Flow to the Mode: Mode-Seeking Diffusion Autoencoders for State-of-the-Art Image Tokenization

Supplementary Material

A. Training setup and hyperparameters

A.1. Stage 1A - Tokenizer pre-training

All hyperparameters are given in Table 1. We found that for FlowMo, using a relatively low commitment loss weight relative to GAN-based tokenizers was beneficial. In fact, bounding the commitment loss by applying a $\tanh$ activation prior to quantization degraded performance. We use token factorization following [11, 17], which enables a large vocabulary size without excessive memory usage in the calculation of the entropy loss.

Due to limited computational resources, we parameterized all FlowMo models in μP and determined the optimal losses, noise schedules, and other hyperparameters in the exploratory "FlowMo (fewer params)" configuration, which was trained on A100, H100, H200, L40S, or A6000 GPUs as available. We then transfer these hyperparameters to the larger and more FLOP-intensive FlowMo-Lo and FlowMo-Hi configurations for final experiments. We train these final configurations for relatively few epochs (130 and 80 respectively) on a single $8\times$H100 node, and it is likely that further training would continue to improve results.

A.2. Stage 1B - Tokenizer post-training

MaskBiT [15] noted the effectiveness of a ResNet-style perceptual loss for tokenizer training, and that this loss may have been used in prior work whose tokenizer training recipe was unknown [4]. However, we were unable to use this network effectively in our loss $\mathcal{L}_{\text{percep}}$, instead obtaining better results in Stage 1A with the LPIPS-VGG network. Interestingly, [16] also applied a perceptual loss based on LPIPS-VGG on the 1-step denoised prediction of the network.

To verify the necessity of the post-training algorithm we propose for FlowMo, we consider an alternative post-training stage in which the ResNet perceptual loss is used but on the 1-step denoised prediction as in $\mathcal{L}_{\text{perc}}$. We sweep a range of loss weights for this ResNet-based perceptual loss in Table 2, using the FlowTok-small configuration. Importantly, none are able to match the performance of $\mathcal{L}_{\text{sample}}$ post-training.

During post-training, due to the computational expense, we reduce the batch size to 64 and correspondingly reduce the learning rate to 5×10^{-5}. In addition to gradient checkpointing, we also use gradient accumulation in this stage. We train Stage 1B for approximately 1 epoch for both FlowMo-Lo and FlowMo-Hi, and apply early stopping to counteract eventual reward hacking [1, 12].

We sample the timesteps to integrate the ODE for $\mathcal{L}_{\text{sample}}$ randomly by taking $u_1, ..., u_n \sim \text{Unif}(0, 1)$ and then setting

$$t_i = (\sum_{j=i}^{n} u_j) / \sum_{j=1}^{n} u_j)$$

A.3. Stage 2 - Generative model training

To verify the quality of an image tokenizer, it is necessary to ensure it can be used to train a high-quality generative model. As in tokenizer training, we lack resources for matching the computational requirements of state-of-the-art generative models which are trained for e.g., 1000-1080 epochs with batch size 1024-2048 [15, 18]. We use a MaskGiT configuration with hidden size 1024 and 28 layers, with 397M total parameters. training for 300 epochs with learning rate 10^{-4} and batch size 128. Both [11] and FlowMo produce 256 tokens of size 18 bits, which we convert to 512 tokens of size 9 bits for training MaskGiT. We use guidance 1.5 for both models, and tune the sampling temperature separately for each model to ensure fair comparison, since we find that models trained atop OpenMagViT-V2 prefer lower temperature sampling, while those trained atop FlowMo prefer high temperature. We use 64 sampling steps for both models with softmax temperature annealing and no guidance decay.

The purpose of our second stage training is to ensure FlowMo can be used to train a second stage generative model of comparable quality to one trained with a traditional tokenizer *given the same computational resources*. Our goal is not to improve the state of the art for ImageNet-1K generation at 256 resolution.

B. Evaluation setup and hyperparameters

For the FlowMo tokenizer, we use a guidance interval [9] of $(0.17, 1.02)$ in the EDM2 [7] convention (translating to (0.145, 0.505) in terms of rectified flow noise levels), with classifier-free guidance weight 1.5. For the second stage, we also apply guidance, as is common practice, by shifting the token logits according to the logit difference with and without conditioning.

We did not observe negative interaction between use of guidance in both the tokenizer and generative model; guidance in both stages is helpful.

For all evaluation metrics (rFID, PSNR, etc.) reported in the main paper, we use 50,000 samples unless otherwise

Hyperparameter	**FlowMo (fewer params)**	**FlowMo-Lo**	**FlowMo-Hi**
Learning rate	0.0001	-	-
Batch size	128	-	-
Weight decay	0	-	-
Num. epochs	40	130	80
λ_{ent}	0.0025	-	-
λ_{commit}	0.000625	-	-
λ_{lpips}	0.1	-	-
Hidden size (μP width)	768	1152	1152
MLP ratio	4	-	-
Encoder patch size	8	4	4
Decoder patch size	8	4	4
Encoder depth	8	-	-
Decoder depth	16	-	-
Latent sequence length	256	-	-
Latent token size	18	18	56
Codebook size for entropy loss	9	9	14
Total number of parameters ($\times 10^6$)	517	945	945

Table 1. Training hyperparameters for tokenizer training. '-' means hyperparameters are identical between FlowMo Small, A, and B configurations

Loss type	**Loss weight**	**rFID**
$\mathcal{L}_{\text{sample}}$	0.01	1.28
$\mathcal{L}_{\text{perc}}$	0.05	2.57
$\mathcal{L}_{\text{perc}}$	0.01	2.38
$\mathcal{L}_{\text{perc}}$	0.005	2.16
$\mathcal{L}_{\text{perc}}$	0.001	1.67
$\mathcal{L}_{\text{perc}}$	0.0005	1.61
$\mathcal{L}_{\text{perc}}$	0.0001	1.81
$\mathcal{L}_{\text{perc}}$	0.00025	1.64
$\mathcal{L}_{\text{perc}}$	0.0005	1.61
$\mathcal{L}_{\text{perc}}$	0.00075	1.60

Table 2. $\mathcal{L}_{\text{sample}}$ ablation. We show the result of training with $\mathcal{L}_{\text{sample}}$ in the first row. We then compare with $\mathcal{L}_{\text{perc}}$ at various weights. We conduct two sweeps, first to find the optimal loss weight order of magnitude (middle rows) and then to fine-tune the loss weight (bottom rows). Regardless, it cannot match the performance of $\mathcal{L}_{\text{sample}}$ Backpropagation through a complete sampling chain is important.

noted. We compute the rFID and other reconstruction metrics in Stage 1 against the entire 50,000-example validation set. Generative metrics such as gFID are computed against the full ImageNet-1K training dataset statistics, as is standard practice. The generation results are evaluated using the ADM codebase [5].

C. Additional Experiments

In this section, we provide some additional analyses and experiments to better understand and illustrate the capabilities of FlowMo.

C.1. Rate-distortion-perception tradeoff

It could be argued that the desired "mode-seeking" behavior could be accomplished by simply increasing the sample likelihood. Unfortunately, evaluating the sample likelihood requires solving the log-determinant of the jacobian of the flow field [10], making it impractical to optimize directly. Neglecting the impact of this term we may increase or decrease the likelihood of x_1 by modulating its norm, since it is a centered isotropic gaussian (somewhat analogous to truncation sampling in GANs [3]).

We attempt this analysis below. Whereas our proposed shifted sampler and post-training strategy improve both PSNR and rFID simultaneously, the naive sampling strategy described above only allows us to trade off rFID and PSNR against each other, constrained by the "rate-distortion-perception tradeoff" [2]. Importantly, rFID and PSNR cannot both be arbitrarily increased. See Figure 1 for a quantitative study of this process. Where the x_1 likelihood is increased too high, the result is an eventual *degradation* in the sample's perceptual quality, while PSNR continues to increase. Even if it were possible to maximize the sample likelihood, as noted in [14], directly maximizing sample likelihood is inappropriate for image realism.

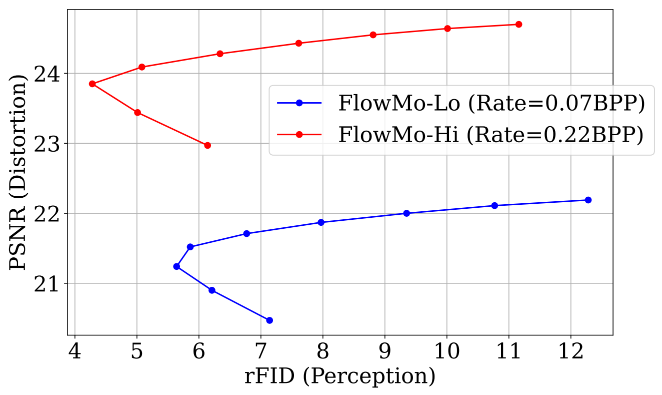


Figure 1. Analysis of FlowMo with respect to the rate-distortion-perception tradeoff [2]. Varying x_1 likelihood sweeps out different rFID/PSNR curves at different rates. For this study, we use 5,000 samples for all metrics to make it computationally feasible.

C.2. Dataset and resolution generalization.

FlowMo-Lo is a diffusion autoencoder, so it naturally generalizes to higher resolutions by employing a patch-wise diffuse-and-blend strategy known in prior work [6]. FlowMo-Lo achieves strong performance on unseen data, at 256 and 512 resolution, in terms of rFID, despite a lower BPP and only training on ImageNet-1K at 256×256. In Table 3, we compare with the Cosmos DI-8x8 tokenizer, a generalizable tokenizer trained at multiple resolutions on a large internet-scale dataset.

	ImageNet-1K		OpenImages		Food-101	
Resolution	256	512	256	512	256	512
Cosmos DI-8x8	0.87	0.39	0.85	0.52	**1.05**	0.94
FlowMo-Hi	**0.55**	**0.30**	**0.69**	**0.40**	1.39	**0.68**

Table 3. **Multiple resolutions.** FlowMo can perform encoding and decoding at arbitrary resolutions following a patchwise diffusion strategy known in prior work [6].

C.3. Tokenizer scaling.

FlowMo is a transformer-only architecture parameterized in μP to facilitate easy scaling. We show in the table below that, holding all else constant, *all metrics improve every time the width factor is increased*. For this experiment we train for 200K steps only. Other config settings are the same as FlowMo (fewer params).

μP width	# Params ($\times 10^6$)	rFID ↓	PSNR ↑	LPIPS ↓
2	260	7.77	20.84	0.169
3	367	5.31	21.28	0.160
4	517	4.45	21.60	0.155
5	710	**3.84**	**21.70**	**0.152**

C.4. "Mode-seeking," precision and recall

We illustrate the 'mode-seeking' effect of post-training below. We compare the statistics of decompressed images to reference dataset statistics via the Precision and Recall metrics [8]. When there is a 1:1 correspondence between the elements of the conditioning dataset and the reference dataset, post-training improves both precision and recall. This corresponds to reducing the diversity of the distribution $p(x|c)$ for each c by concentrating around its modes, which improves fidelity to the original reference dataset.

By contrast, when comparing to the train set statistics (i.e. an unrelated set of images not in 1:1 correspondence with the conditioning dataset), post-training improves precision (quality) at the cost of recall (diversity).

	vs. train stats.		vs. val stats.	
Model name	Prec.↑	Rec.↑	Prec.↑	Rec.↑
FlowMo (fewer params) (no posttrain)	0.734	**0.660**	0.974	0.988
FlowMo (fewer params)	**0.766**	0.634	**0.993**	**0.991**

C.5. More results with post-training

The post-training scheme proposed generalizes to multiple architectures, consistently improving their performance. We have added more results extending Table 5 below, for FlowMo (fewer params), which is defined in Table 1, and FlowMo-Continuous, which is the version against which we compare DiTo in Table 3.

Model name	rFID ↓	PSNR ↑	LPIPS ↓
FlowMo (fewer params) (no post-train)	2.02	21.32	0.143
FlowMo (fewer params)	**1.21**	**22.20**	**0.120**
FlowMo-Continuous (no post-train)	0.74	25.65	0.064
FlowMo-Continuous	**0.65**	**26.61**	**0.054**

C.6. Wall clock times

We provide wall-clock times below. Note that FlowMo encoding is comparable to LlamaGen-32, so it does not bottleneck second-stage generative modeling. The main slowdown is due to a diffusion decoder, which we consider acceptable because it enables significantly better reconstruction. Decoding could be made faster ($\approx 5\times$ to $10\times$) via distillation.

Model name	Encode 1 image (s)	Decode 1 image (s)
LlamaGen-32	0.021	**0.038**
FlowMo-B	0.050	2.391
FlowMo-B (small)	**0.011**	0.322

D. Extended visualization

We provide extended comparisons between our method and various state-of-the-art tokenizers in Figures 2, 3, 4, 5 below. Images are not cherry-picked.

References

[1] Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024. 1

[2] Yochai Blau and Tomer Michaeli. Rethinking Lossy Compression: The Rate-Distortion-Perception Tradeoff. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019. 2, 3

[3] Andrew Brock, Jeff Donahue, and Karen Simonyan. Large Scale GAN Training for High Fidelity Natural Image Synthesis. In *International Conference on Learning Representations (ICLR)*, 2019. 2

[4] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked Generative Image Transformer. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[5] Prafulla Dhariwal and Alexander Nichol. Diffusion Models Beat GANs on Image Synthesis. In *Advances in Neural Information Processing Systems*, 2021. 2

[6] Emiel Hoogeboom, Eirikur Agustsson, Fabian Mentzer, Luca Versari, George Toderici, and Lucas Theis. High-Fidelity Image Compression with Score-based Generative Models. *arXiv preprint arXiv:2305.18231*, 2024. 3

[7] Tero Karras, Miika Aittala, Jaakko Lehtinen, Janne Hellsten, Timo Aila, and Samuli Laine. Analyzing and Improving the Training Dynamics of Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. 1

[8] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved Precision and Recall Metric for Assessing Generative Models. In *Advances in Neural Information Processing Systems*, 2019. 3

[9] Tuomas Kynkäänniemi, Miika Aittala, Tero Karras, Samuli Laine, Timo Aila, and Jaakko Lehtinen. Applying Guidance in a Limited Interval Improves Sample and Distribution Quality in Diffusion Models. *arXiv preprint arXiv:2404.07724*, 2024. 1

[10] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow Matching for Generative Modeling. In *International Conference on Learning Representations (ICLR)*, 2023. 2

[11] Zhuoyan Luo, Fengyuan Shi, Yixiao Ge, Yujiu Yang, Limin Wang, and Ying Shan. Open-MAGVIT2: An Open-Source Project Toward Democratizing Auto-regressive Visual Generation. *arXiv preprint arXiv:2409.04410*, 2025. 1, 6, 7

[12] Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning Text-to-Image Diffusion Models with Reward Backpropagation. *arXiv preprint arXiv:2310.03739*, 2023. 1

[13] 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. 8, 9

[14] Lucas Theis. What makes an image realistic? *arXiv preprint arXiv:2403.04493*, 2024. 2

[15] Mark Weber, Lijun Yu, Qihang Yu, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. MaskBit: Embedding-free Image Generation via Bit Tokens. *arXiv prepprint arXiv:2409.16211*, 2024. 1

[16] Ruihan Yang and Stephan Mandt. Lossy Image Compression with Conditional Diffusion Models. In *Advances in Neural Information Processing Systems (NeurIPS)*. Curran Associates, Inc., 2023. 1

[17] Lijun Yu, José Lezama, Nitesh B. Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Vighnesh Birodkar, Agrim Gupta, Xiuye Gu, Alexander G. Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A. Ross, and Lu Jiang. Language Model Beats Diffusion – Tokenizer is Key to Visual Generation. In *International Conference on Learning Representations (ICLR)*, 2024. 1

[18] Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An Image is Worth 32 Tokens for Reconstruction and Generation. *Neural Information Processing Systems (NeurIPS)*, 2024. 1

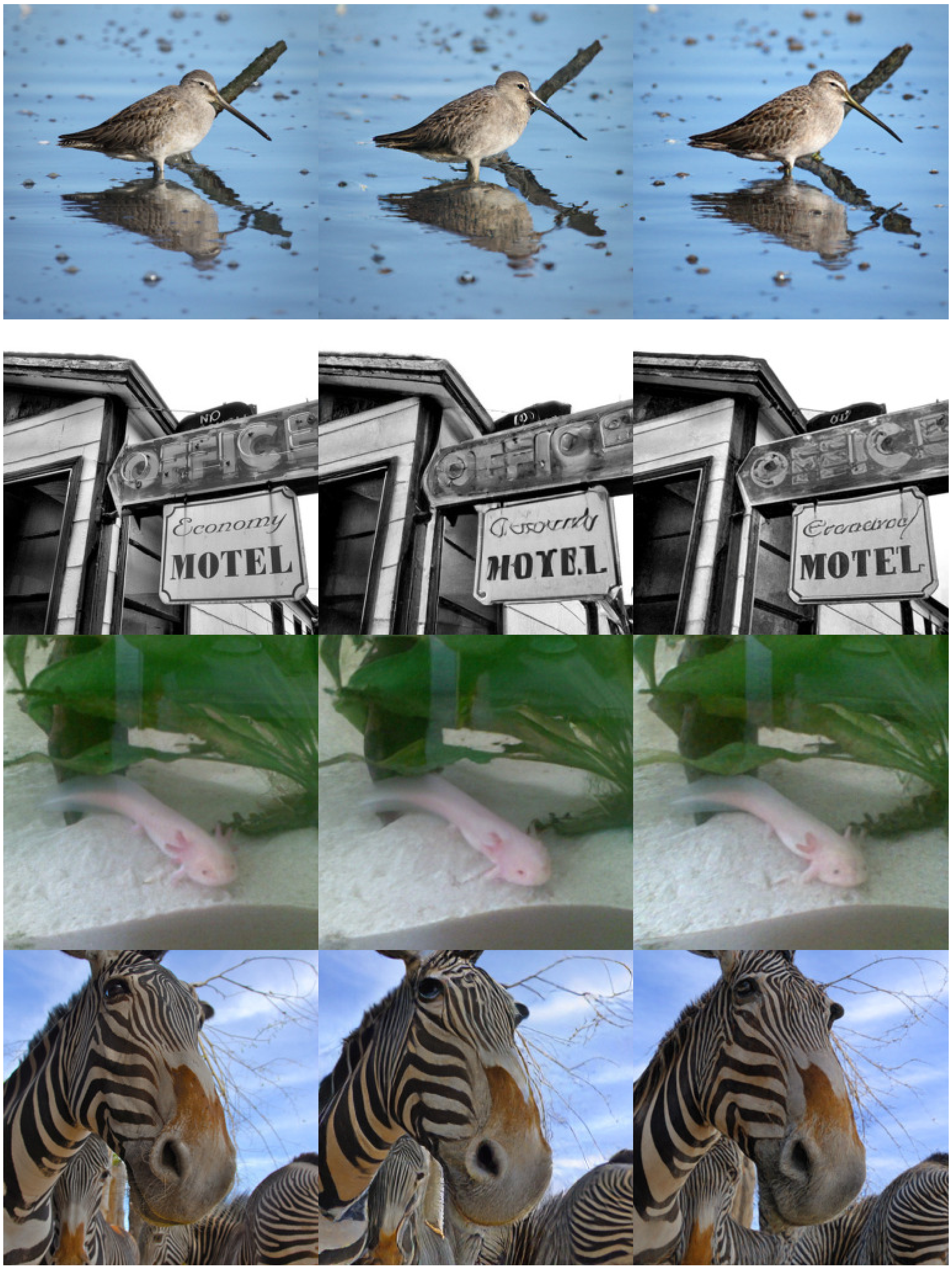


Figure 2. Tokenizer comparison.

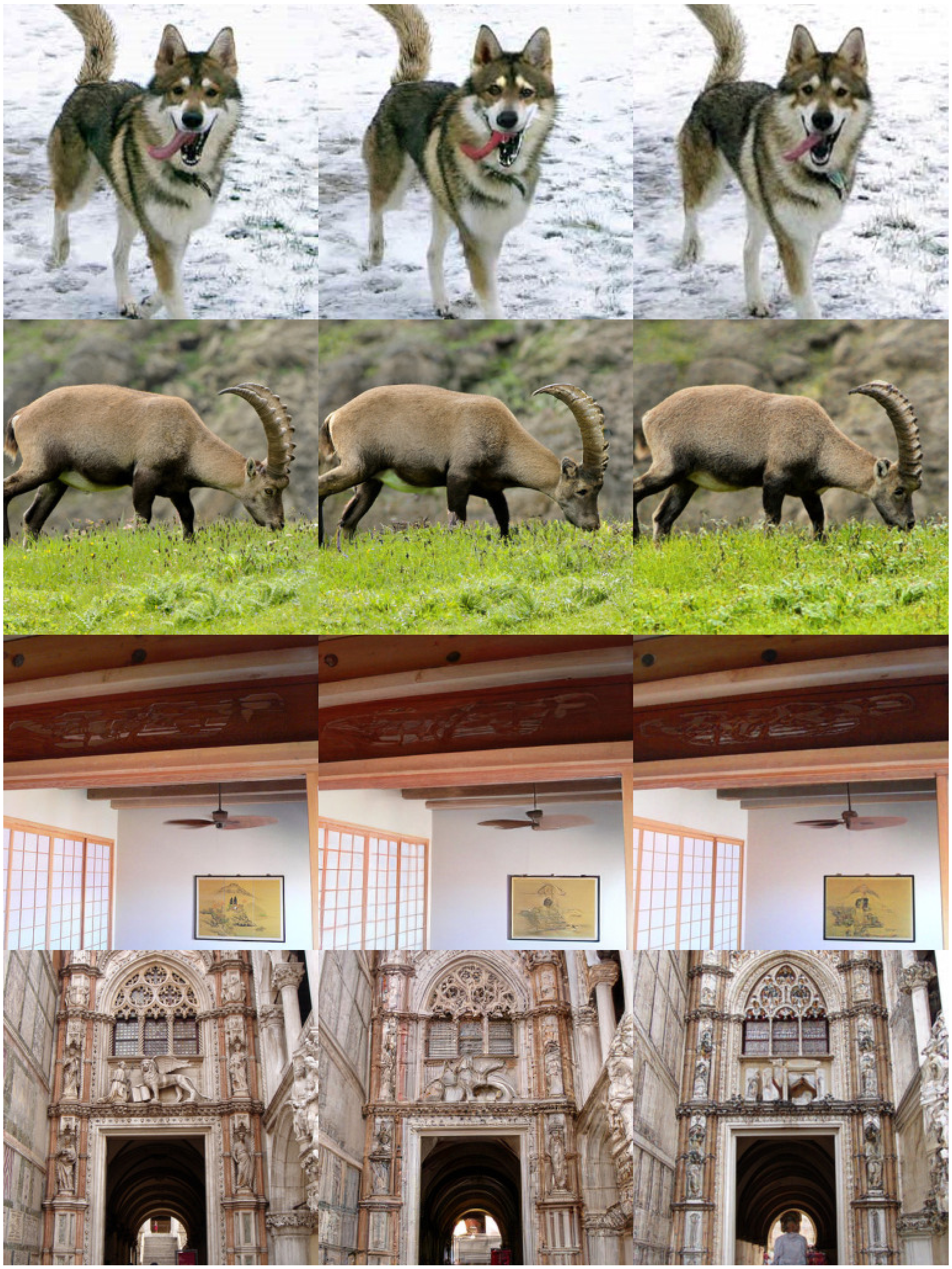
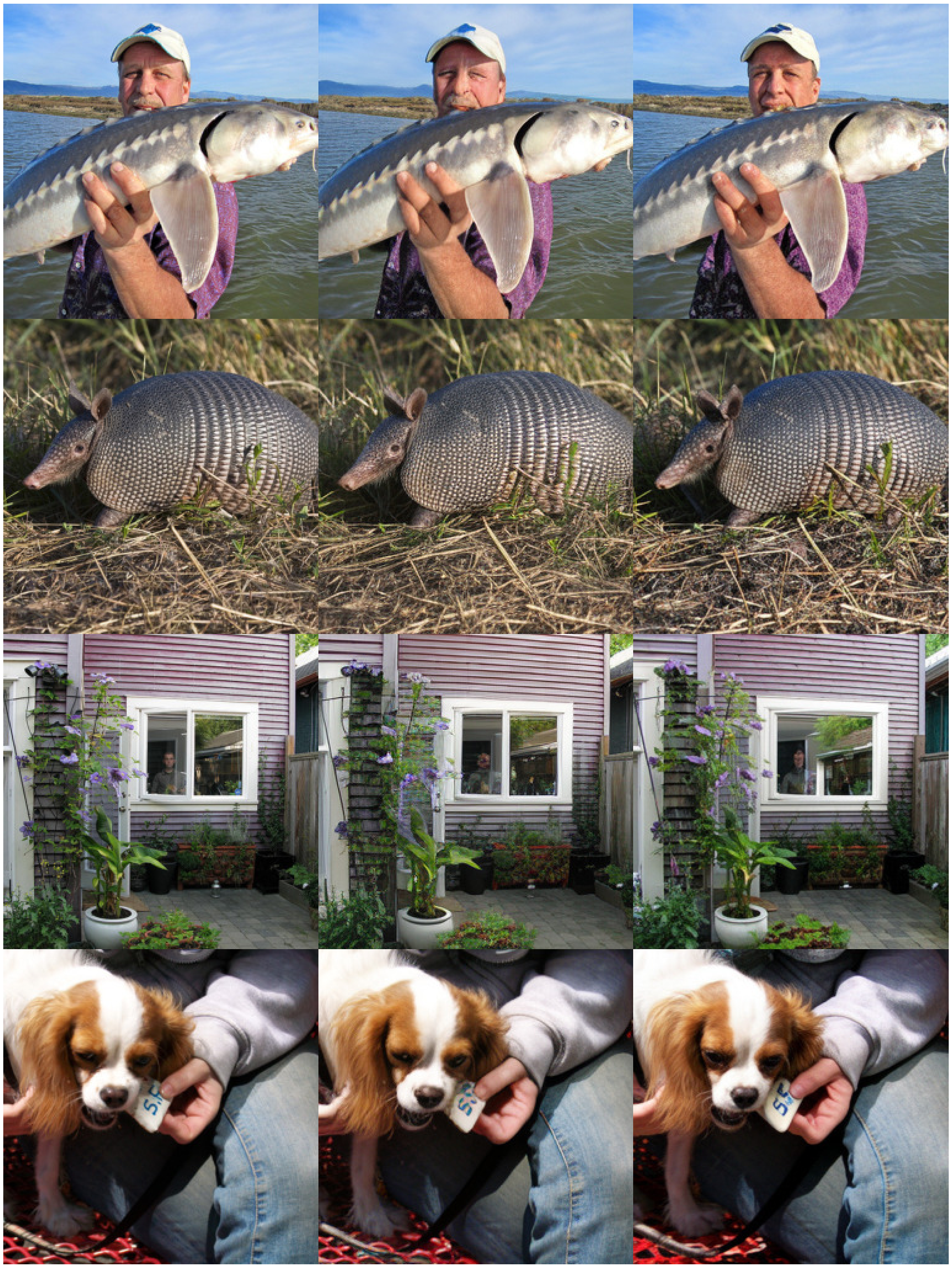


Figure 3. Tokenizer comparison, continued.



Original image

Reconstructed (LlamaGen-32 [13])

Reconstructed (FlowMo-B)

Figure 4. Tokenizer comparison, continued.

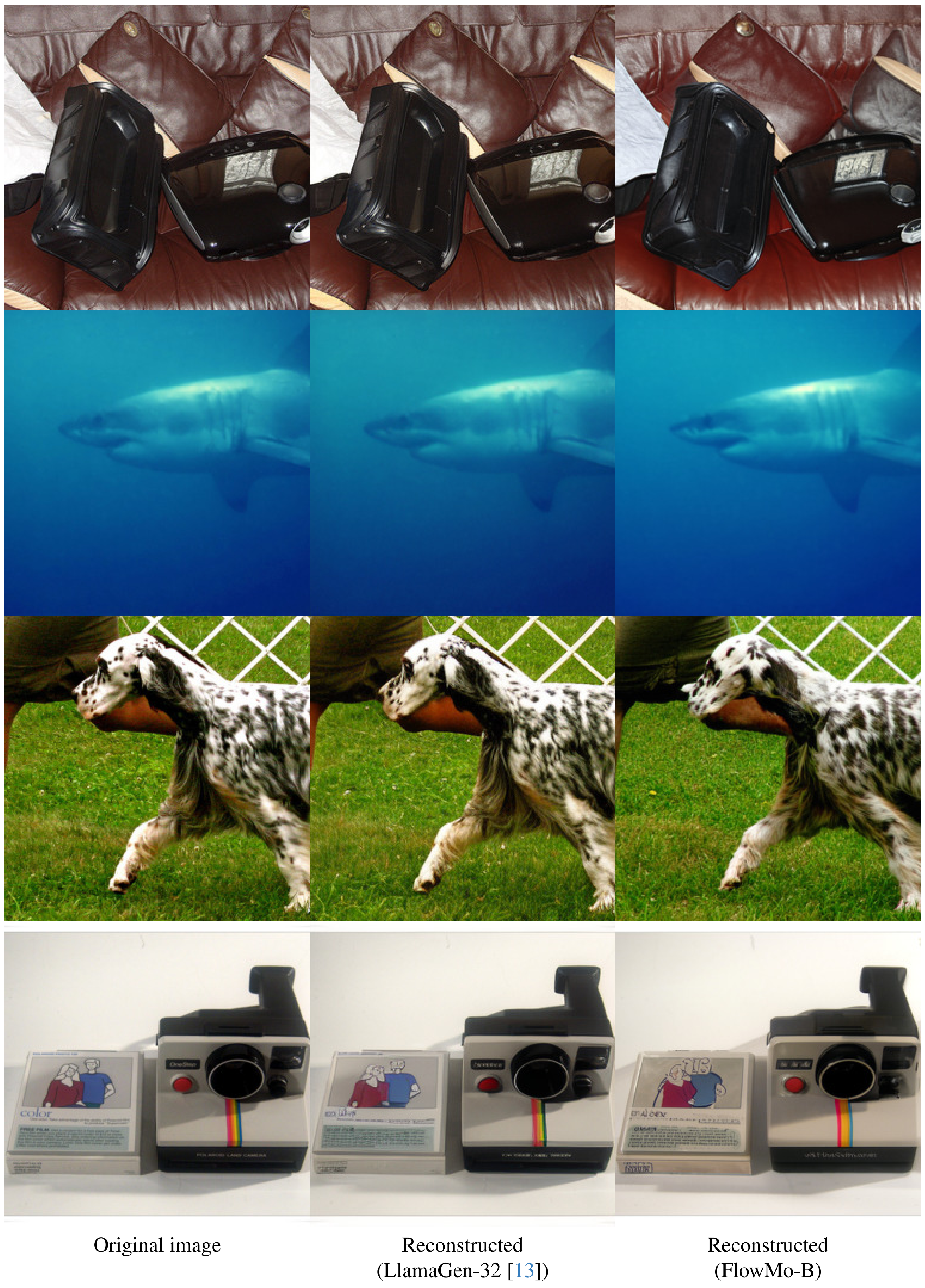


Figure 5. Tokenizer comparison, continued.