Nemotron-Labs-Diffusion-Image: Advancing Masked Discrete Diffusion for High-Resolution Image Synthesis

Shufan Li[1,2], **Greg Heinrich**[1], **Hanrong Ye**[1], **Yonggan Fu**[1], **Aditya Grover**[2], **Jan Kautz**[1], **Pavlo Molchanov**[1]
[1] NVIDIA [2] UCLA

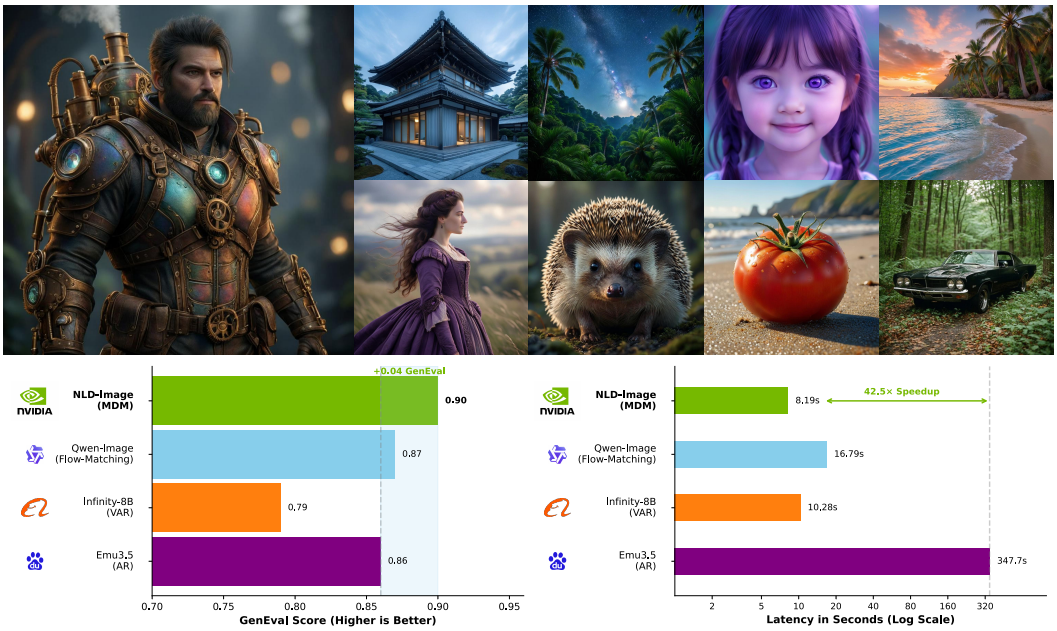


Figure 1: **We propose Nemotron-Labs-Diffusion-Image(NLD-Image),** a masked discrete diffusion model for text-to-image synthesis. It achieves state-of-the-art performance at 1024px text-to-image synthesis, surpassing prior masked image generators. Images are sampled from prompts of the MJHQ dataset.

Abstract

We propose Nemotron-Labs-Diffusion-Image, a state-of-the-art masked discrete diffusion model (MDM) for high-resolution text-to-image synthesis. Compared with prior work on masked image generation, Nemotron-Labs-Diffusion-Image addresses two key challenges. First, unlike continuous diffusion models which progressively refine latent representations across the entire image, standard MDMs lack self-correcting capability because discrete tokens cannot be modified once they are unmasked. Second, although increasing the vocabulary size of discrete image tokenizers improves reconstruction fidelity, it introduces optimization difficulties for generative modeling as the per-token training signal becomes increasingly sparse. To address the first challenge, Nemotron-Labs-Diffusion-Image incorporates a token-editing mechanism that enables the model to dynamically revise already-

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

unmasked tokens during inference, similar to how a sculptor iteratively refines their work. To tackle the second challenge, we propose a Grouped Cross-Entropy (GCE) objective that assigns positive learning signals to tokens neighboring the ground truth in embedding space, thereby alleviating signal sparsity. To further improve training efficiency, we implement a custom fused operator for GCE that significantly reduces VRAM usage in large-vocabulary settings. Experimental results demonstrate that these innovations substantially improve both training efficiency and image fidelity of masked discrete image generators, achieving a score of 0.90 on GenEval, 86.9 on DPG and 10.76 of HPSv3.

1 Introduction

State-of-the-art text-to-image models [36, 23, 56, 46] achieve high-fidelity image synthesis primarily by scaling latent diffusion models (LDMs) [41] to large model sizes trained on massive text-image corpora. Compared with earlier alternatives such as GANs [18], the success of LDMs can largely be attributed to two key factors. First, VAEs provide a smooth and structured latent space that is easier to model than raw image pixels, making training more effective and scalable. Second, unlike GANs, which generate all pixels of an image in a single step, LDMs iteratively refine latent embeddings over multiple denoising steps, enabling progressive self-correction during inference. for people with LLM background, we need to put a table together showing different approaches to image generation: continuous, discrete, AR, MDM etc and highlight pros and cons, like quality, efficiency, training etc.

Recently, there has been growing interest in discrete image generation, where images are first discretized into sequences of tokens and a generative model is trained to model the distribution over these token sequences [1, 6, 55]. Compared with LDMs, discrete image generation offers several promising advantages. For example, discrete representations are naturally compatible with large language models (LLMs), which also operate on discrete tokens, making them an appealing foundation for unified multimodal models [27, 30, 59, 49]. Furthermore, they can directly leverage well-established optimization techniques developed for LLM training and inference, such as sequence packing and pre-tokenization during training [22], as well as KV caching during inference, improving scalability and efficiency [28, 32].

Existing discrete image generators can be categorized into two families: autoregressive (AR) models and masked discrete diffusion models (MDMs). AR models generate image tokens sequentially following a raster-scan order, whereas MDMs start from a fully masked sequence and progressively unmask tokens over multiple diffusion steps. Importantly, MDMs can decode multiple tokens at arbitrary positions in parallel during each inference step, providing several advantages over AR models, including faster inference and native support for tasks such as image inpainting. As a result, recent state-of-the-art discrete image generators [12, 57] predominantly adopt the MDM paradigm.

In this work, we propose Nemotron-Labs-Diffusion-Image, a state-of-the-art discrete image generator based on the masked diffusion paradigm. Compared with prior work, Nemotron-Labs-Diffusion-Image introduces several novel techniques that address two fundamental challenges in discrete image generation: (1) the lack of self-refinement capability and (2) the difficulty of training discrete image generators with large codebooks, which are essential for achieving high-fidelity image generation [64, 7].

The first challenge is the lack of self-refinement. Unlike LDMs, which iteratively refine outputs during inference, vanilla MDMs commit to each token once it is unmasked, preventing later correction even if the prediction is incorrect. This issue is particularly severe because tokens unmasked at each step are sampled independently from position-wise logits. Consequently, the effective joint distribution becomes a product of marginal token distributions, implicitly assuming inter-token independence. In practice, however, image tokens exhibit strong dependencies, creating a mismatch between the sampling process and the true data distribution, which leads to error accumulation in the final output.

To address this issue, we introduce a token editing mechanism that allows tokens to be corrected and refined even after they have been unmasked. We enable this capability by modifying the vanilla MDM training process with token corruption. Specifically, in addition to masking a subset of image tokens, Nemotron-Labs-Diffusion-Image also corrupts a portion of visible tokens and trains the model to predict token probabilities at all positions. For masked positions, the model predicts new tokens as in standard MDMs; for non-masked positions, it predicts whether existing tokens should be corrected.

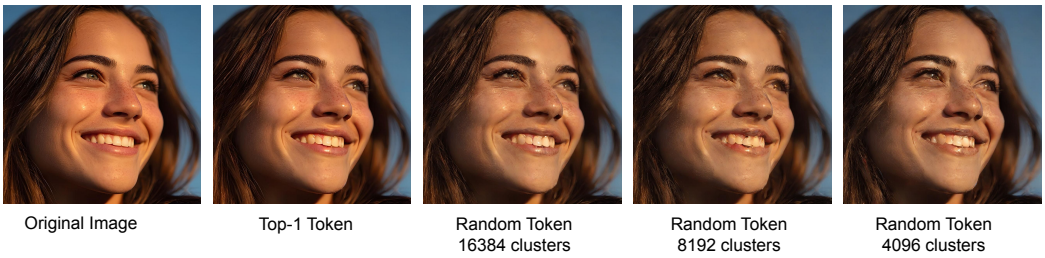


Figure 2: **Semantic similarities of adjacent tokens.** Using precomputed K-means clustering, we visualize the reconstruction results of the top-1 token and randomly selected tokens from the same cluster. The results show that these tokens are semantically similar, yet this relationship is not captured by the vanilla cross-entropy loss.

This design enables iterative refinement similar to continuous LDMs, reducing accumulated sampling errors and improving image fidelity.

The second challenge concerns the difficulty of training generative models with large and expressive codebooks. Discrete image tokenizers typically use vector quantization to map continuous image features to discrete codes. Larger codebooks reduce discretization error and improve reconstruction fidelity [47, 64], but they also make downstream generative modeling substantially more difficult, often requiring larger model capacity and more training data [12]. One key reason is the codebook sparsity problem [26]: with a fixed image corpus, increasing the codebook size decreases the frequency of individual tokens, resulting in sparse and insufficient supervision signals. Two visually similar features that would map to the same code in a smaller codebook may instead map to different codes in a larger one. However, the standard one-hot cross-entropy objective fails to capture this semantic similarity and instead treats all non-ground-truth tokens as equally negative targets, creating optimization challenges. Figure 2 illustrates this phenomenon visually.

To mitigate this problem, we propose a grouped cross-entropy (GCE) objective that provides auxiliary supervision for non–top-1 tokens that are semantically close to the ground-truth token in embedding space. Given a large codebook (e.g., 100k entries), we first cluster the codes into a smaller set of groups (e.g., 16k) using K-means. During training, the model predicts probabilities over all tokens, and cluster-level probabilities are obtained by summing the probabilities of tokens within each cluster. We then apply a cross-entropy loss on the resulting cluster distribution using the ground-truth cluster label, alongside the standard token-level one-hot cross-entropy loss. We further extend this hierarchically with multiple clustering granularities, yielding multi-level supervision that encourages the model to capture semantic similarity beyond exact token matches. To improve efficiency, we implement a custom fused operator that significantly reduces the latency and VRAM overhead compared to a naive PyTorch autograd implementation.

We conduct extensive evaluations of Nemotron-Labs-Diffusion-Image, and experimental results demonstrate strong performance across a wide range of text-to-image benchmarks, including GenEval [17], DPG [21], and MJHQ-30k [25], highlighting the effectiveness of the proposed approaches.

In summary, our contributions are as follows: 1) We propose Nemotron-Labs-Diffusion-Image, an 8B state-of-the-art foundational masked discrete diffusion model for text-to-image synthesis, achieving strong performance across multiple text-to-image benchmarks. 2) We introduce a token editing mechanism that enables MDMs to iteratively refine and self-correct their outputs during inference, improving image fidelity. 3) We propose GCE, a novel training objective for large-vocabulary discrete image generators that alleviates the codebook sparsity problem and improves training efficiency and optimization stability.

2 Background and Related Work

2.1 Discrete Image Generation

Discrete image generators employ discrete image tokenizers [64, 34, 62, 47, 7] to encode images into sequences of discrete codes and then learn to model these sequences. Early works primarily

adopted autoregressive models [53, 40, 61, 51, 11, 55], which generate tokens sequentially. A key limitation of these models is their slow inference speed. To address this limitation, masked discrete diffusion models (MDMs) [43, 48] learn to generate multiple tokens at each inference step, greatly reducing latency. MaskGIT [6] pioneered masked generative image modeling. Meissonic [1] further scaled this paradigm to high-resolution image synthesis through token compression. Several recent works have also explored unified understanding and generation models based on the discrete diffusion paradigm, including MMaDa [59], the LaViDa-O series [27, 28, 30], and Unidisc [20].

Concretely, MDMs begin inference from a fully masked sequence y^1 consisting entirely of the special mask token [M]. The model then gradually unmasks tokens over multiple sampling steps until a clean sequence y^0 is obtained. At intermediate timesteps $0 < t < 1$, the sequence y^t contains a mixture of mask tokens and clean tokens. During training, given a clean sequence y^0, a random timestep t is sampled uniformly from $[0, 1]$, and a partially masked sequence y^t is generated using the forward diffusion process $q(y^t|y^0)$, which randomly masks a subset of tokens in y^0. The model is then trained to predict the original tokens at masked positions using the following ELBO lets say what ELBO is objective:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{y^0,\, t\sim\text{Unif}([0,1]),\, y^t\sim q(y^t|y^0)} \left[-\frac{1}{t} \sum_{i=1}^{L} \mathbf{I}\{y_i^t = [\text{M}]\} \log p_\theta(y_i^0 \mid y^t) \right] \tag{1}$$

2.2 Token Editing

During training, the objective in Equation 1 computes the loss only on masked positions while treating non-mask tokens as correct. During inference, once a token is unmasked, it remains fixed, preventing MDMs from revising earlier decisions and leading to error accumulation. Recent works in the language domain have introduced self-correction mechanisms to address this limitation. Seed-Diffusion [50] augments the forward process $q(y^t|y^0)$ with insertion and deletion operations and employs on-policy learning with Levenshtein-distance rewards. EditFlow [19] extends MDM sampling with CMTC-based insertion and deletion operations. LLaDa-2.1 [4] enables direct modification of previously unmasked tokens without explicit edit operations. However, extending token editing to discrete image generation remains largely unexplored. Image token sequences have fixed lengths, making insertion and deletion operations invalid, and it remains unclear which corruption strategies are effective for image synthesis. Nemotron-Labs-Diffusion-Image is the first work to introduce token-editing mechanisms for discrete image generation.

2.3 Scaling Vocabulary Size

Scaling codebook size is crucial for improving image fidelity in discrete image generation. Larger vocabularies make tokenizers more expressive, especially for large-scale, high-resolution text-to-image models [12, 52]. However, large codebooks suffer from codebook sparsity: as vocabulary size grows, per-token frequency drops sharply. With a dataset of 1M images and 256 tokens per image, an 8,192-entry codebook yields an average frequency of 31,250 per code, but a 200K codebook reduces this to 1,280—only 4% of the former. This sparsity makes training large-vocabulary generators difficult. Existing work mainly relies on brute-force scaling with >20B-parameter models and >13T tokens of data [12]. Concurrent work notes that discrete tokenizers underperform with limited data but improve as data scales [52]. SNCE [26] mitigates sparsity via distance-based soft labels, but gains are limited and soft labels increase memory cost. We propose a more efficient and effective GCE objective to address codebook sparsity.

3 Method

3.1 Model Architecture

Prior works on discrete text-to-image models such as Meissonic [1] typically employ an encoder-decoder framework that combines a frozen text encoder with a trainable image generation network. By contrast, Nemotron-Labs-Diffusion-Image uses a single decoder-only transformer to process both text prompts and image tokens. This design is motivated by the recent success of unified understanding and generation models [13, 59], which demonstrate that a single transformer can

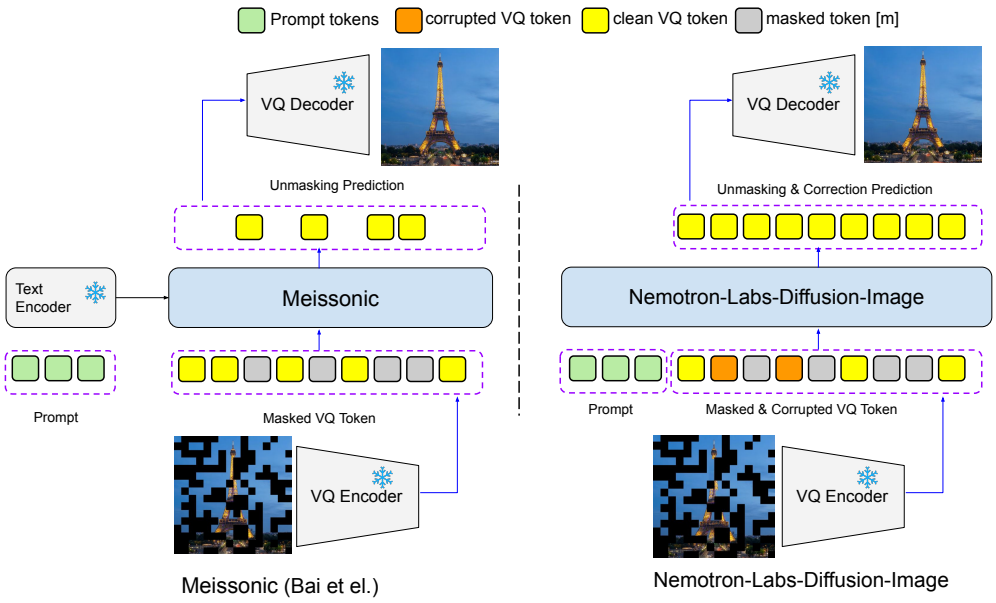


Figure 3: **Overall architecture of Nemotron-Labs-Diffusion-Image.** Unlike prior works such as Meissonic, Nemotron-Labs-Diffusion-Image employs a single decoder-only transformer to process both text and image inputs. During training, it additionally corrupts clean image tokens to enable self-correction. *Corruption effects are amplified for better visual illustration. Figure 2 provides a more accurate depiction.

effectively model both text and images. In our experiments, we initialize our model from a pretrained diffusion language model [16], which is trained with masked modeling objectives on text tokens.

Compared with the encoder-decoder approach, Nemotron-Labs-Diffusion-Image's decoder-only architecture offers several advantages. First, Nemotron-Labs-Diffusion-Image is not constrained by the context length of a frozen text encoder, allowing it to process substantially longer prompts. By contrast, Meissonic's CLIP encoder supports a maximum of only 77 tokens per prompt. Second, Nemotron-Labs-Diffusion-Image can naturally leverage sequence packing optimization [22], which is widely used in LLM pretraining to reduce unnecessary padding when prompts have varying lengths, thereby improving training efficiency. Third, by initializing Nemotron-Labs-Diffusion-Image from a pretrained diffusion language model, we can effectively utilize the language understanding capabilities of the base model and avoid learning text semantics from scratch, making training more efficient and stable. To adapt the base model for image generation, we replace the final language modeling head with a newly initialized MLP that predicts image tokens. We defer additional implementation details to Appendix B. In total, Nemotron-Labs-Diffusion-Image contains 8B parameters optimized end-to-end.

3.2 Self-Correction via Token Editing

To equip Nemotron-Labs-Diffusion-Image with self-correction capabilities similar to those of continuous diffusion models, we introduce a token editing mechanism that enables the model to iteratively refine its outputs. Specifically, instead of predicting token probabilities only at masked positions, Nemotron-Labs-Diffusion-Image also predicts a "correction distribution" for clean image tokens.

Inference. At each inference step, given a partially unmasked sequence y^t, Nemotron-Labs-Diffusion-Image not only performs the standard unmasking process but also edits already unmasked tokens based on a confidence threshold. Specifically, if the model predicts a different token at an already unmasked position with confidence above a specified threshold τ, we replace the original token with the newly predicted token. This process largely follows the setup of LLaDa-2.1 [4], a masked diffusion language model.

Training. To enable token editing capabilities, we modify the unmasking objective in Equation 1 to incorporate a token-editing component. Specifically, given a clean sequence y^0 of length L, vanilla

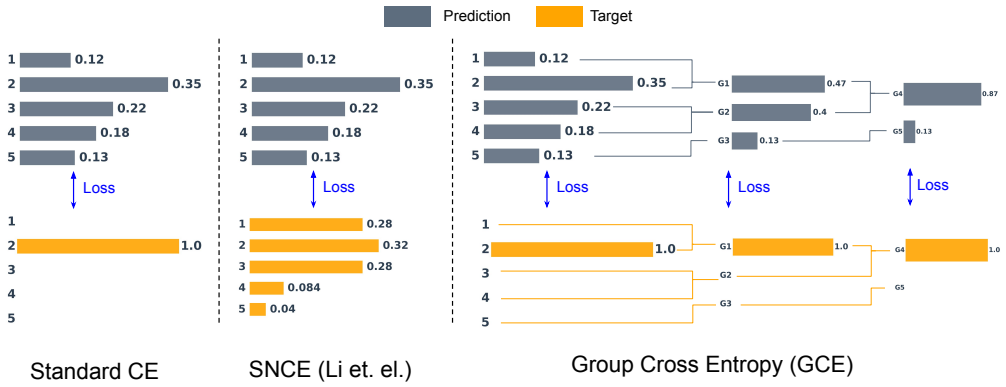


Figure 4: **Design of grouped cross-entropy (GCE).** (Left) Standard cross-entropy uses a one-hot target. (Middle) SNCE employs a fixed soft target. (Right) GCE groups codes into clusters and applies one-hot cross-entropy at multiple hierarchical levels.

MDM training first samples a timestep $t \in [0, 1]$ uniformly at random and then samples a partially masked sequence from the forward distribution $q(y^t|y^0)$. This is implemented by randomly replacing each clean image token with the special mask token [M] with probability p. In expectation, y^t contains Lp masked tokens and $L(1-p)$ clean image tokens. Nemotron-Labs-Diffusion-Image extends this process by introducing additional corruption to the $L(1-p)$ clean tokens instead of directly copying them from the clean sequence y^0.

An important distinction from similar approaches in the language domain lies in the choice of corruption operations. In text generation, corruption often includes edit operations such as insertion and deletion [14]. While these operations are natural for variable-length text generation, they are not applicable to image synthesis because the number of image tokens is fixed for a given resolution. Another common strategy is to replace clean tokens with random tokens uniformly sampled from the vocabulary [63]. However, we find this approach performs poorly in practice because randomly sampled tokens follow a substantially different distribution from incorrectly predicted tokens, leading to a mismatch between training and inference.

Instead, we employ two corruption strategies. In the first strategy, clean tokens are replaced with tokens randomly sampled from the same image. In the second strategy, clean tokens are replaced with random tokens selected from their nearest neighbors in the tokenizer embedding space. The overall training paradigm is illustrated in Figure 3. Formally, we denote this augmented forward process as $q'(y^t|y^0)$. The final objective is defined as follows:

$$\mathcal{L}_{\text{ELBO}_{\text{Edit}}} = \mathbb{E}_{y^0,\, t\sim\text{Unif}([0,1]),\, y^t\sim q'(y^t|y^0)} \left[-\frac{1}{t} \sum_{i=1}^{L} \log p_\theta(y_i^0 \mid y^t) \right] \tag{2}$$

Compared with the vanilla MDM objective in Equation 1, we compute loss terms at all token positions rather than only masked positions. Additionally, y^t now contains corrupted tokens in addition to clean and masked tokens. We defer further implementation details to Appendix A.2.

3.3 Grouped Cross-Entropy Objective

To address the sparse training signals caused by large codebooks, we design a hierarchical grouped cross-entropy objective that provides positive supervision for non-top-1 tokens that are semantically similar to the ground-truth token. Given a large codebook V with size $|V|$, we first perform offline K-means clustering to group semantically similar tokens into M clusters $C_1, C_2, \ldots, C_M$. During training, the model predicts unnormalized logits $h \in \mathbb{R}^{|V|}$. The probability of each code $i \in \{1, 2, \ldots, |V|\}$ is obtained through the softmax operation $p_i = \frac{\exp(h_i)}{\sum_{j=1}^{|V|} \exp(h_j)}$. We then derive cluster probabilities by summing the probabilities of individual tokens belonging to the same cluster: $\mathbb{P}(C_i) = \sum_{j \in C_i} p_j$.

In Equation 2, the term $\log p_\theta(y_i^0 \mid y^t)$ is implemented using a cross-entropy loss with one-hot targets. Similarly, we introduce auxiliary losses at the cluster level by applying cross-entropy losses with one-hot cluster labels on the cluster probabilities defined above. In practice, we use a tokenizer with a vocabulary size of 132K and include multiple clustering granularities with different numbers of clusters (e.g., 16,384 and 8,192). Let $C^j(\cdot)$ denote the cluster assignment under the j-th clustering. The final GCE objective is defined as:

$$J_{\text{GCE}}(y_i^0, y^t) = \underbrace{\log p_\theta(y_i^0 \mid y^t)}_{\text{top-1 term}} + \underbrace{\sum_j \log \mathbb{P}\left(C^j(y_i^0) \mid y^t\right)}_{\text{clustering terms}} \tag{3}$$

During training, we replace $\log p_\theta(y_i^0 \mid y^t)$ in Equation 2 with J_{GCE}. For each clustering term in J_{GCE}, the gradient with respect to the unnormalized logits is given by:

$$\frac{\partial}{\partial h_i} \log P(C) = p_i \left(\frac{\mathbf{I}\{i \in C\}}{P(C)} - 1 \right) \tag{4}$$

We make several observations. First, within each clustering term, all codes belonging to the target cluster ($i \in C$) receive positive gradients, while all other codes receive negative gradients. Second, among tokens within the target cluster, the gradient magnitude is proportional to the post-softmax probability p_i, allowing the model to reinforce tokens that already have relatively high confidence. Third, when considering J_{GCE} as a whole, tokens receive positive supervision proportional to their semantic proximity to the ground-truth token. For example, the ground-truth token receives positive signals from all terms, while semantically similar non-top-1 tokens that belong to the same fine-grained clusters receive positive signals from both fine-grained and coarse-grained clustering terms. Tokens that are farther away receive positive supervision from only a subset of clustering levels. These properties make J_{GCE} particularly effective for mitigating the codebook sparsity problem. We visualize this design in Figure 4. The naive implementation of GCE based on PyTorch introduces considerable overhead. Hence, we designed a custom operator to accelerate GCE while reducing its memory overhead. We defer implementation details to Appendix A.3.

Comparison with SNCE. SNCE [26] replaces the one-hot target with distance-based soft labels, but GCE offers several advantages. First, SNCE uses a fixed smoothed target, whereas GCE allows the model to dynamically adjust its confidence. For example, if SNCE assigns a 0.7 soft target to the top-1 token but the model predicts 0.8, the top-1 token receives negative gradients, causing instability. In contrast, GCE always assigns positive gradients to the top-1 target, and its clustering terms can be optimized either by concentrating probability mass on the top-1 token or distributing it across tokens within the same cluster. Second, SNCE requires storing real-valued $L \times |V|$ soft targets, introducing additional memory overhead, while GCE stores only discrete cluster indices. Third, SNCE optimizes a different objective: even with perfect predictions, its smoothed-label loss remains non-zero. In GCE, once the top-1 term is perfectly optimized, both the top-1 and clustering terms become zero. These properties make GCE a more efficient and effective alternative to SNCE.

4 Experiments

We initialize our model from an 8B pretrained diffusion language model [16] and equip it with a tokenizer containing a 131K codebook [12]. Following prior work [1, 27], we employ a progressive upscaling strategy that starts training at 256×256 resolution and gradually scales to 1024×1024 during training. We train the model for 300K steps on 64 H100 GPUs. Additional training details and dataset composition are deferred to Appendix B. can we show training curves when initialized from scratch vs from pretrained weights?

4.1 Main Results on Large-Scale Text-to-Image Generation

We report text-to-image generation results on the GenEval [17], DPG [21], and MJHQ [25] benchmarks. These results are presented in Table 1 and Table 2. For GenEval and DPG, we report the corresponding benchmark scores. For MJHQ, we report both FID and HPSv3 metrics to evaluate

Table 1: **Text-to-Image Generation Performance on Geneval Benchmark.**

Model	Params	Single Obj.↑	Two Obj.↑	Counting↑	Colors↑	Position↑	Color Attri.↑	Overall↑
Unified MLLM								
Emu3 [55]	8B	-	-	-	-	-	-	0.66
Janus-Pro [11]	7B	0.99	0.89	0.59	0.90	0.79	0.66	0.80
MMaDA [59]	8B	0.99	0.76	0.61	0.84	0.20	0.37	0.63
Show-o [57]	1.3B	0.98	0.80	0.66	0.84	0.31	0.50	0.68
BAGEL [13]	14B	0.98	0.95	0.84	0.95	0.78	0.77	0.88
LaViDa-O [27]	10B	0.99	0.85	0.71	0.86	0.65	0.58	0.77
Show-o2 [58]	7B	1.00	0.87	0.58	0.92	0.52	0.62	0.76
Gen. Only								
PixArt-α [9]	0.6B	0.98	0.50	0.44	0.80	0.08	0.07	0.48
DALL-E 3 [35]	-	0.96	0.87	0.47	0.83	0.43	0.45	0.67
SD3-Medium [15]	2B	0.99	0.94	0.72	0.89	0.33	0.60	0.74
FLUX.1-dev [23]	12B	0.98	0.81	0.74	0.79	0.22	0.45	0.66
Meissonic [1]	1B	0.99	0.66	0.42	0.86	0.10	0.22	0.54
Qwen-Image-2507[56]	20B	0.99	0.92	0.89	0.88	0.76	0.77	0.87
Nemotron-Labs-Diffusion-Image	8B	0.98	0.93	0.83	0.94	0.88	0.82	0.90

Table 2: **Text-to-Image Generation Performance on DPG Benchmark and MJHQ-30k Dataset.** *Finetuned on 6M synthetic data for better image quality.

Model	Params	Codebook	DPG↑	MJHQ-30k	
				FID↓	HPSv3↑
SD3[15]	8B	-	83.5	11.92	9.42
GPT-4o [15]	-	-	85.3	-	-
Flux-Dev[23]	12B	-	-	10.15	-
Janus-Pro[11]	7B	16,384	84.1	10.10	8.81
Emu3 [55]	8B	32,678	80.6	-	-
Show-o [57]	1B	8,192	-	15.18	7.20
MMaDa[59]	8B	8,192	53.4	32.85	5.43
LaViDa-O [27]	10B	8,192	81.8	6.68	8.81
Nemotron-Labs-Diffusion-Image	8B	131,072	85.2	6.46	9.61
Nemotron-Labs-Diffusion-Image*	8B	131,072	86.9	12.23	10.76

image fidelity. We note that FID relies on an outdated feature extraction network trained on low-resolution images and therefore does not fully capture high-resolution image quality. Furthermore, multiple prior works have shown that FID correlates poorly with human perception of image quality [38, 8]. By contrast, HPSv3 [33] is a VLM-based reward model with a high-resolution image encoder specifically fine-tuned on human preference data, making it a more reliable indicator of image fidelity. We include FID primarily for completeness and consistency with prior literature.

We compare against specialist text-to-image models such as Flux-dev [23], SD3-Medium [15], Meissonic [1], and DALLE-3 [35], as well as unified multimodal models such as BAGEL [13], MMaDa [59], and LaViDa-O [27]. Among models based on the masked diffusion paradigm, Nemotron-Labs-Diffusion-Image significantly outperforms the state-of-the-art specialist model Meissonic as well as unified multimodal models such as LaViDa-O. Notably, Nemotron-Labs-Diffusion-Image achieves competitive performance with frontier models such as Qwen-Image-2507 on GenEval and GPT-4o on the DPG benchmark, highlighting the effectiveness of our proposed approach.

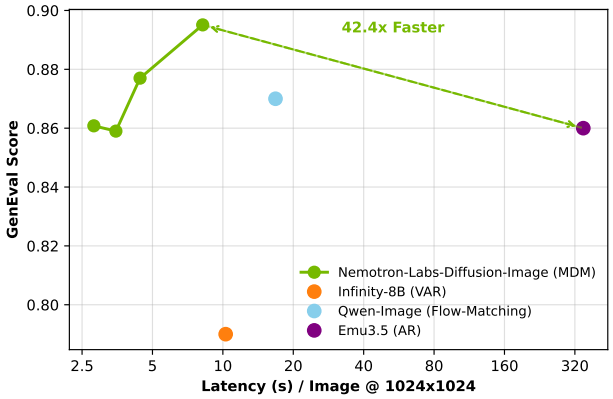
4.2 Ablations for Token Editing and Self-Correction

To validate the effectiveness of our proposed token editing pipeline and investigate whether self-correction improves image fidelity, we conduct both qualitative and quantitative evaluations on the MJHQ dataset. In Figure 5, we fix the random seed and visually compare images generated with and without the token editing pipeline. Token editing consistently improves image fidelity by correcting artifacts and refining texture details in the generated images.

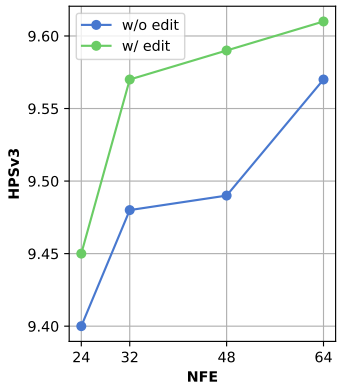
In Figure 6b, we report HPSv3 scores on the MJHQ dataset with and without token editing. We evaluate Nemotron-Labs-Diffusion-Image under different numbers of sampling steps, also known as the Number of Function Evaluations (NFEs). We draw two main conclusions from these results. First, token editing consistently improves image quality across all NFEs. Second, although reducing NFEs decreases image quality in both settings, the degradation is substantially smoother when token editing is enabled. Notably, generations with token editing at 32 NFEs achieve performance comparable to generations without token editing at 64 NFEs, representing an effective $2\times$ reduction in forward calls given a fixed target quality.



Figure 5: **Qualitative comparison of generated samples with and without token editing.**



(a) Comparision of GenEval score and inference latency between Nemotron-Labs-Diffusion-Image and other state-of-the-art models. Results are measured with a batch size of 1 on a H100 GPU.



(b) HPSv3 scores under different numbers of sampling steps with and without token editing.

Figure 6: **Speed-Quality Tradeoff of Nemotron-Labs-Diffusion-Image.**

4.3 Ablations for the Grouped Cross-Entropy Objective

To fairly evaluate the effectiveness of our proposed GCE objective and demonstrate its advantages over alternatives such as SNCE, we conduct extensive ablation studies in a controlled setting. Following the setup of SNCE [26], we perform experiments on class-conditional image generation using the ImageNet [42] dataset at 256×256 resolution. We use the exact same model architecture, tokenizer, and training schedule as SNCE [26], with the optimization objective being the only varying factor. Results are reported in Table 3. GCE consistently outperforms both SNCE and the vanilla cross-entropy baseline, demonstrating its effectiveness and advantages.

4.4 Compute Efficiency

Optimized Operator. To validate the effectiveness of our optimized operator, we benchmark both VRAM usage and latency when processing 16,384 tokens on a H100 GPU, corresponding to either 4 images at 1024×1024 resolution or 64 images at 256×256 resolution. We report latency, input tensor VRAM usage, and maximum VRAM consumption during both forward and backward

Table 3: **Class-conditioned Image Synthesis on ImageNet256 dataset.** *Models have identical-sized transformer layers. Parameter count increased due to larger token embedding and final linear head.

Objective	Params	Tokenizer	Tokenizer Pretraining	Codebook	Epoch	FID ↓
CE	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	100	7.53
SNCE	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	100	3.62
GCE	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	100	3.40
CE	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	300	5.44
SNCE	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	300	3.42
GCE	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	300	3.00
CE	846M*	FVQ [64]	ImageNet256	262,144	300	4.11
SNCE	846M*	FVQ [64]	ImageNet256	262,144	300	3.20
GCE	846M*	FVQ [64]	ImageNet256	262,144	300	2.69

Table 4: **Performance Comparison of Loss Function Implementations.**

Operation	Latency ↓	Input VRAM↓	Active VRAM↓	Max VRAM↓
`F.cross_entropy` (hard label)	12.71 ms	8.1 GB	8.1 GB	16.1 GB
`F.cross_entropy` (soft label)	25.00 ms	16.1 GB	8.1 GB	24.2 GB
GCE (eager)	44.14 ms	8.1 GB	17.1 GB	25.2 GB
GCE (optimized forward)	17.86 ms	8.1 GB	16.1 GB	24.2 GB
GCE (optimized fwd & bwd)	20.04 ms	8.1 GB	8.1 GB	16.1 GB

computation. We define active VRAM as the difference between maximum VRAM and input VRAM, which measures the additional memory overhead introduced by loss computation. For fairness, we enable `torch.compile` in all experiments. We report results in Table 4 Compared with the eager implementation, our optimized operator reduces latency from 44 ms to 20 ms while decreasing maximum VRAM usage from 25 GB to 16 GB.

Compared with the standard cross-entropy baseline using one-hot labels, our method introduces no additional memory overhead and only an 8 ms increase in latency. We note that this overhead is negligible given that the overall training time is approximately 3.2 s per step. All measurements are conducted on H100 GPUs.

Generation Latency. We also compare the inference latency of Nemotron-Labs-Diffusion-Image against state-of-the-art text=to-image models of including Qwen-Image (flow-matching), Infinity-8B (VAR) and Emu3.5 (AR). These results are visualized in Figure 6a. Compared with Emu3.5, Nemotron-Labs-Diffusion-Image is 42.4× faster while also acheveing higher GenEval score.

5 Conclusion

We introduced Nemotron-Labs-Diffusion-Image, a new state-of-the-art masked discrete diffusion model for high-resolution text-to-image synthesis. To overcome the inherent limitations of standard masked discrete models, Nemotron-Labs-Diffusion-Image incorporates a novel token-editing mechanism that enables iterative refinement during inference. In addition, we proposed the Grouped Cross-Entropy (GCE) objective, which alleviates the sparsity of training signals in large-vocabulary discrete token spaces by assigning positive supervision to semantically neighboring tokens in the embedding space. To further enhance practical scalability, we designed a custom fused operator for GCE that significantly reduces memory footprint and computational overhead, enabling efficient scaling. Experiments demonstrate that Nemotron-Labs-Diffusion-Image achieves substantial improvements in both image fidelity and training efficiency compared to prior masked image generation methods, paving the way for more powerful and scalable discrete image generation.

References

[1] Jinbin Bai, Tian Ye, Wei Chow, Enxin Song, Xiangtai Li, Zhen Dong, Lei Zhu, and Shuicheng Yan. Meissonic: Revitalizing masked generative transformers for efficient high-resolution text-to-image synthesis. *arXiv preprint arXiv:2410.08261*, 2024.

[2] Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

[3] Victor Besnier, Mickael Chen, David Hurych, Eduardo Valle, and Matthieu Cord. Halton scheduler for masked generative image transformer. *arXiv preprint arXiv:2503.17076*, 2025.

[4] Tiwei Bie, Maosong Cao, Xiang Cao, Bingsen Chen, Fuyuan Chen, Kun Chen, Lun Du, Daozhuo Feng, Haibo Feng, Mingliang Gong, et al. Llada2. 1: Speeding up text diffusion via token editing. *arXiv preprint arXiv:2602.08676*, 2026.

[5] Minwoo Byeon, Beomhee Park, Haecheon Kim, Sungjun Lee, Woonhyuk Baek, and Saehoon Kim. Coyo-700m: Image-text pair dataset. `https://github.com/kakaobrain/coyo-dataset`, 2022.

[6] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11315–11325, 2022.

[7] Yifan Chang, Jie Qin, Limeng Qiao, Xiaofeng Wang, Zheng Zhu, Lin Ma, and Xingang Wang. Scalable training for vector-quantized networks with 100% codebook utilization. *arXiv preprint arXiv:2509.10140*, 2025.

[8] Jiuhai Chen, Zhiyang Xu, Xichen Pan, Yushi Hu, Can Qin, Tom Goldstein, Lifu Huang, Tianyi Zhou, Saining Xie, Silvio Savarese, et al. Blip3-o: A family of fully open unified multimodal models-architecture, training and dataset. *arXiv preprint arXiv:2505.09568*, 2025.

[9] Junsong Chen, Jincheng Yu, Chongjian Ge, Lewei Yao, Enze Xie, Yue Wu, Zhongdao Wang, James Kwok, Ping Luo, Huchuan Lu, and Zhenguo Li. Pixart-α: Fast training of diffusion transformer for photorealistic text-to-image synthesis, 2023.

[10] Junying Chen, Zhenyang Cai, Pengcheng Chen, Shunian Chen, Ke Ji, Xidong Wang, Yunjin Yang, and Benyou Wang. Sharegpt-4o-image: Aligning multimodal models with gpt-4o-level image generation. *arXiv preprint arXiv:2506.18095*, 2025.

[11] Xiaokang Chen, Zhiyu Wu, Xingchao Liu, Zizheng Pan, Wen Liu, Zhenda Xie, Xingkai Yu, and Chong Ruan. Janus-pro: Unified multimodal understanding and generation with data and model scaling. *arXiv preprint arXiv:2501.17811*, 2025.

[12] Yufeng Cui, Honghao Chen, Haoge Deng, Xu Huang, Xinghang Li, Jirong Liu, Yang Liu, Zhuoyan Luo, Jinsheng Wang, Wenxuan Wang, et al. Emu3. 5: Native multimodal models are world learners. *arXiv preprint arXiv:2510.26583*, 2025.

[13] Chaorui Deng, Deyao Zhu, Kunchang Li, Chenhui Gou, Feng Li, Zeyu Wang, Shu Zhong, Weihao Yu, Xiaonan Nie, Ziang Song, et al. Emerging properties in unified multimodal pretraining. *arXiv preprint arXiv:2505.14683*, 2025.

[14] Fangyu Ding, Ding Ding, Sijin Chen, Kaibo Wang, Peng Xu, Zijin Feng, Haoli Bai, Kai Han, Youliang Yan, Binhang Yuan, et al. Beyond masks: Efficient, flexible diffusion language models via deletion-insertion processes. *arXiv preprint arXiv:2603.23507*, 2026.

[15] Patrick Esser, Sumith Kulal, Andreas Blattmann, Rahim Entezari, Jonas Müller, Harry Saini, Yam Levi, Dominik Lorenz, Axel Sauer, Frederic Boesel, et al. Scaling rectified flow transformers for high-resolution image synthesis. In *Forty-first international conference on machine learning*, 2024.

[16] Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, Chenhan Yu, Ye Yu, Weijia Chen, Sajad Norouzi, Shiyi Lan, Ligeng Zhu, Jin Wang, Jindong Jiang, Morteza Mardani, Mehran Maghoumi, Song Han, Ante Jukic, Nima Tajbakhsh, Jan Kautz, and Pavlo Molchanov. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *preprint*, May 2026.

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

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

[19] Marton Havasi, Brian Karrer, Itai Gat, and Ricky TQ Chen. Edit flows: Flow matching with edit operations. *arXiv preprint arXiv:2506.09018*, 2025.

[20] Minghui Hu, Chuanxia Zheng, Heliang Zheng, Tat-Jen Cham, Chaoyue Wang, Zuopeng Yang, Dacheng Tao, and Ponnuthurai N Suganthan. Unified discrete diffusion for simultaneous vision-language generation. *arXiv*, 2022.

[21] Xiwei Hu, Rui Wang, Yixiao Fang, Bin Fu, Pei Cheng, and Gang Yu Ella. Equip diffusion models with llm for enhanced semantic alignment. *arXiv preprint arXiv:2403.05135*, 5(7):16, 2024.

[22] Mario Michael Krell, Matej Kosec, Sergio P Perez, and Andrew Fitzgibbon. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *arXiv preprint arXiv:2107.02027*, 2021.

[23] Black Forest Labs. Flux. `https://github.com/black-forest-labs/flux`, 2024.

[24] José Lezama, Huiwen Chang, Lu Jiang, and Irfan Essa. Improved masked image generation with token-critic. In *European Conference on Computer Vision*, pages 70–86. Springer, 2022.

[25] Daiqing Li, Aleks Kamko, Ehsan Akhgari, Ali Sabet, Linmiao Xu, and Suhail Doshi. Playground v2.5: Three insights towards enhancing aesthetic quality in text-to-image generation, 2024.

[26] Shufan Li, Jiuxiang Gu, Kangning Liu, Zhe Lin, Aditya Grover, and Jason Kuen. Snce: Geometry-aware supervision for scalable discrete image generation. *arXiv preprint arXiv:2603.15150*, 2026.

[27] Shufan Li, Jiuxiang Gu, Kangning Liu, Zhe Lin, Zijun Wei, Aditya Grover, and Jason Kuen. Lavida-o: Elastic masked diffusion models for unified multimodal understanding and generation. *arXiv preprint arXiv:2509.19244*, 2025.

[28] Shufan Li, Jiuxiang Gu, Kangning Liu, Zhe Lin, Zijun Wei, Aditya Grover, and Jason Kuen. Sparse-lavida: Sparse multimodal discrete diffusion language models. *arXiv preprint arXiv:2512.14008*, 2025.

[29] Shufan Li, Konstantinos Kallidromitis, Hritik Bansal, Akash Gokul, Yusuke Kato, Kazuki Kozuka, Jason Kuen, Zhe Lin, Kai-Wei Chang, and Aditya Grover. Lavida: A large diffusion language model for multimodal understanding. *arXiv preprint arXiv:2505.16839*, 2025.

[30] Shufan Li, Yuchen Zhu, Jiuxiang Gu, Kangning Liu, Zhe Lin, Yongxin Chen, Molei Tao, Aditya Grover, and Jason Kuen. Lavida-r1: Advancing reasoning for unified multimodal diffusion language models. *arXiv preprint arXiv:2602.14147*, 2026.

[31] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

[32] Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. dkv-cache: The cache for diffusion language models. *arXiv preprint arXiv:2505.15781*, 2025.

[33] Yuhang Ma, Xiaoshi Wu, Keqiang Sun, and Hongsheng Li. Hpsv3: Towards wide-spectrum human preference score. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15086–15095, 2025.

[34] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: Vq-vae made simple. *arXiv preprint arXiv:2309.15505*, 2023.

[35] OpenAI. Dall·e 3. `https://openai.com/index/dall-e-3/`, 2023.

[36] OpenAI. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[37] William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 4172–4182. IEEE, 2023.

[38] Dustin Podell, Zion English, Kyle Lacey, Andreas Blattmann, Tim Dockhorn, Jonas Müller, Joe Penna, and Robin Rombach. Sdxl: Improving latent diffusion models for high-resolution image synthesis. *arXiv preprint arXiv:2307.01952*, 2023.

[39] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PmLR, 2021.

[40] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with clip latents. *arXiv preprint arXiv:2204.06125*, 1(2):3, 2022.

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

[42] 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. *International journal of computer vision*, 115(3):211–252, 2015.

[43] Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

[44] Christoph Schuhmann. Laion-aesthetics. `https://laion.ai/blog/laion-aesthetics/`, 2022. Accessed: 2024 - 03 - 06.

[45] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *Advances in neural information processing systems*, 35:25278–25294, 2022.

[46] Team Seedream, Yunpeng Chen, Yu Gao, Lixue Gong, Meng Guo, Qiushan Guo, Zhiyao Guo, Xiaoxia Hou, Weilin Huang, Yixuan Huang, et al. Seedream 4.0: Toward next-generation multimodal image generation. *arXiv preprint arXiv:2509.20427*, 2025.

[47] Fengyuan Shi, Zhuoyan Luo, Yixiao Ge, Yujiu Yang, Ying Shan, and Limin Wang. Scalable image tokenization with index backpropagation quantization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16037–16046, 2025.

[48] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024.

[49] Qingyu Shi, Jinbin Bai, Zhuoran Zhao, Wenhao Chai, Kaidong Yu, Jianzong Wu, Shuangyong Song, Yunhai Tong, Xiangtai Li, Xuelong Li, et al. Muddit: Liberating generation beyond text-to-image with a unified discrete diffusion model. *arXiv preprint arXiv:2505.23606*, 2025.

[50] Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

[51] Peize Sun, Yi Jiang, Shoufa Chen, Shilong Zhang, Bingyue Peng, Ping Luo, and Zehuan Yuan. Autoregressive model beats diffusion: Llama for scalable image generation. *arXiv preprint arXiv:2406.06525*, 2024.

[52] Meituan LongCat Team, Bin Xiao, Chao Wang, Chengjiang Li, Chi Zhang, Chong Peng, Hang Yu, Hao Yang, Haonan Yan, Haoze Sun, et al. Longcat-next: Lexicalizing modalities as discrete tokens. *arXiv preprint arXiv:2603.27538*, 2026.

[53] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[54] Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. *arXiv preprint arXiv:2503.00307*, 2025.

[55] Xinlong Wang, Xiaosong Zhang, Zhengxiong Luo, Quan Sun, Yufeng Cui, Jinsheng Wang, Fan Zhang, Yueze Wang, Zhen Li, Qiying Yu, et al. Emu3: Next-token prediction is all you need. *arXiv preprint arXiv:2409.18869*, 2024.

[56] Chenfei Wu, Jiahao Li, Jingren Zhou, Junyang Lin, Kaiyuan Gao, Kun Yan, Sheng-ming Yin, Shuai Bai, Xiao Xu, Yilei Chen, et al. Qwen-image technical report. *arXiv preprint arXiv:2508.02324*, 2025.

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

[58] Jinheng Xie, Zhenheng Yang, and Mike Zheng Shou. Show-o2: Improved native unified multimodal models. *arXiv preprint arXiv:2506.15564*, 2025.

[59] Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. Multimodal large diffusion language models. *arXiv preprint arXiv:2505.15809*, 2025.

[60] Zebin You, Shen Nie, Xiaolu Zhang, Jun Hu, Jun Zhou, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. Llada-v: Large language diffusion models with visual instruction tuning. *arXiv preprint arXiv:2505.16933*, 2025.

[61] 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. *arXiv preprint arXiv:2206.10789*, 2(3):5, 2022.

[62] Lijun Yu, José Lezama, Nitesh B Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, et al. Language model beats diffusion–tokenizer is key to visual generation. *arXiv preprint arXiv:2310.05737*, 2023.

[63] Shuibai Zhang, Fred Zhangzhi Peng, Yiheng Zhang, Jin Pan, and Grigorios G Chrysos. Corrective diffusion language models. *arXiv preprint arXiv:2512.15596*, 2025.

[64] Lei Zhu, Fangyun Wei, Yanye Lu, and Dong Chen. Scaling the codebook size of vq-gan to 100,000 with a utilization rate of 99%. *Advances in Neural Information Processing Systems*, 37:12612–12635, 2024.

A Additional Technical Details

A.1 Formulation of Masked Diffusion Models

In this section, we provide an overview of the standard formulation of discrete diffusion models that are adopted by the literature [43, 31, 60, 29, 27]. Notations are adapted from these references to be consistent with the ones used in the main paper to avoid potential confusion.

Given a sequence y^0 consisting of L discrete tokens $y_1^0 ... y_L^0$, the forward discrete diffusion process $q(y^t|y^s)$ gradually replace the original tokens in y^0 to a special mask token [M] over the time interval $[0, 1]$, with $1 \geq t \geq s \geq 0$. At $t = 1$, the sequence y^1 is a fully masked sequence. This forward process is formally defined as

$$q(y_i^t|y_i^s) = \begin{cases} \text{Cat}(y_i^t; \mathbf{M}), & \text{if } y_i^s = [M] \\ \text{Cat}(y_i^t; \frac{1-t}{1-s}\mathbf{Y}_i^s + \frac{t-s}{1-s}\mathbf{M}), & \text{if } y_i^s \neq [M], \end{cases} \tag{5}$$

where $\text{Cat}(\cdot)$ denotes a discrete categorical distribution. $\mathbf{M}, \mathbf{Y_i^s} \in \mathbb{R}^{|V|}$ are one-hot probability vectors, and $|V|$ is the vocabulary size. Specifically, $\mathbf{M}$ is the one-hot vector of the special token $[M]$, and $\mathbf{y_i^s}$ is a one-hot vector of the token y_i^s. It has been shown that this forward process has the following marginal:

$$q(y_i^t|y_i^0) = \text{Cat}(y_i^t; (1-t)\mathbf{Y}_i^0 + t\mathbf{M}). \tag{6}$$

MDLM [43] shows that the posterior of the reverse process $p(y^s|y^t, y^0)$ has the following form:

$$p(y_i^s|y_i^t, Y_i^0) = \begin{cases} \text{Cat}(y_i^s; \mathbf{y}_i^t), & \text{if } y_i^s \neq [M] \\ \text{Cat}(y_i^s; \frac{t-s}{t}\mathbf{Y}_i^0 + \frac{s}{t}\mathbf{M}), & \text{if } y_i^s = [M]. \end{cases} \tag{7}$$

At inference, the clean sequence y^0 is not known at start, so it is substituted with the prediction from a masked diffusion model $p_\theta(Y_i^0|y^t)$, which gives the following empirical sampling process:

$$p_\theta(y_i^s|y^t) = \begin{cases} \text{Cat}(y_i^s; \mathbf{y}_i^t), & \text{if } y_i^s \neq [M] \\ \text{Cat}(y_i^s; \frac{t-s}{t}p_\theta(Y_i^0|y^t) + \frac{s}{t}\mathbf{M}), & \text{if } X_s^i = [M]. \end{cases} \tag{8}$$

When sampling a sequence from p_θ, we initialize y^1 as a fully masked sequence and iterative applies equation 8 until we reach y^0. Notably, once token i is unmasked at timestep t, then $y_i^s \neq [M]$ holds for all timesteps s such that $s < t$, and we have $p_\theta(y_i^s|y^t) = \text{Cat}(y_i^s; \mathbf{y}_i^t)$. This means that it will not change in subsequent sampling steps.

During training, we optimize the maximum likelihood objective

$$\mathcal{L}_{\text{MDM}} = -\mathbb{E}_{(y,x)\sim\mathcal{D}}[\log p_\theta(y|x)] \tag{9}$$

However, $\log p_\theta(y|x)$ is intractable and requires integrating over all possible trajectories, we instead optimize the ELBO described in equation 1 from the main paper:

$$\mathcal{L}_{\text{ELBO}} = \mathbb{E}_{y^0,\, t\sim\text{Unif}([0,1]),\, y^t\sim q(y^t|y^0)} \left[-\frac{1}{t} \sum_{i=1}^{L} \mathbf{I}\{y_i^t = [\text{M}]\} \log p_\theta(y_i^0 \mid y^t) \right] \tag{10}$$

We can safely introduce the indicator term $\mathbf{I}\{y_i^t = [\text{M}]\}$ because when $y_i^t = [\text{M}]$, $p_\theta(y_i^s|y^t) = \text{Cat}(y_i^s; \mathbf{y}_i^t)$ does not depend on p_θ. However, these assumptions no longer holds when token editing is introduced.

A.2 Token Editing

In this section, we provide a detailed description of the token editing process.

Training. To enable token editing, we modify the forward process $q(y_i^t|y_i^0)$ to $q'(y_i^t|y_i^0)$ by introducing a corruption term:

$$q'(y_i^t|y_i^0) = \text{Cat}(y_i^t; (1-t)(1-\alpha)\mathbf{Y}_i^0 + (1-t)\alpha\mathbf{C}(y_i^0) + t\mathbf{M}). \tag{11}$$

where α denotes the corruption level and $\mathbf{C} : V \to \mathbb{R}^{|V|}$ defines the distribution of corrupted tokens conditioned on the original clean token y_i^0.

In our implementation, we design $\mathbf{C}$ such that its probability mass is distributed over tokens from the same image as well as neighboring tokens in the embedding space. To obtain neighboring tokens, we directly reuse the precomputed K-means clustering used for the GCE objective. As shown in Figure 2 of the main paper, tokens from the same cluster typically produce visually similar images with only minor degradation in quality, making them a good approximation of erroneous model predictions during inference.

Using the augmented forward process $q'(y_i^t|y_i^0)$, we optimize the edit-aware ELBO objective defined in Equation 2 of the main paper.

Inference. We largely adopt the inference pipeline from LLaDa-2.1 [4], which explores token editing for text diffusion models. Specifically, at each inference step, we first perform the standard unmasking operation defined in Equation 8. In addition, we compute $p_\theta(y_i^0|y^t)$ at non-masked positions. We edit token y_i^t whenever the model predicts an alternative token $\widehat{y}_i^t$ with confidence above a predefined threshold τ.

In our experiments, we find that $\tau = 0.6$ and $\alpha = 0.1$ work best. Detailed ablation results are provided in Appendix B.

A.3 Grouped Cross-Entropy

In this section, we provide a detailed description of GCE. Recall from Section 3.3 of the main paper that each clustering term is defined as

$$\log \mathbb{P}(C(y_i^0)|y^t) = \log \left(\sum_{j \in C(y_i^0)} p_j \right). \tag{12}$$

We can further express this term using the unnormalized logits h_j:

$$\log \left(\sum_{j \in C(y_i^0)} p_j \right) = \log \left(\sum_{j \in C(y_i^0)} \frac{\exp(h_j)}{\sum_{k=1}^{|V|} \exp(h_k)} \right) \tag{13}$$

$$= \log \left(\frac{\sum_{j \in C(y_i^0)} \exp(h_j)}{\sum_{k=1}^{|V|} \exp(h_k)} \right) \tag{14}$$

$$= \log \left(\sum_{j \in C(y_i^0)} \exp(h_j) \right) - \log \left(\sum_{k=1}^{|V|} \exp(h_k) \right). \tag{15}$$

Given a batch of logits represented as a tensor of shape $N \times |V|$, where $N = \text{SeqLen} \times \text{NumSeqs}$, the second term can be efficiently implemented using the optimized `torch.logsumexp` operator over the vocabulary dimension. The first term, however, is more challenging because clusters have different sizes.

A naive implementation would mask logits by setting codes outside the cluster $C(y_i^0)$ to $-\infty$. However, this approach requires allocating a full copy of the logits tensor, leading to substantial

memory and latency overhead. To address this issue, we note that cluster sizes are bounded. Instead of allocating a tensor of size $N \times |V|$, we only allocate a tensor of size $N \times |C_{\max}|$, where $|C_{\max}|$ denotes the size of the largest cluster. Empirically, this value is 391 when using 8,192 clusters and 192 when using 16,384 clusters. In both cases, this requires less than 1% of the memory needed by the naive implementation.

During the backward pass, instead of relying on autograd, we manually compute gradients using

$$\frac{\partial}{\partial h_i} \log P(C) = p_i \left(\frac{\mathbf{I}\{i \in C\}}{P(C)} - 1 \right) = p_i \frac{\mathbf{I}\{i \in C\}}{P(C)} - p_i. \tag{16}$$

Although p_i is dense, the first term is non-zero only for tokens within the target cluster. Therefore, we can apply a similar optimization strategy and allocate only $N \times |C_{\max}|$ memory for the sparse term, then combine the two terms efficiently using in-place `torch.scatter_add` operations. We provide the full PyTorch implementation below:

```
class OptimalCappedGroupedCE(torch.autograd.Function):
    @staticmethod
    def forward(ctx, logits, target_clusters, cluster_map, cluster_sizes, cap
    ):
        logits_f32 = logits.float()

        # log_z = log sum(exp(h_i)) for all i in V
        log_z = torch.logsumexp(logits_f32, dim=1, keepdim=True)

        # Map targets to their cluster members
        relevant_indices = cluster_map[target_clusters]
        relevant_logits = torch.gather(logits_f32, 1, relevant_indices)

        # Apply capacity mask to the cluster
        mask = torch.arange(cap, device=logits.device).unsqueeze(0) < \
               cluster_sizes[target_clusters].unsqueeze(1)
        masked_relevant = torch.where(mask, relevant_logits, torch.tensor(-1
    e20))

        # log_num = log sum(exp(h_k)) for k in C_y
        log_num = torch.logsumexp(masked_relevant, dim=1, keepdim=True)

        loss = (log_z - log_num).mean()

        ctx.save_for_backward(logits_f32, log_z, log_num, relevant_indices,
    mask)
        ctx.original_dtype = logits.dtype
        return loss

    @staticmethod
    def backward(ctx, grad_output):
        logits_f32, log_z, log_num, relevant_indices, mask = ctx.
    saved_tensors

        # Gradient Step 1: Compute p_i for all classes (Dense)
        grad_logits = torch.exp(logits_f32 - log_z)

        # Gradient Step 2: Compute pi / P(C_y) for cluster classes (Sparse)
        relevant_logits = torch.gather(logits_f32, 1, relevant_indices)
        target_grads = torch.exp(relevant_logits - log_num) * mask.float()

        # Gradient Step 3: In-place subtraction (Memory Efficient)
        # grad = p_i - pi / P(C_y)
        grad_logits.scatter_add_(1, relevant_indices, -target_grads)

        # Scale by external gradient and batch size
        grad_logits.mul_(grad_output / logits_f32.shape[0])

        return grad_logits.to(ctx.original_dtype), None, None, None, None
```

Table 5: **Training configurations across two stages.**

	Stage 1	**Stage 2**
Learning Rate	1×10^{-4}	1×10^{-5}
Steps	200k	100k
β_1	0.99	0.99
β_2	0.999	0.999
optimizer	AdamW	AdamW
Learning Rate Schedule	Cosine	Cosine
Final Learning Rate	1×10^{-5}	1×10^{-6}
Model Size	8B	8B
Image Resolution	256	$512 \rightarrow 1024$
Global Batch Size	1,024	256
Token Editing	Disabled	Enabled

Note J_{GCE} is a maximization objective. To be compatible with semantic of vanilla Pytorch cross-entropy loss which is a minimization objective, the loss term and gradients in the code are negated.

B Additional Experiment Details and Results

In this section, we provide additional training details and experimental results, including qualitative samples and ablation studies.

B.1 Training Data

Our training dataset consists of 137M text-image pairs sourced from public datasets. The data pipeline largely follows the prior work LaViDa-O [27]. Specifically, we source raw images from LAION-2B [45], COYO-700M [5], BLIP3o-60k [8], and ShareGPT4o-Image [10]. These datasets are heavily filtered to remove NSFW prompts, low CLIP-score samples [39], low aesthetic-score samples [44], and low-resolution images.

For all images from LAION-2B and COYO-700M, we use Qwen3-VL [2] to re-caption the images instead of relying on the original alt-text annotations, which are often noisy. However, we retain raw captions with high CLIP scores and randomly choose between VLM-generated captions and raw captions for these samples. We make this choice primarily to support keyword-based prompting such as "high quality" and "4k" during inference, since such keywords do not naturally emerge in VLM-generated captions.

B.2 Training Setup and Hyperparameters

We adopt the Emu-3.5 tokenizer [12], which has a vocabulary size of 131,072. We initialize Nemotron-Labs-Diffusion-Image from a pretrained diffusion language model [16]. Training consists of two stages. In the first stage, we pretrain the model on 256×256 images for 200k steps with a global batch size of 1024. In the second stage, we scale to 512×512 resolution for 20k steps and then to 1024×1024 resolution for 80k additional steps with a global batch size of 256. Training is conducted on 64 H100 GPUs. Additional details are provided in Table 5.

B.3 Ablation Studies of Editing Thresholds

We study the effect of varying the editing threshold τ under different numbers of inference steps. We visualize the resulting HPSv3 scores in Figure 7. Overall, enabling token editing consistently outperforms the no-editing baseline. Among all evaluated settings, $\tau = 0.6$ achieves the best image quality for most NFEs.

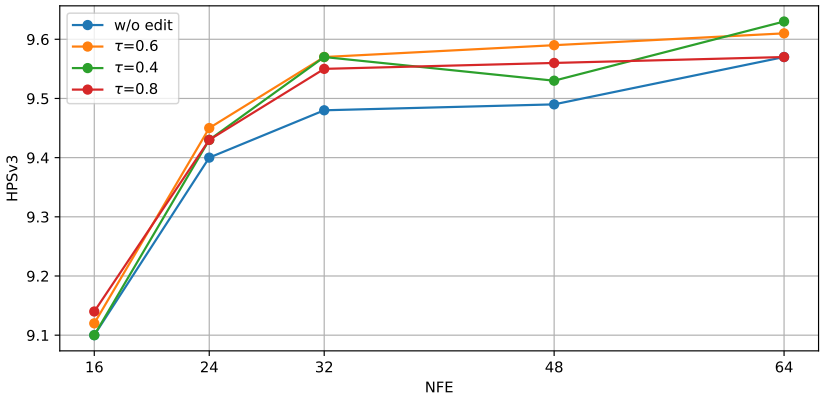


Figure 7: **Effect of different token-editing thresholds on HPSv3 scores.**

Table 6: **Ablation Experiments on Cluster size.** *Model have identical-sized transformer layer. Parameter count increased due to larger token embedding and final linear head.

Cluster Sizes	Params	Tokenizer	Tokenizer Pretraining	Codebook	Epoch	FID ↓
8,192	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	100	3.67
16,384	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	100	3.44
Both	577M*	Emu3.5-IBQ [12]	Large-Scale T2I	131,072	100	3.40

B.4 Ablation Studies of Cluster Sizes

We investigate the effect of varying the number of clusters used in the GCE objective and report FID scores on ImageNet-256 in Table 6. We find that combining both 16,384-cluster and 8,192-cluster supervision performs better than using either clustering level alone. When using only a single clustering level, the 16,384-cluster setting performs better, presumably because the 8,192-cluster setting is coarser and provides less refined supervision signals.

B.5 Ablation Studies of Corruption Type and Scale

We explore different corruption strategies for the token-editing objective by evaluating HPSv3 scores after 10K steps 1024 resolution training in Stage-2. Specifically, we evaluate random noise corruption, corruption using neighboring tokens in embedding space, and resampling tokens from the same input image. We also explored self-conditiong, which uses model's own prediction error as corruption source. Results are reported in Table 8. Using a combination of neighboring tokens and resampled tokens achieves the best performance.

Additionally, we experiment with different corruption ratios α and observe no significant differences in image quality when α lies within a reasonable range. However, image quality degrades when α becomes too large. This is because high corruption levels (e.g., $\alpha = 0.5$) make it substantially more difficult to distinguish clean tokens and corrupted tokens due to the lower signal-to-noise ratio, increasing optimization difficulty. In our final experiments, we use $\alpha = 0.1$.

Table 7: Noise Type Comparison

Noise Type	HPSv3 (10k step)
No corruption	7.89
Random	8.53
Adjacent Tokens	8.81
Adj. Tokens + Resamp.	8.99
Self-Conditioning	8.92

Table 8: Threshold α Impact

α Value	HPSv3 (10k step)
0	7.89
0.1	8.99
0.3	8.97
0.5	7.52

Model	FID
DiT-XL [37]	2.27
MaskGIT [6]	6.56
MaskGIT + Token-Critic [24]	4.69
MaskGIT + Remasking [54]	4.45
Halton-MaskGIT [3]	3.74
MaskGIT-GCE	2.71
MaskGIT-GCE + Remasking	2.64
MaskGIT-GCE + TokenEdit Loss	2.68
MaskGIT-GCE + TokenEdit Loss + TokenEdit Inference	**2.24**

Table 9: Additional Results on ImageNet.

B.6 Additional Small-Scale Experiments

We conduct additional experiments on ImageNet using MaskGIT as the base model to further study the effectiveness of token editing and GCE loss. We also comapred with Token-Critic [24] and Remasking [54] baselines. Results are shown in Table 9. The proposed GCE and token editing effectively improves image fidelity, resulting lower FID scores.

B.7 Additional Qualitative Results

In this section, we provide additional qualitative samples to further demonstrate the effectiveness of Nemotron-Labs-Diffusion-Image. Figure 8 presents additional text-to-image generation results. Figures 9 and 10 compare generations produced with and without token editing under the same random seed. We observe that token editing consistently improves image fidelity by refining details and correcting artifacts.

Few-Step Generation. In most experiments, we use 64 diffusion steps to generate image samples. We also explored few-step generation and show qualitative results in Figure 11. Compared with continuous flow-matching models which predicts a blurry mean field at few-step setting, Nemotron-Labs-Diffusion-Image can generate images with reasonable quality at only 4 steps. This capability naturally emerges without any distillation process, highlighting another advantage of discrete diffusion models.

Denoising Dynamics. To further understand why Nemotron-Labs-Diffusion-Image works well with few sampling steps out of box, we visualize the denoising dynamics at different stages of the 64-step sampling schedule in Figure 12. We observe that the clean image prediction y^0 for Nemotron-Labs-Diffusion-Image converges faster to reasonable quality, while the clean image prediction for the continuous baseline Qwen-Image [56] remain noisy and blurry. This illustration highlight the effectiveness of the discrete diffusion process in image synthesis.

C Compute Resources

We train the model on 64 H100 GPUs across 8 nodes. Training takes approximately 16 days in total.

D Limitations

Despite the effectiveness of Nemotron-Labs-Diffusion-Image, several limitations remain. First, although we demonstrate that the token-editing mechanism improves image quality, it does not eliminate all artifacts, and the model may still generate erroneous outputs. Second, although we achieve substantial performance gains by optimizing the GCE objective with a carefully designed fused operator, additional improvements may still be possible through customized low-level CUDA kernels. We leave this direction for future work.



a beautiful sunset, bright and colourful, ultra realistic, UHD, 8k

fluffy white bunny as a Jedi, blue lightsaber, fierce expression

Panna Cotta with Mixed Berry Compote in a country kitchen, warm light

underground fantasy city with magic

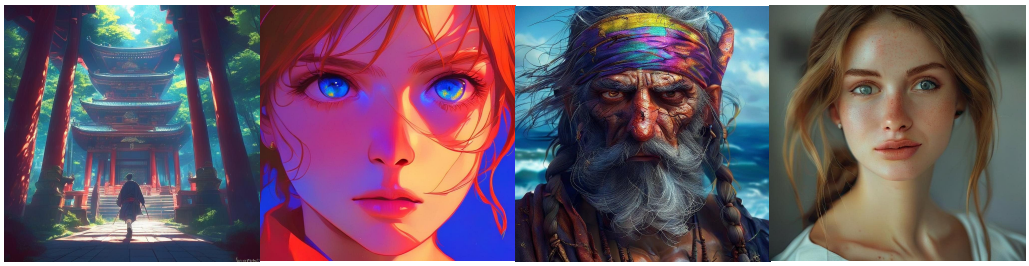


landscape Minecraft artstyle, photo contest, high contrast, bright and clear photo, sharp, detailed, delicate

Medieval fantasy landscape magical wizard sunset vertigo flux of light golden light fantasy landscape beautiful calm illustration sunset

landscape of aurora borealis sky and snowy ground, bright colors, realistic, wide shot from far away, breathtaking composition

DD art, Tall thin, male human sorcerer, handsome face, gold and green magic sparkling around him photorealistic

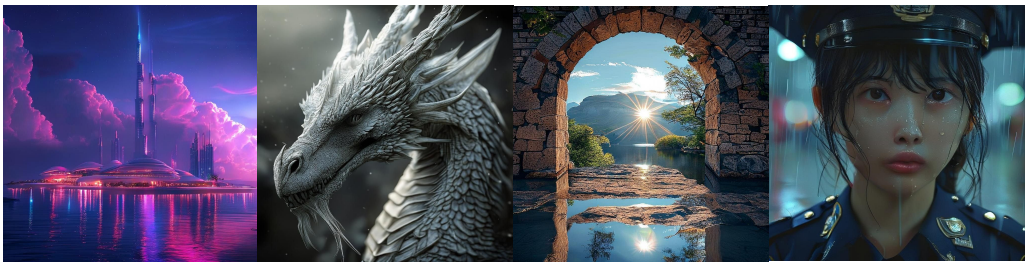


hecate walking in a temple forest, anime, magical realism, hd, photorealistic, anime

closeup portrait, in the style of hand painted anime by ilya kuvshinov, vibrant colors, attention to detail, highlydetailed bright eyes, high resolution face,

pirate character portrait gray hair weathered face thick beard weathered face colorful headband patched clothing and ocean for the background

dental assistant, female, in her 20s, 4k, realistic young female in her 20s, dental assistant, realistic photograph



futuristic island starport on the ocean near dubai with a tall spire that extends into the clouds, night, reflections on water, dynamic lighting, neon, starwars

dragon from Skyrim ,full body, cinematic, photorealism, intricate ultra detail, high sharpness, 8K cinematic, photography, realistic,hyper realistic,3d,hdr

view on the nature through a stone arch, with reflections of the sun, photography, 8k, hyper realist, ultra detailled

ultra realistic, hyperealistic, photography, japanese girls police officers, realistic style, rain, 4k

Figure 8: Additional qualitative text-to-image generation results.



Figure 9: Qualitative comparison between generations with and without token editing.

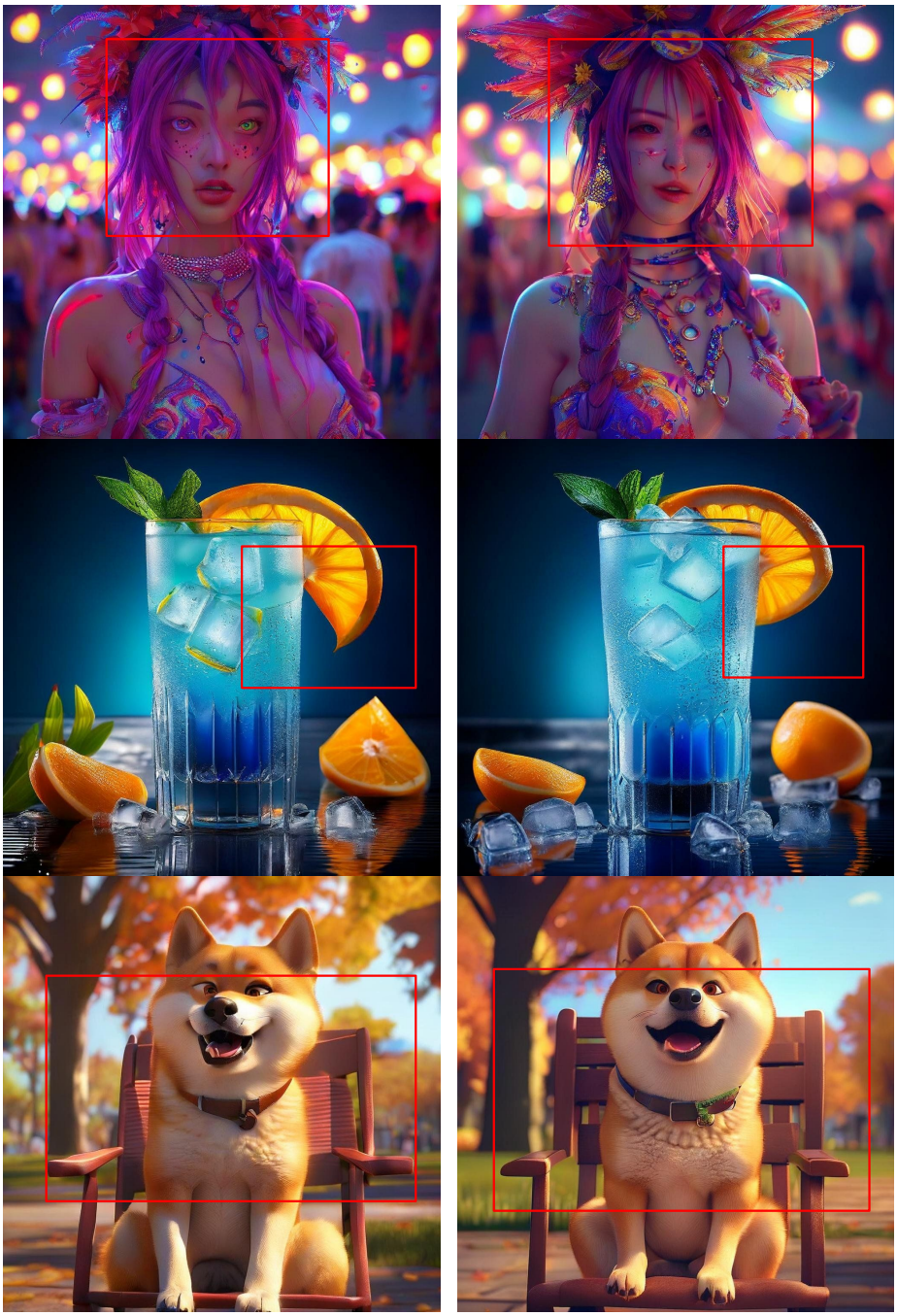


Figure 10: Additional qualitative comparisons for token editing.



Figure 11: **Few-Step Generation Results.** We visualize the sampled images with 1,2,3,4,5 total steps and compare with a continuous model Qwen-Image [56].



Figure 12: **Denoising Dynamics throughout the sampling process.** We visualize the y^0 prediction at 1st,4th,8th,32th, and 64th step out of a total of 64 steps.

E Broader Impact

Nemotron-Labs-Diffusion-Image has strong text-to-image generation capabilities, which may be misused to generate harmful or offensive content. We strongly caution against such use cases. Additionally, our model may inherit biases present in the base language model as well as biases contained in the training data. Our model is intended primarily for research purposes to facilitate future exploration of foundational discrete image generators. We do not recommend its use for other purposes.

F Licenses

We make use of the following assets:

Models: Emu-3.5-Tokenizer [12] (Apache-2.0), Qwen3-VL [2] (Apache-2.0), Nemotron-Labs-Diffusion [16] (Nvidia Open Model License)

Datasets: LAION [45] (MIT), COYO [5] (CC-BY-4.0), MJHQ [25] (CC-BY-4.0), BLIP3o-60k [8] (Apache-2.0), and ShareGPT4o-Image [10] (CC-BY-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: We provide extensive experiments to show the effectiveness of Nemotron-Labs-Diffusion-Image for masked discrete image generation.

   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: Section D

   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: No theoretical contribution

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: Appendix B

   Guidelines:

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

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

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

Answer: [No]

Justification: Pending internal approval at the time of submission.

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: Appendix B

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [No]

   Justification: 1) Training models with different seeds at pretraining scale is computationally expensive. 2)Baseline numbers cited from literature do not contain confidence interval, so its hard to perform statistic test. Nevertheless, we argue that the scale of experiments is sufficiently large scale to mitigate uncertain. For example, MJHQ benchmark consists of 30k prompts.

   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: C

   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: We followed guidelines

   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: See Appendix E

    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: We filtered dataset and remove NSFW prompts, see Appendix B

    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: See Appendix F

    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: No 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: No crowdsourcing

    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: No 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: We did not use LLM in unconventional, non-standard way

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.