess the contribution of each component. Removing the optimized training strategy has a noticeable detrimental impact on the FID score (an increase of 0.91). However, ablating the optimized generation strategy has an even more significant negative effect, deteriorating the FID by a substantial 3.58. This indicates that while both components are important, the previous non-autoregressive models had more room for improvement in generation strategies

Optimization efficacy. Our AutoNAT algorithm is iterative, yet as evidenced in Table 6b, it rapidly converges to a decent solution within a few iterations. Compared to the baseline FID of 8.40, a single optimization iteration markedly improves the FID to 4.61. Within just three iterations, the algorithm largely converges, evidenced by a minimal FID difference of 0.05 between the last two iterations, culminating in a performance of 4.30 FID, substantially exceeding the baseline. This underscores AutoNAT's effectiveness in optimizing training and generation configurations for non-autoregressive Transformers.

Alternating *vs*. concurrent optimization. We compare our alternating optimization strategy against optimizing the training and generation strategy concurrently (denoted as Concurrent) in Table 6c. Optimization efficiency is quantified in terms of computational cost measured by training time on a single A100 node. Our alternating optimization method demonstrates significantly better performance (4.30 *vs.* 8.01) under similar computational costs. Even when the resource allocation for concurrent optimization was doubled, its performance still remained considerably worse than that of our method. This observation demonstrates the effectiveness of our proposed alternating optimization in facilitating more rapid and robust convergence.

Transferability of the searched strategies. In Table 6d, we examine the transferability of strategies developed on our smaller model to larger models. To investigate this issue

Figure 5. **Selected visualizations of AutoNAT**. Samples are generated in 8 steps with AutoNAT-L on ImageNet-512 and ImageNet-256.

Train Strategy	Gen. Strategy	FID ↓
✓	✓	**4.30**
	✓	5.21 (+0.91)
✓		7.88 (+3.58)
		8.40 (+4.10)

(a) **Optimization contribution**. Assessing the effect of optimized training and generation strategies on FID scores against baselines.

Optimization Steps	FID ↓
-	8.40
1	4.61
2	4.35
3	**4.30**

(b) **Optimization efficacy**. Our optimization process quickly converges and surpasses the baseline results by a large margin.

Method	Cost ↓	FID ↓
Alternating	1.9 days	**4.30**
Concurrent	2.0 days	8.01
Concurrent	4.0 days	7.22

(c) **Alternating *vs*. concurrent optimization**. Alternating optimization is faster and more effective. Cost measured in one A100 node.

Model	Strategy	Steps=4	Steps=8
Base	Baseline	6.56	5.11
	Optimized	**3.83**	**2.89**
	Transfer	3.87 (+0.04)	3.03 (+0.14)
Large	Baseline	5.42	4.67
	Optimized	**3.26**	**2.68**
	Transfer	3.36 (+0.10)	2.82 (+0.14)

(d) **Strategy transferability**. The strategies from smaller models (*Transfer*) exhibit competitive performance on larger models.

Re-masking ratios r	Sampling Temperatures τ_1	Re-masking Temperatures τ_2	Guidance scales s	FID ↓	Δ
✓	✓	✓	✓	**4.30**	-
	✓	✓	✓	4.58	+0.28
✓		✓	✓	6.57	+2.27
✓	✓		✓	5.68	+1.38
✓	✓	✓		5.43	+1.13

(e) **Hyperparameter impact on generation**. We exclude one of the hyperparameters from our optimization set and measure its impact on FID.

Table 6. **Ablation studies** on ImageNet-256. We report FID-50K. If not specified, we adopt our small model trained for 300K iterations and generate images in 4 steps. Default settings are marked in gray.

comprehensively, we introduced an additional "base" model (consisting of 12 layers with an embedding dimension of 768) and trained it with the same configuration as other models on ImageNet-256. The empirical results indicate that strategies formulated for smaller models are not only transferable but also highly effective when applied to larger models. Both directly transferred and re-optimized strategies for larger models significantly outperform the baseline at all generation steps. The re-optimized strategies show a slight advantage (about 0.1) over directly transferred strategies. This minor difference underscores the robust generalization capacity of the strategies across varying model sizes.

Impacts of each generation hyperparameter. Finally, we evaluated the impact of each hyperparameter by omitting one from our optimization set. As demonstrated in Table 6e, excluding any hyperparameter variably affects the model's performance. The re-masking ratios r exhibit a minor influence on performance, with a 0.28-point increase in FID when left unoptimized. Conversely, the most significant impact is observed when the sampling temperature τ_1 remains unoptimized, leading to a 2.27-point increase in FID. These findings indicate the extent of sub-optimality in current heuristically designed generation hyperparameters and may offer valuable insights for developing enhanced generation strategies in non-autoregressive Transformers.

6. Conclusion

In this paper, we investigated the optimal design of training and inference strategies for non-autoregressive Transformers (NATs) in image synthesis. Distinct from prior works, our approach circumvents the reliance on heuristic rules and manual fine-tuning by embracing a heuristic-free, automatic approach. This was achieved by formulating the optimal configuration of training and generation strategies as a unified hyperparameter optimization problem. We further proposed to solve this problem with a tailored alternating optimization algorithm for rapid convergence. Extensive experiments validated that our method, AutoNAT, significantly enhances the performance of NATs and achieves results comparable to the latest diffusion models while requiring substantially fewer computational resources.

Acknowledgements

This work is supported in part by the National Science and Technology Major Project (2022ZD0114903) and the National Natural Science Foundation of China (42327901, 62321005). The authors would like to sincerely thank Yang Yue for his insightful discussion.

References

[1] Fan Bao, Chongxuan Li, Yue Cao, and Jun Zhu. All are worth words: a vit backbone for score-based diffusion models. In *CVPR*, 2023. 1, 5, 6, 7

[2] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. In *ICLR*, 2022. 3

[3] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. 5

[4] Andrew Brock, Theodore Lim, James M Ritchie, and Nick Weston. Neural photo editing with introspective adversarial networks. In *ICLR*, 2017. 2

[5] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large scale gan training for high fidelity natural image synthesis. In *ICLR*, 2019. 2

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *CVPR*, 2022. 1, 2, 3, 4, 5, 6

[7] Huiwen Chang, Han Zhang, Jarred Barber, AJ Maschinot, Jose Lezama, Lu Jiang, Ming-Hsuan Yang, Kevin Murphy, William T Freeman, Michael Rubinstein, et al. Muse: Text-to-image generation via masked generative transformers. In *ICML*, 2023. 1, 2, 3, 4, 5, 7

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: pre-training of deep bidirectional transformers for language understanding. google ai language. In *NAACL*, 2019. 3

[9] Prafulla Dhariwal and Alexander Nichol. Diffusion models beat gans on image synthesis. In *NeurIPS*, 2021. 1, 2, 4, 6

[10] Patrick Esser, Robin Rombach, and Bjorn Ommer. Taming transformers for high-resolution image synthesis. In *CVPR*, 2021. 1, 2, 3, 5, 6, 7

[11] Wan-Cyuan Fan, Yen-Chun Chen, DongDong Chen, Yu Cheng, Lu Yuan, and Yu-Chiang Frank Wang. Frido: Feature pyramid diffusion for complex scene image synthesis. In *AAAI*, 2023. 6

[12] Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *EMNLP*, 2019. 3

[13] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *NeurIPS*, 2014. 2

[14] Jiatao Gu and Xiang Kong. Fully non-autoregressive neural machine translation: Tricks of the trade. In *ACL*, 2021. 3

[15] Shuyang Gu, Dong Chen, Jianmin Bao, Fang Wen, Bo Zhang, Dongdong Chen, Lu Yuan, and Baining Guo. Vector quantized diffusion model for text-to-image synthesis. In *CVPR*, 2022. 6

[16] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 3

[17] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. In *NeurIPS*, 2017. 4, 5

[18] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS Workshop*, 2021. 6

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020. 1

[20] Minguk Kang, Jun-Yan Zhu, Richard Zhang, Jaesik Park, Eli Shechtman, Sylvain Paris, and Taesung Park. Scaling up gans for text-to-image synthesis. In *CVPR*, 2023. 2

[21] Tero Karras, Samuli Laine, and Timo Aila. Analyzing and improving the image quality of stylegan. In *CVPR*, 2020. 2

[22] Wouter Kool, Herke Van Hoof, and Max Welling. Stochastic beams and where to find them: The gumbel-top-k trick for sampling sequences without replacement. In *ICML*, 2019. 3

[23] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, 2022. 7

[24] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and WOOK SHIN HAN. Draft-and-revise: Effective image generation with contextual rq-transformer. In *NeurIPS*, 2022. 6, 7

[25] José Lezama, Huiwen Chang, Lu Jiang, and Irfan Essa. Improved masked image generation with token-critic. In *ECCV*, 2022. 1, 3, 6

[26] Lijiang Li, Huixia Li, Xiawu Zheng, Jie Wu, Xuefeng Xiao, Rui Wang, Min Zheng, Xin Pan, Fei Chao, and Rongrong Ji. Autodiffusion: Training-free optimization of time steps and architectures for automated diffusion model acceleration. In *ICCV*, 2023. 2

[27] Tianhong Li, Huiwen Chang, Shlok Mishra, Han Zhang, Dina Katabi, and Dilip Krishnan. Mage: Masked generative encoder to unify representation learning and image synthesis. In *CVPR*, 2023. 1, 3, 5, 6

[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 2, 7

[29] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver: A fast ode solver for diffusion probabilistic model sampling in around 10 steps. In *NeurIPS*, 2022. 1, 2, 6, 7

[30] Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *arXiv preprint arXiv:2211.01095*, 2022. 1, 2, 6

[31] Simian Luo, Yiqin Tan, Longbo Huang, Jian Li, and Hang Zhao. Latent consistency models: Synthesizing high-resolution images with few-step inference. *arXiv preprint arXiv:2310.04378*, 2023. 2

[32] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *ICLR*, 2018. 2

[33] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *ICCV*, 2023. 1, 6

[34] Shengju Qian, Huiwen Chang, Yuanzhen Li, Zizhao Zhang, Jiaya Jia, and Han Zhang. Strait: Non-autoregressive generation with stratified image transformer. *arXiv preprint arXiv:2303.00750*, 2023. 1, 3, 4

[35] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *ICML*, 2021. 2

[36] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 2022. 1, 2

[37] Ali Razavi, Aaron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with vq-vae-2, 2019. 3, 6

[38] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1, 2, 6, 7

[39] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016. 5

[40] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. In *IJCV*, 2015. 1, 2, 3

[41] Chitwan Saharia, William Chan, Saurabh Saxena, Lala Li, Jay Whang, Emily L Denton, Kamyar Ghasemipour, Raphael Gontijo Lopes, Burcu Karagol Ayan, Tim Salimans, et al. Photorealistic text-to-image diffusion models with deep language understanding. In *NeurIPS*, 2022. 2

[42] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. In *NeurIPS*, 2016. 4

[43] Axel Sauer, Dominik Lorenz, Andreas Blattmann, and Robin Rombach. Adversarial diffusion distillation. *arXiv preprint arXiv:2311.17042*, 2023. 2

[44] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *ACL*, 2018. 2, 7

[45] Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *ICLR*, 2021. 1, 2

[46] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, 2017. 1, 3

[47] Daniel Watson, William Chan, Jonathan Ho, and Mohammad Norouzi. Learning fast samplers for diffusion models by differentiating through sample quality. In *ICLR*, 2021. 2

[48] Tianwei Yin, Michaël Gharbi, Richard Zhang, Eli Shechtman, Fredo Durand, William T Freeman, and Taesung Park. One-step diffusion with distribution matching distillation. *arXiv preprint arXiv:2311.18828*, 2023. 2

[49] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *TMLR*, 2022. 1, 2

[50] Lili Yu, Bowen Shi, Ramakanth Pasunuru, Benjamin Muller, Olga Golovneva, Tianlu Wang, Arun Babu, Binh Tang, Brian Karrer, Shelly Sheynin, et al. Scaling autoregressive multimodal models: Pretraining and instruction tuning. *arXiv preprint arXiv:2309.02591*, 2023. 2

[51] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, et al. Language model beats diffusion–tokenizer is key to visual generation. In *ICLR*, 2024. 3