

## A Appendix

### Single-pass Adaptive Image Tokenization for Minimum Program Search

This appendix begins by summarizing our core contributions, followed by a detailed description of the experimental setup, including datasets, computational resources, and training protocols in Sec. A.2. We then present additional ablations (Sec. A.3) and extended comparisons with prior methods, ALIT and One-D-Piece (Sec. A.4). Minor main paper missing detail: First two approaches of Tab. 1 and Tab. 2 are trained on IN100 vs rest trained on IN1K.

#### A.1 Summarizing the Core Contributions

- Our first major contribution is the proposal of a **single-pass adaptive tokenizer**.

Why do we need an adaptive tokenizer that operates in a single pass? Modern large-scale models increasingly adopt hierarchical representations of varying dimensionality, allowing flexibility based on inference-time constraints such as device capability, latency, or retrieval cost. However, in practice, adaptive tokenization is often reduced to selecting a fixed number of tokens or embedding size across all data points, guided solely by downstream task requirements. This rigid strategy can be inefficient: complex images may be underrepresented with too few tokens, while simple images may receive unnecessarily verbose representations.

In this work, we propose an adaptive tokenizer that dynamically selects a variable number of tokens per image in a single forward pass, *driven not just by the downstream task but also by the intrinsic complexity of the image*. Our formulation is based on ensuring that each image reconstruction satisfies a target reconstruction quality—specifically, an  $\ell_1$  loss below a pre-defined threshold. Looking ahead, we aim to extend this framework by incorporating perceptual similarity metrics and constraints from downstream performance, enabling even more task-aware and semantically meaningful adaptivity.

- Our second contribution is the **training strategy for single-pass adaptive tokenizer**.

Prior approaches [16, 19] often rely on iterative procedures (either searching over nested embeddings or via recurrent architectures) at inference time to identify the minimal token or memory budget needed to meet a given task constraint, introducing additional computational overhead and latency.

In contrast, we propose training the tokenizer to implicitly learn the appropriate token budget in a single forward pass, conditioned on task-specific requirements. This approach is lightly inspired by principles of Kolmogorov Complexity: if a task can be accomplished using  $T$  tokens, then an additional budget of  $T + T'$  is not required to achieve the same task. We operationalize this idea by ensuring that a model trained with a budget of  $T + T'$  tokens can still satisfy the same constraint (e.g., a reconstruction loss threshold) while masking or ignoring the extra  $T'$  tokens. This allows the model to automatically determine the minimal effective token count per input, without the need for expensive runtime search at inference time.

- Additional minor contribution is **alignment of hierarchical representations with AIT**.

Finally, our third contribution is to propose a high-level conceptual bridge between *Algorithmic Information Theory* (AIT) and hierarchical or adaptive tokenization strategies. While this connection is still preliminary, we highlight an intuitive correspondence: the idea of training models to halt once sufficient representational capacity is reached echoes the principle of Kolmogorov Complexity. Similarly, the progressive allocation of tokens in hierarchical tokenizers aligns with second-order AIT concepts such as *sophistication* and *logical depth*. We emphasize that this contribution is intended as a conceptual perspective rather than a formal theoretical claim, and we view it as a first step toward further exploration of these relationships.

#### A.2 Experimental Details

**Datasets:** Throughout the paper, we utilize Imagenet or a 100-class subset of Imagenet dataset to train and evaluate our models. The 100-class subset of Imagenet is utilized by multiple prior works, serving a good dataset with decent scale (0.1M images) and lesser compute requirements. Human evaluation (Fig. 6) was done on Savioas Dataset [31] which contains human annotations of complexity on several known computer vision datasets.

Threshold ( $\epsilon$ )	% of images with recon. loss different from input condition ( $\epsilon$ )						Avg Err ( $> \epsilon$ only)
	$> \epsilon$	$> \epsilon+0.01$	$> \epsilon+0.02$	$> \epsilon+0.03$	$> \epsilon+0.04$	$> \epsilon+0.05$	
0.01	0%	0%	0%	0%	0%	0%	0.000
0.03	1%	0%	0%	0%	0%	0%	0.050
0.05	10%	3%	1%	1%	0%	0%	0.060
0.07	25%	12%	6%	4%	2%	1%	0.086
0.09	18%	9%	5%	3%	2%	1%	0.107
0.11	36%	19%	10%	5%	3%	2%	0.126

Table 3: **Analysis of the loss-conditioned training strategy for single-pass adaptive tokenization:**

Only a small fraction of images exceed the input target reconstruction loss threshold ( $\epsilon$ ) when masking is applied. The columns indicate the fraction of images exceeding  $\epsilon$  by more than specified margin values. The last column shows the average reconstruction error for these images, which remains only marginally higher than the input threshold.

**Compute Requirements:** Majority of the training was done on single A100 or H100 machine with eight 80GB gpus or on machines with equivalent GPU memory (distributed training on 4 H200 gpus).

**Training procedure:** The overall training framework consists of jointly optimizing two objectives: maximizing token utilization with the *aim of lossless compression*, and minimizing unnecessary input memory or token usage with the *aim of minimal program search* for an image —

**Details regarding loss-conditioning:** We initialize a list of discrete target L1 reconstruction loss values. This list includes small values (0.0, 0.01, 0.02) to encourage near-lossless compression, uniformly sampled moderate values in the range 0.03–0.11, and a few higher values sampled from the range 0.14–0.4. Each loss value is associated with an embedding vector of the same dimensionality as the latent-distillation encoder’s input tokens. These embeddings serve as conditioning signals and are concatenated as an additional token to the encoder input.

During training, the number of input tokens is randomly selected from  $\{16, 32, \dots, 384\}$ —i.e., multiples of 16. For the initial *lossless compression run*, a small loss value from the list is used as the condition. Once the reconstruction loss for this run is computed, we map it to the smallest discrete loss value in the list that is strictly greater than the attained loss. The corresponding loss embedding is then used to condition a second training run focused on *minimum program search*.

In this second run, we sample additional tokens—ranging from 16 to 256—to be appended to the original input. The total number of input tokens is fixed at 256, so we add a maximum of  $\max(0, 256 - \text{tokens used in the lossless run})$  extra tokens. Only these additional tokens are subject to halting supervision. Specifically, we supervise the encoder to assign a high halting probability (close to 0) to the extra tokens. The training objective is a binary cross-entropy loss: tokens used in stage 1 are assigned a "keep" label (1), while the added tokens are assigned a "halt" label (0) and are expected to be masked out.

In cases where no extra tokens are added in the second run (i.e., the first stage already uses the full 256-token budget), we augment training by randomly sampling loss values that are smaller than the one attained during the lossless run. This ensures the model still receives diverse training pairs of (target loss condition, input token budget), even when no masking occurs in the second stage.

Tab. 3 evaluates the learned tokenizer’s ability to meet the input reconstruction loss condition—specifically, how often the reconstructed image satisfies the loss threshold when a portion of the input tokens is masked. When no tokens are masked, some images may still exceed the loss threshold due to an insufficient token budget. As shown in the table, only a very small fraction of the 5000 IN100 validation images exhibit a reconstruction loss greater than 0.03 beyond the input threshold when masking is applied. For images that utilize only a subset of the 256 input tokens yet exceed the input loss condition ( $\epsilon$  in the algorithm from Fig. 1), the average reconstruction error is only marginally above the threshold. These results validate the efficacy of our training approach.

**Details regarding multi-stage training pipeline:** Inspired by recent works [26, 19], the training pipeline proceeds in two stages: a latent distillation phase and a GAN-based finetuning stage. In the **latent distillation pretraining stage**, a pre-trained image tokenizer (either a VQGAN or a VAE) maps input images to 2D token grids. The latent distillation encoder-decoder modules then compress these 2D tokens into 1D representations and reconstruct them back, with the reconstruction loss over the 2D token space serving as the main learning objective. Both the encoder and decoder perform full self-attention – encoder operating on a concatenated 2D image tokens and initialized 1D tokens

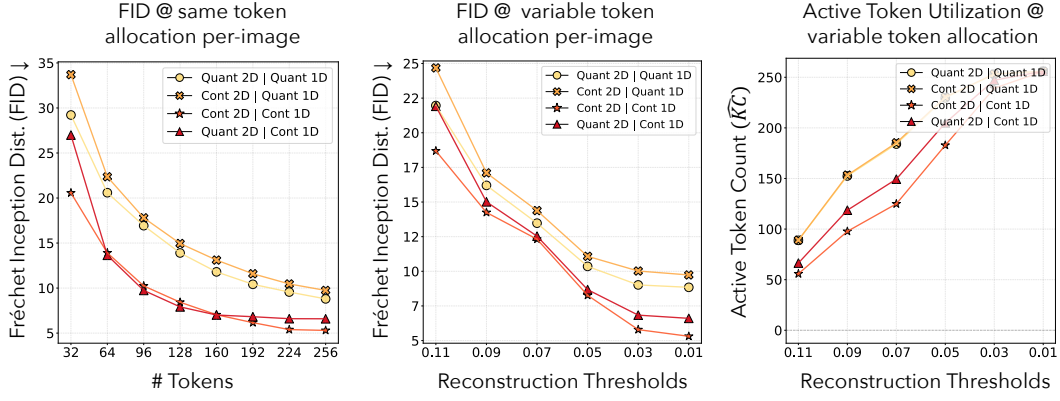


Figure 7: **Scaling laws for continuous vs discrete tokenization:** Cont1D is most crucial for FID (Plot 1 & 2). Cont2D (VAE) with Cont1D leads to shortest representation ( $\bar{K}\bar{C}$ ) on average (Plot 3).

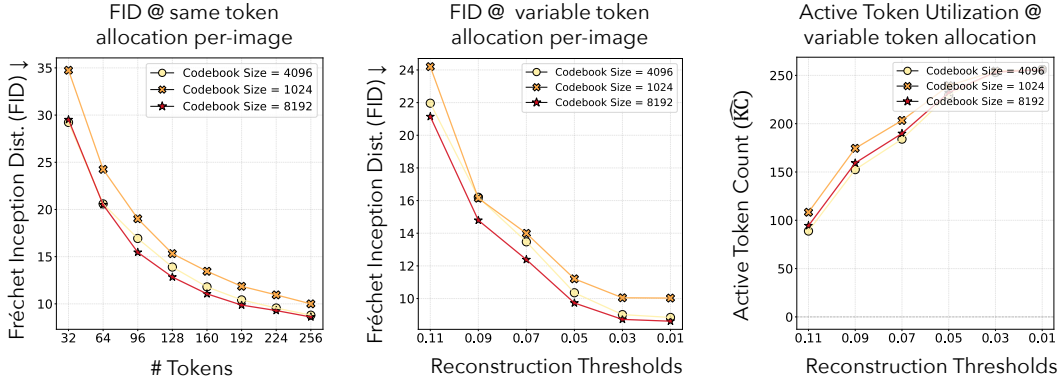


Figure 8: **Scaling laws for 1D token codebook size:** Larger codebooks (4096 and 8192 codes) lead to shorter representation length ( $\bar{K}\bar{C}$ ) on average because of code specialization.

and the decoder operates on learned 1D tokens concatenated with masked 2D tokens (for inpainting). When using VQGAN as the base tokenizer, we apply a cross-entropy loss between the predicted logits from the latent distillation module and the ground-truth VQGAN codebook indices at each 2D token position. When using a VAE as the base tokenizer, we instead use a mean squared error (MSE) reconstruction loss between the predicted and ground-truth token embeddings. In the **GAN-based finetuning stage**, the reconstruction losses shift from intermediate 2D image token space to the pixel-level image space, with additional GAN-based adversarial loss and LPIPS perceptual loss. Quantization-based losses (commitment and codebook loss) similar to prior works are also added. We perform quantization on a factorized 12-dimension embedding (factorization of encoder output before quantization helps as shown by prior works). With recent works [27, 17] adopting diffusion auto-encoders instead of gan-based adversarial losses, exploring diffusion-loss with KARL would be an exciting future research avenue.

### A.3 Scaling Laws & Ablations

Most adaptive tokenization works [19, 17] present scaling laws by varying the number of tokens per image, but they typically assign the same token count to all images for a given configuration. As noted in Core Contribution #1, the ideal goal of an adaptive tokenizer is to allocate a variable number of tokens per image at test time, depending on the content or complexity of each image. To capture both perspectives, we design scaling laws in two ways: (a) uniform token allocation across images, as done in prior work, and (b) variable token allocation based on a target reconstruction loss. Additionally, inspired by the Algorithmic Information Theory (AIT) notion of Kolmogorov Complexity, we analyze scaling laws from a KC perspective—studying how well we can minimize the token count (i.e., program length) per image while preserving recons. quality, target  $\ell_1$  loss.

Before delving into the scaling laws, we clarify an important detail: ideally, the Kolmogorov Complexity (KC) of a data point includes the full length of the program required to generate it.

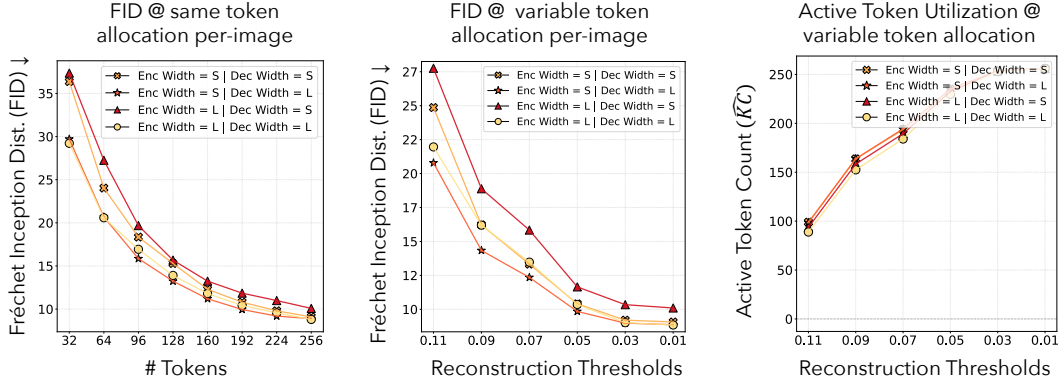


Figure 9: **Scaling laws for encoder/decoder width (feature dimensionality):** A *Small Encoder with a Large Decoder* consistently achieves the best performance. Notably, variable token allocation per image enables a more informative comparison, clearly distinguishing the *Small Encoder, Large Decoder* configuration from the *Large Encoder, Large Decoder* variant.

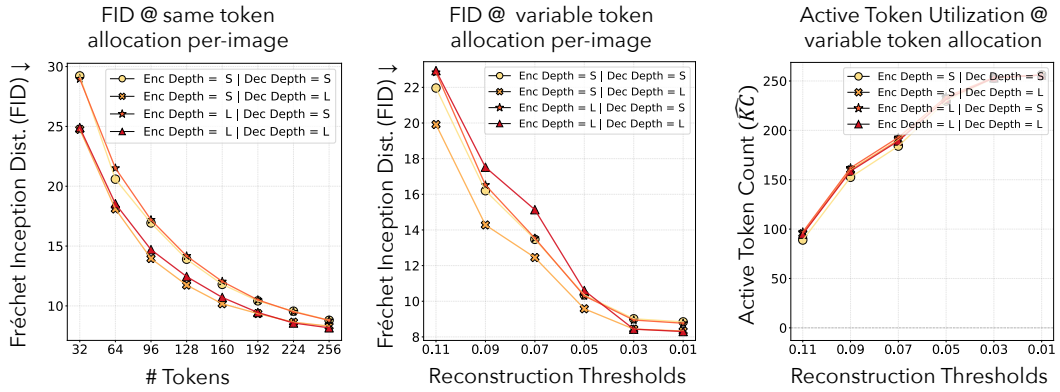


Figure 10: **Scaling laws for encoder/decoder depth (number of layers):** Similar to the width ablation, the *Small Encoder with a Large Decoder* configuration yields the best performance at the same average active token count as other variants. This distinction becomes more evident when using variable token allocation per image, as opposed to fixed token counts.

870 In our context, this would encompass not only the number of encoded tokens but also the token  
871 feature dimensionality, decoder size, and codebook size. However, since these latter components are  
872 amortized across many images, we approximate KC using only the token count and refer to this as  
873  $\hat{K}\hat{C}$  throughout the paper. Note that the encoder is not part of the generative program—it acts as a  
874 program synthesizer rather than the program itself.

875 **Continuous vs Discrete Tokenization:** We compare continuous and discrete adaptive tokenization  
876 approaches at both 1D and 2D token levels under both fixed and variable token allocation regimes.  
877 Key observations from Fig. 7 include (new or significant findings highlighted in green):

- 878 • Continuous 1D tokens consistently outperform quantized 1D tokens in terms of reconstruction  
879 quality (measured by FID) across nearly all token counts—echoing observations of prior works.
- 880 • This performance advantage holds even when token counts are allocated adaptively per image  
881 based on a target reconstruction loss.
- 882 • Continuous 1D tokenizers achieve better reconstruction at lower average token counts (i.e.,  
883 lower estimated  $\hat{K}\hat{C}$ ), highlighting their efficiency in capturing image complexity.
- 884 • The choice of 2D tokenizer (continuous VAE vs. discrete VQGAN) has *minimal* impact on the  
885 minimum token count required to reach a target loss—when the 1D tokens are quantized.
- 886 • However, when 1D tokens remain continuous, pairing them with a discrete 2D tokenizer (e.g.,  
887 VQGAN) leads to *higher*  $\hat{K}\hat{C}$  across all target loss levels, potentially because more tokens are  
888 required to compensate for the loss of detail at 2D token level.

Approach	FID @ variable token regime ( $\downarrow$ )			FID @ same token regime ( $\downarrow$ )		
	0.03	0.05	0.09	32	128	256
ALIT	8.4	9.3	13.5	<b>22.6</b>	11.7	<b>8.2</b>
KARL	9.1	10.3	16.2	28.7	13.6	8.8
One-D-Piece	<b>8.2</b>	<b>8.3</b>	<b>10.5</b>	24.1	<b>9.7</b>	8.2

Table 4: **FID Comparison on IN100:** While KARL slightly trails prior approaches in FID, it outperforms them on single-image metrics that better align with the Kolmogorov Complexity (KC) perspective—favoring the shortest possible representation for each individual data point rather than alignment with dataset-level statistics. As is well known, FID is not a reliable metric for assessing reconstruction quality. Visual results in Fig. 2 Fig. 11 Fig. 12 Fig. 13 and Fig. 14 clearly demonstrate that both ALIT and KARL produce reconstructions that are visually superior to One-D-Piece. Notably, KARL is the only single-pass method for adaptive tokenization and achieves the best performance on single-image quality metrics. FID could likely be improved through additional training techniques—for instance, using separate GAN discriminators for different representation lengths or token counts, as done in ALIT, instead of a shared one. Moreover, conditioning masking on perceptual losses such as LPIPS or DreamSim, rather than  $\ell_1$  loss, may further enhance FID.

**1D Token Codebook Size ablations:** From Fig. 8 we observe that larger codebooks consistently yield lower (i.e., better) FID scores. Note: Prior works have shown that this trend holds only up to a certain codebook size, beyond which codebook utilization drops significantly. In our ablations, *the average number of active tokens per image ( $\hat{K}C$ ) is lowest for codebook sizes 4096 and 8192, much lower than  $\hat{K}C$  for the smallest codebook size of 1024*. This is intuitive—smaller codebooks offer fewer specialized codes, requiring more tokens to achieve the same representational fidelity.

**Encoder and Decoder Depth / Width Ablations:** As highlighted in the main paper and shown in Fig 9 and Fig 10, the scaling laws indicate that the best-performing architecture features a *small encoder and a large decoder*. This is intuitive: the encoder’s role is to distill the already abstracted 2D tokens from the VQGAN into 1D tokens, while the decoder takes on the more challenging task of inpainting 1D tokens into a masked slate.

A key observation is that the variable token allocation strategy enhances the separation between architectural variants. Specifically, it reveals a more pronounced performance gap between the Small Encoder, Large Decoder and Large Encoder, Large Decoder configurations. When the same number of tokens is assigned to all images — regardless of image complexity — high-complexity images may perform poorly under limited token budgets, making it harder to distinguish between architectural strengths. In contrast, when token counts are adaptively allocated per image, each model can adjust token usage to meet reconstruction targets, enabling fairer comparison and clearer differentiation.

The average number of active tokens remains nearly constant across ablations. The Small Encoder, Large Decoder setup (small in both width and depth) performs best under this similar active token utilization. **Note:** For the width ablations, both the depth (i.e., number of transformer layers) and the number of heads (in multi-head attention) are held constant. Conversely, in the depth ablations, the width is fixed to a large value, and the number of heads scales proportionally with the depth.

#### 912 A.4 Additional Comparison with Prior Adaptive Tokenizers

Beyond the comparisons presented in the main paper, we further evaluate KARL against ALIT, a recurrent adaptive tokenizer, and One-D-Piece, a Matryoshka-based tokenizer, under both fixed per-image token allocation and variable token allocation regimes.

Fig. 11 and Fig. 12 clearly demonstrate the superior performance of ALIT and KARL over One-D-Piece, which frequently produces blurry reconstructions (see the second column in both figures). Such blurry outputs misleadingly yield good FID scores (see Tab. 4) while performing poorly on per-image metrics such as LPIPS, SSIM, PSNR, and DreamSIM. This observation *underscores a limitation of FID: it may not be an ideal metric when the focus is on accurate per-image reconstruction*. Furthermore, from a Kolmogorov Complexity (KC) perspective, the goal of tokenization is to produce the shortest representation that faithfully captures the individual data point—not merely the distribution. In this regard, KARL achieves the best performance on most per-image reconstruction



metrics, slightly outperforming ALIT. Unlike ALIT, which requires multiple recurrent encoder passes, KARL determines the minimal token count in a single forward pass.

Fig. 13 and Fig. 14 compare different adaptive tokenizers under a variable token allocation regime, where each image is assigned a different number of tokens based on a target reconstruction loss threshold. For One-D-Piece, this involves a single encoder pass but multiple decoder runs to find the shortest embedding that achieves reconstruction  $\ell_1$  loss below the target. In contrast, ALIT performs a joint encoder-decoder pass at each iteration, requiring multiple iterations until the stopping criterion is satisfied. See Tab. 2 in the main paper for additional details.

Thanks to its recurrent training and explicit test-time search, ALIT performs best under a relaxed reconstruction criterion of  $\ell_1$  loss  $< 0.09$  (as shown in Fig. 13). However, KARL performs competitively—often surpassing ALIT on per-image metrics such as LPIPS, SSIM, and  $\ell_1$  loss.

Fig. 14 analyses reconstructions under a tighter constraint of  $\ell_1$  loss  $< 0.05$ . KARL not only achieves better or comparable performance on these metrics but also does so with equal or fewer tokens, particularly when compared to One-D-Piece. Based on our experiments, we conjecture that KARL with an input condition of  $\epsilon = 0.05$  represents the most practical configuration for adaptive tokenization in real-world settings.

**Figures in the next pages.**



Figure 11: **Reconstruction Comparison on IN100 (visualization 1/2):** KARL and ALIT produce higher-quality reconstructions than One-D-Piece, with KARL achieving strong single-image metrics. One-D-Piece consistently generates blurry images at low token counts satisfying FID metric more.



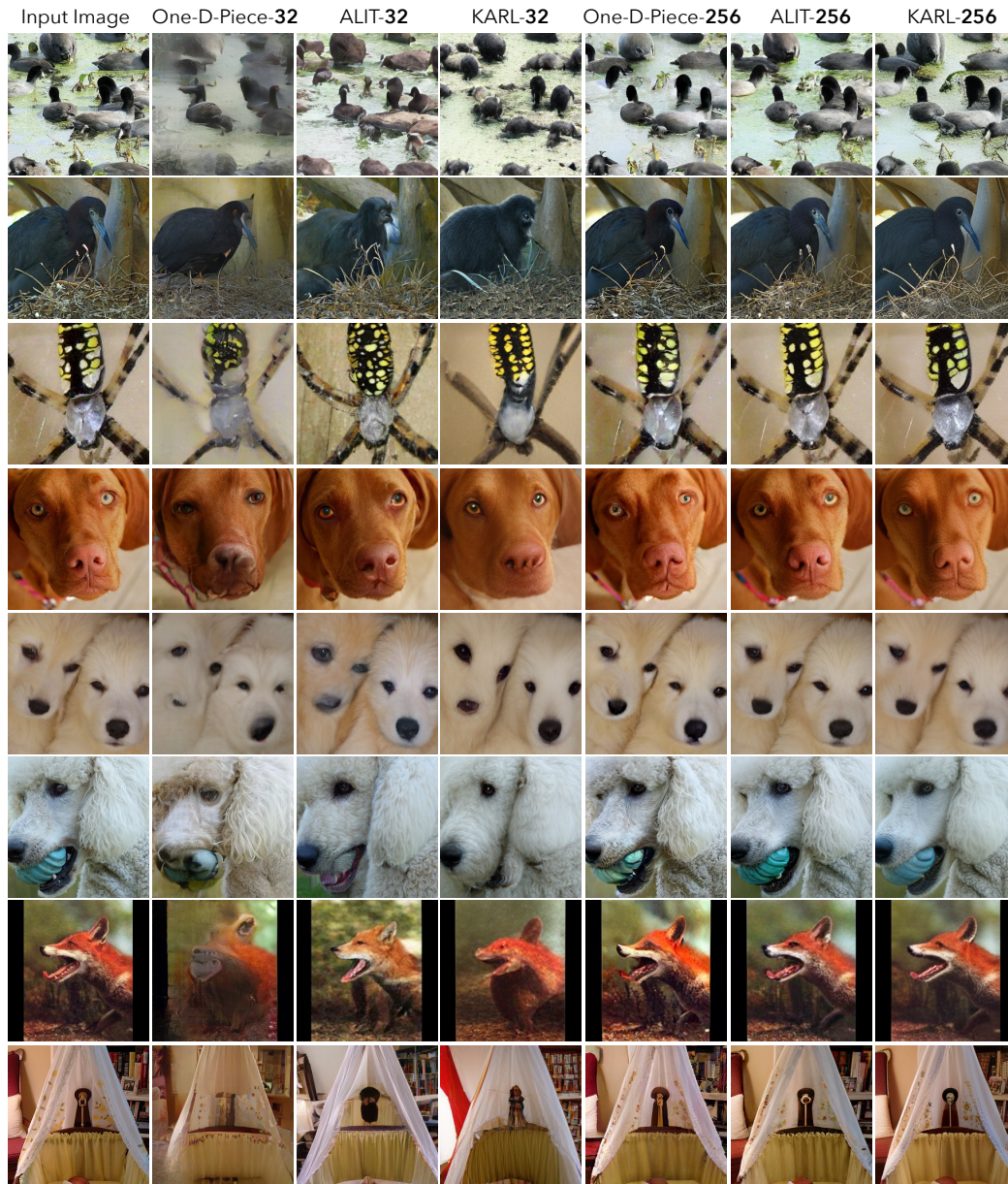
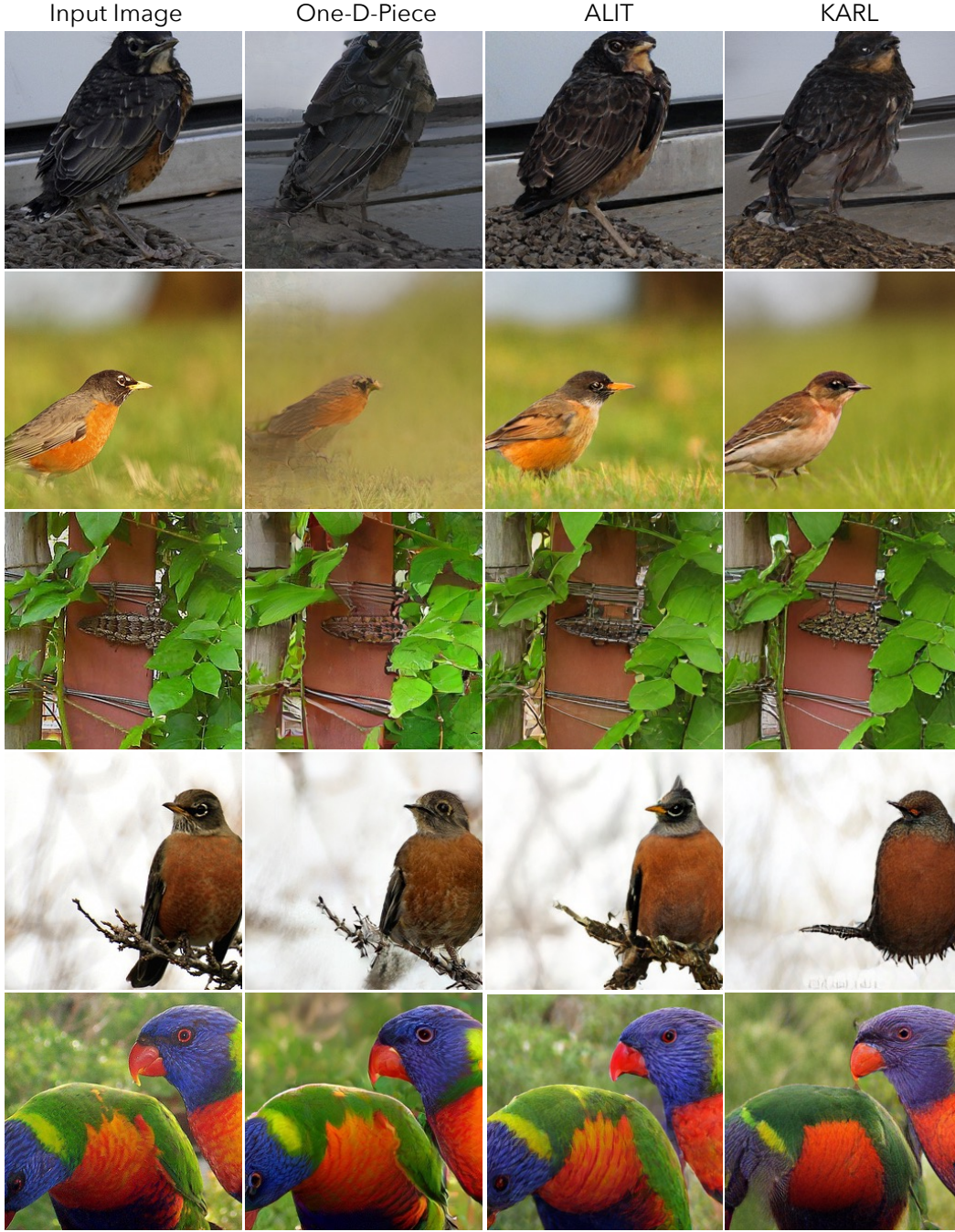


Figure 12: **Reconstruction Comparison on IN100 (visualization 2/2):** KARL and ALIT produce higher-quality reconstructions than One-D-Piece, with KARL achieving strong single-image metrics. One-D-Piece consistently generates blurry images at low token counts satisfying FID metric more.





Images	One-D-Piece				ALIT				KARL			
	$\hat{K}C$ ( $\downarrow$ )	$L1 \times 10$ ( $\downarrow$ )	SSIM ( $\uparrow$ )	LPIPS ( $\downarrow$ )	$\hat{K}C$	$L1 \times 10$	SSIM	LPIPS	$\hat{K}C$	$L1 \times 10$	SSIM	LPIPS
Row 1	<b>44</b>	0.88	0.35	0.42	64	<b>0.68</b>	<b>0.41</b>	<b>0.30</b>	64	0.82	0.36	0.36
Row 2	<b>12</b>	0.87	0.61	0.42	32	0.59	0.61	0.27	32	<b>0.59</b>	<b>0.65</b>	<b>0.26</b>
Row 3	256	1.07	0.27	0.40	<b>192</b>	0.89	0.31	0.36	224	<b>0.83</b>	<b>0.37</b>	<b>0.30</b>
Row 4	75	0.90	<b>0.59</b>	<b>0.32</b>	<b>32</b>	<b>0.85</b>	0.58	0.35	64	1.00	0.59	0.37
Row 5	256	0.92	0.32	0.34	<b>128</b>	<b>0.79</b>	<b>0.32</b>	<b>0.34</b>	<b>128</b>	0.87	0.30	0.38

Figure 13: **Comparison of One-D-Piece, ALIT, and KARL with variable token allocation per image satisfying reconstruction  $\ell_1 \times 10$  loss  $< 0.9$ :** At this reconstruction loss threshold, the goal is **not perfect reconstruction**, but rather to evaluate which method achieves acceptable reconstruction quality within the target constraint while using the fewest tokens (i.e., minimizing  $\hat{K}C$ ). ALIT and KARL use fewer tokens than One-D-Piece for acceptable recons., satisfying  $\ell_1 \times 10$  loss  $< 0.9$ .



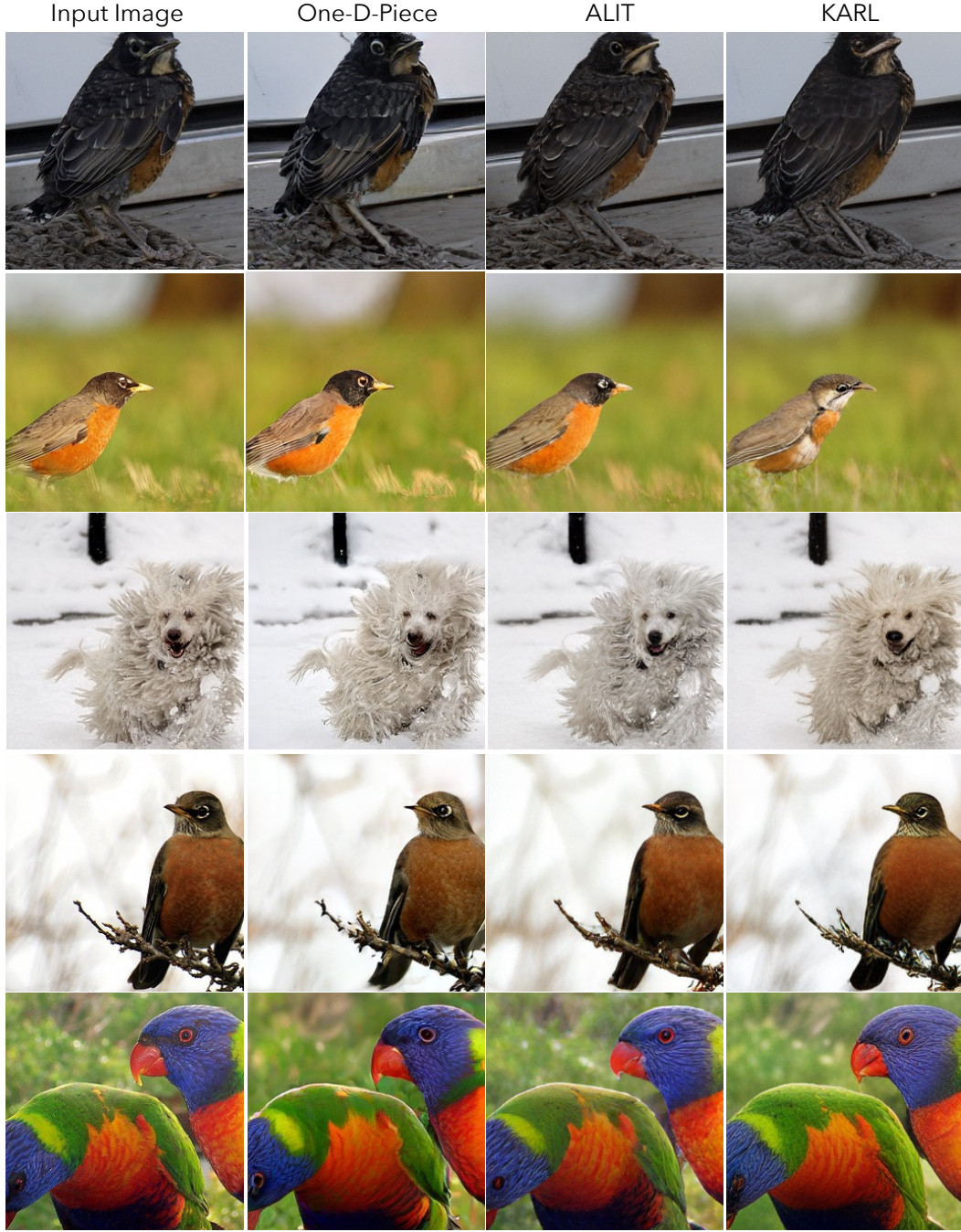


Image	One-D-Piece				ALIT				KARL			
	$\hat{K}\hat{C}$ ( $\downarrow$ )	L1 $\times$ 10 ( $\downarrow$ )	SSIM ( $\uparrow$ )	LPIPS ( $\downarrow$ )	$\hat{K}\hat{C}$	L1 $\times$ 10	SSIM	LPIPS	$\hat{K}\hat{C}$	L1 $\times$ 10	SSIM	LPIPS
Row 1	256	0.81	0.41	0.30	256	0.56	0.47	0.26	256	<b>0.51</b>	<b>0.51</b>	<b>0.22</b>
Row 2	256	0.54	<b>0.69</b>	<b>0.21</b>	<b>64</b>	<b>0.47</b>	0.65	0.22	96	0.49	0.67	0.23
Row 3	256	0.66	0.50	0.31	256	0.53	0.53	0.25	<b>224</b>	<b>0.49</b>	<b>0.58</b>	<b>0.22</b>
Row 4	256	0.64	0.64	0.26	<b>192</b>	0.49	0.67	0.22	224	<b>0.48</b>	<b>0.71</b>	<b>0.19</b>
Row 5	256	0.92	0.32	0.34	256	0.69	0.35	0.30	256	<b>0.65</b>	<b>0.39</b>	<b>0.25</b>

Figure 14: **Comparison of One-D-Piece, ALIT, and KARL with variable token allocation per image satisfying reconstruction  $\ell_1 \times 10$  loss  $< 0.5$ :** At this reconstruction loss threshold, the goal becomes much closer to near **perfect reconstruction** while using the fewest tokens ( $\hat{K}\hat{C}$ ). ALIT and KARL often save a few tokens from the full 256 token budget, satisfying  $\ell_1 \times 10$  loss  $< 0.5$ . Threshold= 0.05 can serve as a de-facto threshold for majority of vision tasks.