BUDGETPIX: COMPUTE-ADAPTIVE TOKENIZATION FOR PIXEL-SPACE IMAGE DIFFUSION

Anonymous authors
Paper under double-blind review

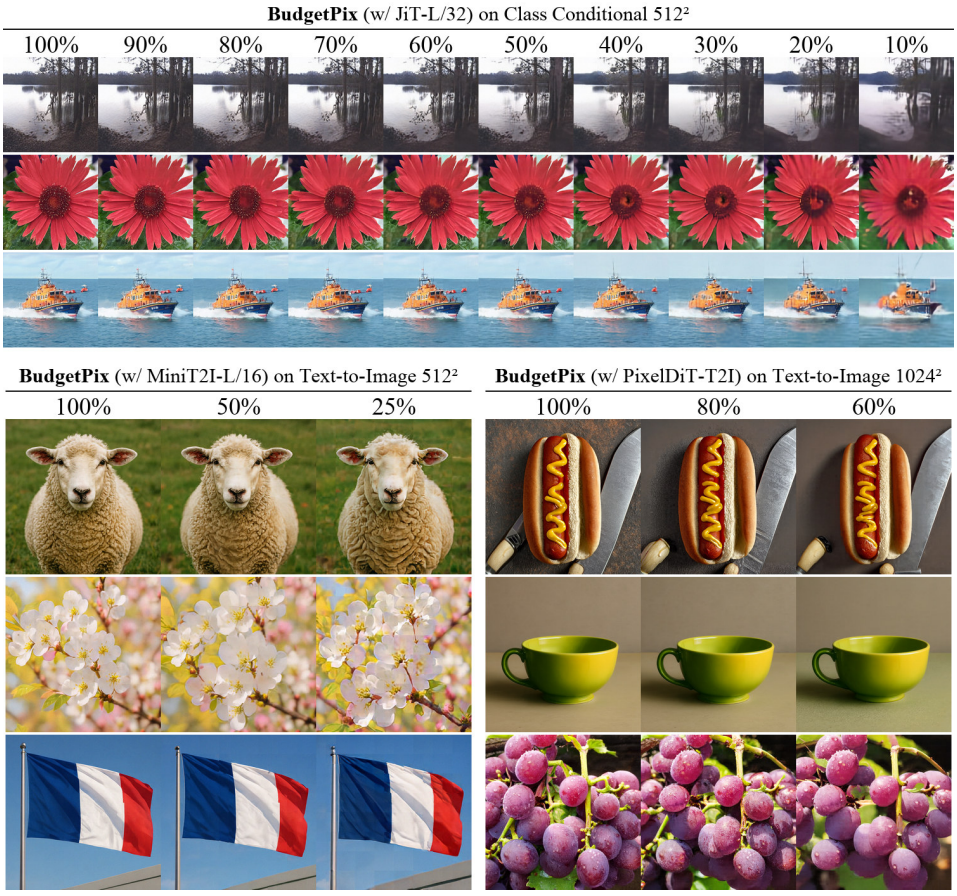

Figure 1: **Test-time compute adjustment.** BudgetPix dynamically scales inference costs, achieving a strong quality-efficiency tradeoff across JiT (top), MiniT2I (bottom left), PixelDiT (bottom right), 3 popular architectures for pixel-space image diffusion. Percentages show the fraction of tokens used.

ABSTRACT

Most image generation models rely on uniform tokenization, allocating the exact same computational budget to equally-sized image patches. This static paradigm cannot adapt to different resource constraints at inference time, and yields suboptimal quality-cost tradeoff by devoting the same effort to both plain backgrounds and intricate details. We propose BudgetPix, an adaptive tokenization framework that dynamically allocates compute based on visual complexity and spatial layout, enabling flexible computational budgeting at inference time. BudgetPix comprises three key components: (1) an adaptive encoder that maps a fixed-size image to a variable-length token sequence using an entropy-guided quadtree alongside a multi-scale patch embedder; (2) a scale-aware decoder reconstructs fixed-resolution images from multi-scale token sets; and (3) a flexible training and sampling schedule that enables pixel-space denoisers to operate across variable token

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

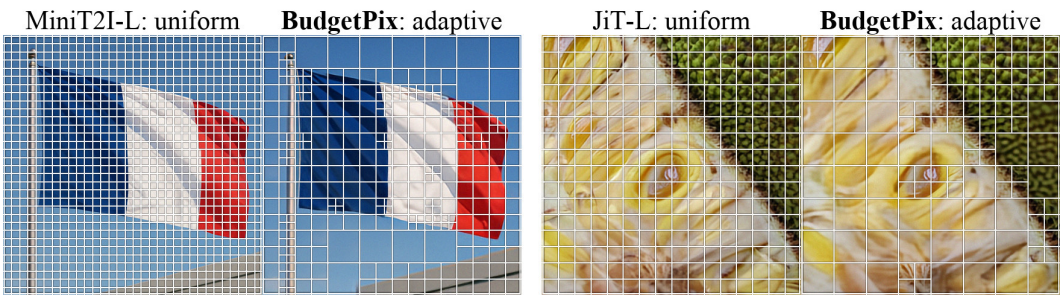

Figure 2: **Uniform vs. content-adaptive tokenization.** For a 512×512 image, uniform tokenization requires 1,024 tokens with 16×16 patches (a) and 256 tokens with 32×32 patches (c). BudgetPix adapts to visual content, requiring only 307 (b) and 127 (d) tokens. Baselines in (a) and (c) represent full-budget MiniT2I-L and JiT-L; BudgetPix in (b) and (d) operates at 30% and 50% budgets, respectively.

counts. BudgetPix seamlessly integrates with existing pixel-space diffusion architectures, enabling a single checkpoint to be operated at a wide range of compute budgets. Evaluated on text-to-image generation, BudgetPix matches the fidelity of MiniT2I-L at 512^2 (GenEval: 0.874 vs. 0.882) and PixelDiT at 1024^2 (0.725 vs. 0.721) using just 25% of the original compute budget. In class-conditional using a MeanFlow backbone, BudgetPix requires merely 60% of the full compute budget to produce images with near-zero quality degradation, observing a marginal 0.8-point increase in FID (3.46 vs. 2.65). Comprehensive assessments by human and VLM judges confirm that BudgetPix establishes a significantly improved quality-efficiency tradeoff over prior budget-adaptive baselines.

1 Introduction

Diffusion Transformers (DiTs) (Peebles & Xie, 2023) have established a new standard for image generation. While most architectures operate in the latent space of a pretrained autoencoder (Rombach et al., 2022), emerging pixel-space DiTs (Li & He, 2026; Yu et al., 2026) process raw pixels directly, bypassing the reconstruction ceiling and eliminating the reliance on a separate tokenizer. However, this architectural purity introduces a severe computational bottleneck: the uniform tokenization. Because self-attention scales quadratically with the number of tokens, pixel-space models incur massive computational costs. Furthermore, because this grid resolution is hardcoded during training, the inference cost becomes rigidly fixed. A deployed model cannot scale down to accommodate resource-constrained edge devices, nor can it dynamically adjust to varying user latency requirements, without undergoing entirely separate retraining.

This static uniform tokenization is fundamentally misaligned with how visual data is structured. Because images are heterogeneous in their information density, a cloudless sky presents minimal visual entropy, while heavily textured objects demand the resolution of dense, high-frequency details. Forcing a uniform grid to blindly expend identical compute across both regions creates severe computational waste. As shown in Figure 2a, standard MiniT2I (Wang et al., 2026b) wastes 1,024 tokens rendering a simple flag against an open sky, whereas our content-adaptive framework reconstructs the scene perfectly with just 307 tokens (Figure 2b). Even for complex, edge-to-edge textures like a jackfruit—which strictly requires 256 tokens in standard JiT (Li & He, 2026) (Figure 2c)—BudgetPix matches the baseline visual quality while cutting the required token budget by 50% (Figure 2d).

Content-adaptive tokenization is well established in discriminative vision models via token merging (Bolya et al., 2023), token pruning (Rao et al., 2021), or dynamic multi-scale representations (Beyer et al., 2023; Dehghani et al., 2023; Kim et al., 2025; Choudhury et al., 2026). However, extending this paradigm to generative models introduces a fundamental challenge. While discriminative models dynamically allocate tokens over an *already-provided* input image to predict a class label (Choudhury et al., 2026), generative denoisers must dynamically allocate compute over an image

that is *actively being generated*. This creates a chicken-and-egg dilemma: the denoiser must gauge the local spatial complexity of the image before the image content itself has fully materialized.

In this work, we propose BudgetPix, a framework that adapts any pretrained pixel-space diffusion transformer into a compute-adaptive image generator. BudgetPix introduces three novel components. First, an entropy-driven encoder maps images into variable-length sequences of multi-scale tokens. Second, a complementary decoder reconstructs these variable-length sequences back into fixed-resolution images. Finally, a training and inference strategy, coupled with content-layout augmentation, enables standard checkpoints trained on fixed token budgets to become dynamically adjustable during inference. Our extensive experiments (Section 4.3) demonstrate that BudgetPix can be integrated seamlessly with established pixel-space architectures–including JiT (Li & He, 2026), MiniT2I (Wang et al., 2026b), PixelDiT (Yu et al., 2026), and MeanFlow (Geng et al., 2026)–and provide a significantly-better quality-efficiency tradeoff (see plots in Section 4.3).

We evaluate BudgetPix on class-conditional and text-to-image pixel-space diffusion models across resolutions ranging from 256^2 to 1024^2. Under strict low-compute budgets where prior budget-adaptive methods undergo catastrophic collapse, BudgetPix degrades gracefully, preserving high perceptual quality. For class-conditional generation, BudgetPix synthesizes coherent, high-quality images using as little as 10% of the original compute budget (Figure 1, top 3 rows)–a regime where competing methods fail entirely (cf. Figure 6). This robustness mirrors phenomena observed in lossy image compression, where visual data can be heavily down-sampled without significant perceptual degradation.

Quantitatively, for text-to-image generation, BudgetPix matches the fidelity of full-compute baselines while utilizing only 25% of the original token budget, achieving GenEval scores of 0.874 (vs. 0.882) on MiniT2I-L at 512^2, and 0.725 (vs. 0.721) on PixelDiT at 1024^2. For class-conditional generation, integrating BudgetPix into a MeanFlow (Lu et al., 2026) backbone allows the model to operate at merely 60% of the original compute budget with negligible degradation in visual quality, observing a minimal FID increase of only 0.81 points (from 2.65 to 3.46). Finally, comprehensive evaluations by both human and Vision-Language Model (VLM) judges confirm that BudgetPix achieves a better quality-efficiency tradeoff compared to existing budget-adaptive methods.

Our primary contributions are:

- We propose BudgetPix, the first[1] compute-adaptive tokenization framework for pixel-space image diffusion, designed to integrate smoothly into diverse baseline architectures such as JiT(Li & He, 2026), MiniT2I (Wang et al., 2026b), PixelDiT (Yu et al., 2026), and MeanFlow (Lu et al., 2026).
- BudgetPix consists of three novel components: (i) an entropy-based encoder that maps images into variable-length sequences of multi-scale tokens; (ii) a specialized decoder that reconstructs the fixed-resolution image from these multi-scale tokens; and (iii) a flexible training and sampling schedule that allows denoisers to operate across continuously variable token counts.
- Comprehensive experiments show that BudgetPix achieves effective quality-efficiency tradeoff compared with alternative adaptive denoising methods.

2 Related work

Pixel-space diffusion models. For several years the cost of resolution was handled by not paying it: latent diffusion generates inside a pretrained autoencoder (Rombach et al., 2022), at the price of a reconstruction ceiling and a second training stage. Pixel-space models historically managed it with U-Nets, omitting attention at all but the lowest-resolution levels of the hierarchy, in cascades (Ho et al., 2022) and single models whose noise schedule shifts with resolution (Hoogeboom et al., 2023; 2025), then hierarchical transformers (Crowson et al., 2024; Chen et al., 2025). The most recent line drops the hierarchy altogether: JiT (Li & He, 2026) and PixelDiT (Yu et al., 2026) denoise large patches with a plain transformer and a narrow patch embedding, matching latent-model quality at the same sequence length. What this buys in simplicity it concentrates in a single uniform patch grid whose size alone fixes the cost of every step, and in one known weakness, decoding a large patch, which PixelDiT answers with a second pixel-level stage and PixNerd with a per-patch neural field

[1] We note that RTI is a concurrent unpublished work and we compare directly with RTI in Section 4.

(Wang et al., 2026a). We adapt a released model of exactly this kind (AdaLN conditioning (Peebles & Xie, 2023), flow matching (Lipman et al., 2023), classifier-free guidance (Ho & Salimans, 2021)) rather than proposing another. Our refiner acts inside one coarse token, not as a stage over the image.

Content-adaptive tokenization. Unequal tokens for unequal regions is settled in discriminative vision, arriving first as reduction inside a pretrained transformer (Bolya et al., 2023; Rao et al., 2021; Liang et al., 2022; Yin et al., 2022; Wang et al., 2021) and then as tokenization proper, through flexible patch sizes and content-following tokenizers (Beyer et al., 2023; Dehghani et al., 2023; Lew et al., 2024; Ronen et al., 2023; Havtorn et al., 2023; Kim et al., 2025); our layout rule is the training-free entropy criterion of Choudhury et al. (2026). Adapting such strategies remains difficult for generation, because a denoiser must write an image *through* the layout and no image exists when the layout is chosen. Adaptive work here has therefore stayed off the input grid, varying what is updated or cached along the trajectory (Liu et al., 2026; Bolya & Hoffman, 2023; Wu et al., 2025; Zou et al., 2025), varying patch size uniformly over the image (Anagnostidis et al., 2025; Li et al., 2026), or retraining a latent tokenizer (Shen et al., 2026). Concurrent with us, RTI (Zamfir et al., 2026) fits a frozen pixel-space model with LoRA to consume region tokens. Our method, BudgetPix, instead fine-tunes the whole checkpoint to accept a varying-size input from the first block on, and reads the layout from the model's own estimate of the clean image rather than from its features.

3 BudgetPix

3.1 Overview

BudgetPix is designed to transform any existing pixel-space diffusion transformer into a compute-adaptive image generator. As illustrated in Figure 3, our approach consists of three core components: an encoder that compresses images into variable-length, multi-scale token sequences; a complementary decoder that reconstructs these token sequences back into fixed-resolution images; and a unified training and inference strategy. By replacing the uniform tokenization of standard baselines with this content-adaptive pipeline, BudgetPix enables existing diffusion backbones to seamlessly process flexible token counts and adjust their computational budget based on visual complexity. BudgetPix first constructs a content-adaptive image layout (Section 3.2). The encoder then uses this layout to map the image into a sequence of multi-scale tokens (Section 3.3), which are processed by the transformer and reconstructed into pixel space by the decoder (Section 3.4).

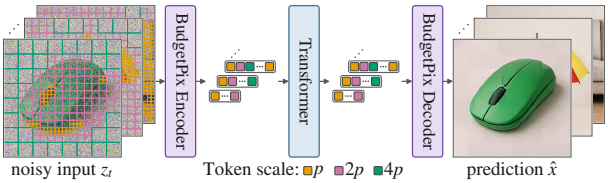

Figure 3: **BudgetPix Overview.** BudgetPix enables compute-adaptive pixel-space diffusion via three components: (1) an encoder mapping images to variable-length, multi-scale token sequences; (2) a decoder reconstructing fixed-resolution images from these sequences; and (3) a flexible training and inference strategy for varying token counts. The BudgetPix Encoder and Decoder (purple) are trained from scratch, whereas the central Transformer (blue) is initialized from existing baselines.

3.2 Non-Uniform Layout Construction

The image layout determines the spatial partition of the image–specifically, which regions maintain the base $p \times p$ patch resolution and which are merged into larger patches to be encoded as a single token. Because visual complexity is spatially non-uniform, fine details concentrate on specific regions while homogeneous areas (*e.g.*, sky or a plain wall) contain minimal information. Inspired by APT (Choudhury et al., 2026), we construct this non-uniform layout based on local intensity entropy. The layout is computed from the image during both training and inference. During training, it is derived directly from the ground-truth image x. At inference, because the clean image is unavailable, the model initially runs a few warm-up denoising steps on a uniform $p \times p$ grid to generate a predicted image $\hat{x}$, which is then used to estimate the content-adaptive layout.

Given N patch scales $s = 2^j p$, $j \in \{0, 1, \ldots, N-1\}$, the scale at $N-1$ represents the coarsest level, corresponding to the largest patch size. For each spatial cell R at this coarsest level, we compute a K-bin histogram of its internal pixel intensities and calculate the corresponding Shannon entropy: $H(R) = -\sum_{k=1}^{K} q_k(R) \log_2 q_k(R)$, where $q_k(R)$ denotes the fraction of pixels falling

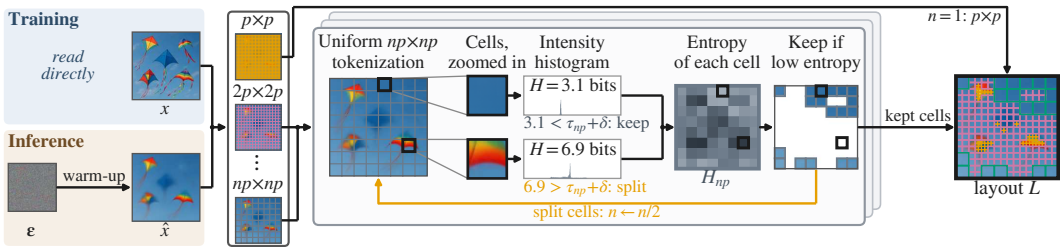

Figure 4: **Non-uniform layout construction.** Given an image (the original image x during training, or the denoising image $\hat{x}$ during inference), the objective is to partition it into cells across multiple scales: $p \times p$, $2p \times 2p, \ldots, np \times np$, where $n = 2^j$ for $j \geq 1$. The process begins by dividing the image into $np \times np$ cells, which is the coarsest scale. For each cell, we construct a pixel intensity histogram and compute its entropy H. If $H < \tau_{np} + \delta$ (where τ_{np} is the scale-specific threshold and δ is a global offset), the cell remains a single token. Otherwise, it is subdivided into four cells of the next finer scale ($n \leftarrow n/2$). This recursive subdivision terminates at $n = 1$, corresponding to the minimum cell size of $p \times p$.

into bin k. Token allocation proceeds top-down from the coarsest level: a cell is retained as a single token if its entropy satisfies $H(R) \leq \tau_s + \delta$, where τ_s is a scale-specific base threshold and $\delta \in \mathbb{R}$ is a global offset. If the entropy exceeds this threshold, the cell is split into its four quadrants, which are subsequently evaluated at the next finer scale. This recursive quadtree splitting continues until the finest scale ($p \times p$) is reached, ensuring the resulting multi-scale tokens perfectly tile the image (Figure 4).

3.3 BudgetPix Encoder: Embedding Non-Uniform Patches

The BudgetPix encoder receives non-uniform image patches defined by the entropy-guided quadtree and encodes them into a set of tokens with the same embedding size. Specifically, a patch P_i at scale $s = np$ (where $n = 2^j$ and $j \in \{0, 1, \ldots, N-1\}$) is mapped to a single token vector $\mathbf{x}_i$.

To process these variable-sized patches, the encoder utilizes a dual-path architecture (Figure 5, left). The *fine path* processes base patches ($n = 1$): each $p \times p$ patch is embedded directly via the pretrained patch embedder E inherited from the base model. The *coarse path* processes larger patches ($n > 1$). These patches are bilinearly downsampled to $p \times p$, denoted as $\mathrm{resize}_p(P_i)$, and subsequently embedded via E. While this main stream preserves macro-level semantics, it inherently discards high-frequency internal details. To recover this lost detail, a localized secondary stream, the *Patch Aggregator* (PA_s), is introduced. PA_s partitions P_i into n^2 sub-patches, denoted as $\mathrm{split}(P_i)$. Each sub-patch is independently embedded with E, augmented with a learned relative positional encoding $\mathbf{e}_{\mathrm{pos}}$ to specify its spatial location within P_i, and finally fused into a single correction vector using a scale-specific mixer. This scale-mixer comprises a stack of convolutional layers followed by an MLP layer (architectural details are provided in supplementary). Both paths add a learned scale embedding $\mathbf{e}_{\mathrm{scale}}(s)$, so the token is

$$\mathbf{x}_i = \begin{cases} E(P_i) + \mathbf{e}_{\mathrm{scale}}(s), & n = 1, \\ E\big(\mathrm{resize}_p(P_i)\big) + \mathbf{e}_{\mathrm{scale}}(s) + \mathrm{PA}_s\big(\mathrm{split}(P_i)\big), & n > 1. \end{cases} \tag{1}$$

The central diffusion transformer subsequently adds its own position embeddings and evaluates the tokens as usual. Note that the position embeddings applied within the Patch Aggregator and Patch Refiner operate in a localized, patch-relative coordinate system, whereas the embeddings utilized by the central diffusion transformer operate in a global, image-relative coordinate system.

3.4 BudgetPix Decoder: Decoding Multi-scale Image Patches

The BudgetPix decoder receives the output token sequence from the central transformer–where each token corresponds to a specific patch in the non-uniform spatial layout–and reconstructs them into pixel-space patches at their designated scales. These reconstructed patches subsequently tile together to form the final predicted image $\hat{x}$. Specifically, the decoder maps the latent token $\mathbf{h}_i$, corresponding to a patch P_i of scale $s = np$, back into an $np \times np$ pixel space patch $\hat{P}_i^{(s)}$.

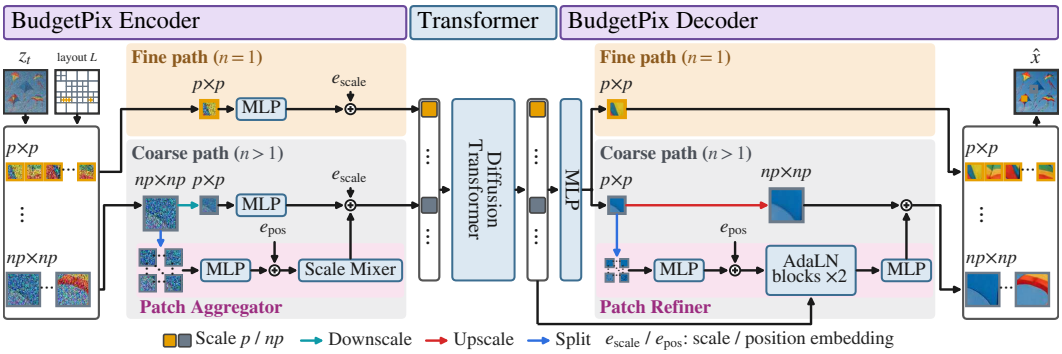

Figure 5: **BudgetPix Architecture.** The encoder receives the noisy input z_t and its corresponding spatial layout L, converts the image into a sequence of multi-scale patches. The finest $p \times p$ tokens route directly through the fine path, while coarser patches navigate a dedicated coarse path in both the encoder and decoder. In the encoder, each coarse patch undergoes simultaneous downscaling (to $p \times p$) and splitting via a Patch Aggregator (into n^2 sub-patches). These streams are processed through MLP layers and a scale mixer before being fused into a single coarse token. In the decoder, each denoised coarse patch is upscaled back to its target resolution ($np \times np$). Concurrently, a Patch Refiner splits the patch into 4×4 sub-patches, processes them through an MLP and a lightweight transformer, and fuses them with the upscaled patch to restore local texture. The central diffusion transformer is directly initialized from existing pixel-space models, modified only to operate across variable, dynamically determined token counts rather than a fixed count.

To reconstruct tokens of varying scale back into pixel space, the decoder (Figure 5, right) mirrors the dual-path architecture of the encoder. The *fine path* processes base tokens ($n = 1$): a pretrained decoding head maps the latent token $\mathbf{h}_i$ directly into a $p \times p$ pixel patch $\hat{P}_i \in \mathbb{R}^{p \times p \times C}$, where $C = 3$ represents the RGB image channels. The *coarse path* handles larger tokens ($n > 1$). Initially, the main stream decodes the token and bilinearly upsamples the output $\hat{P}_i$ to the target resolution of $np \times np$, denoted as $\text{resize}_{np}(\hat{P}_i)$. While this main stream successfully recovers the region's low-frequency structural layout, the resulting patch is inherently over-smoothed, lacking internal high-frequency textures. To resolve this, a localized secondary stream–the *Patch Refiner* (PR_s)–generates and injects the missing high-frequency details as an additive residual:

$$\hat{P}_i^{(s)} = \text{resize}_{np}\big(\hat{P}_i\big) + \text{PR}_s\big(\text{split}(\hat{P}_i), \mathbf{h}_i\big), \tag{2}$$

where $\hat{P}_i$ is the base patch from the pretrained head; on the fine path the resize is the identity and the residual is absent. The refiner is a lightweight transformer that stays inside one coarse token: it embeds 4×4 sub-patches $\text{split}(\hat{P}_i)$, adds local positional embeddings $\mathbf{e}_{\text{pos}}$, processes the sequence with two self-attention layers under adaptive layer normalization (AdaLN) conditioned on $\mathbf{h}_i$, which brings the token's global context into the local details, and projects the result linearly to the $np \times np$ residual. Because it runs on short, local token sequences, its cost per token does not grow with image resolution, and it adds only $\sim$1% to the parameter count.

3.5 Training and Inference

Training Objective. Any pretrained pixel-space diffusion transformer can be integrated into the BudgetPix framework by encapsulating it with our trainable encoder and decoder modules. The entire pipeline is then fine-tuned utilizing the base model's flow-matching objective for full-image x-prediction. Let the noisy latent state be $z_t = t\,x + (1 - t)\,\boldsymbol{\epsilon}$, and the network prediction be $\hat{x} = f_\theta(z_t, t, y, L)$, where t is the timestep, y is the conditioning signal, and L is the spatial layout. The fine-tuning loss is formalized as: $\mathcal{L}(\theta) = \mathbb{E}_{x,y,L,t,\boldsymbol{\epsilon}}\big[w(t)\|x - \hat{x}\|_2^2\big]$ where $w(t)$ represents the noise-schedule weighting derived from the base model.

Layout Augmentation. To ensure the transformer can reliably process sequences of varying lengths, we dynamically assign one of three distinct layout types to each image during training. Let $Q(\cdot, \tau + \delta)$ represent the entropy quadtree defined by thresholds $\tau_s + \delta$, and let $H(x)$ denote the true entropy map of the training image x. ***Uniform layout*** (probability $a < 1$): The model uses L_{uniform}, the standard dense grid of the base architecture. This anchors the training process and prevents catastrophic drift from the pretrained weights. ***Random layout*** (probability b, where

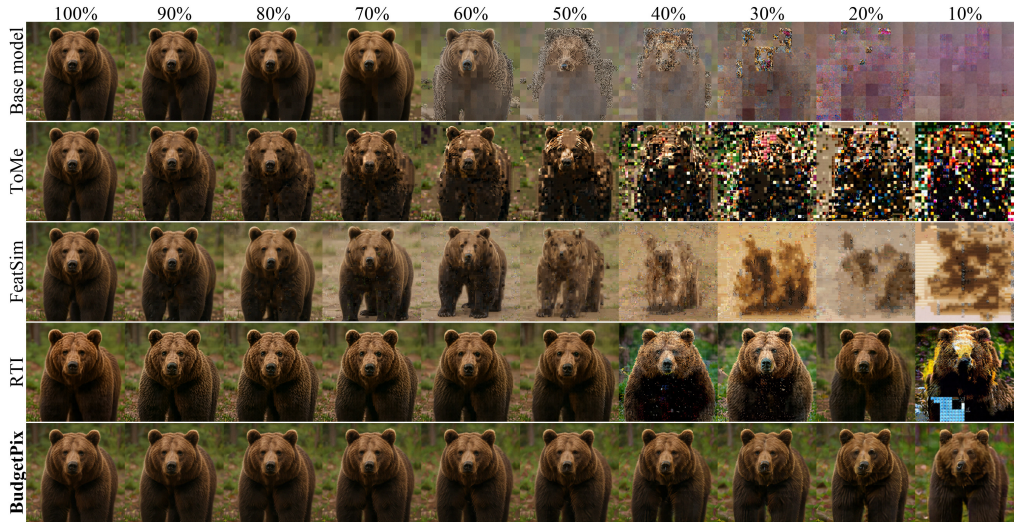

Figure 6: **Qualitative comparison across budgets, MiniT2I-L/16 at 512^2.** Same prompt and noise were used for all methods. BudgetPix generates image with high quality even at 10%.

$a + b < 1$): The quadtree $Q(\tilde{H}, \tau + \delta)$ is applied to a synthetic entropy map $\tilde{H}$, where values are drawn uniformly between 0 and $\log_2 K$ bits per cell. Crucially, this forces coarse tokens to randomly encompass highly detailed image regions. This simulates the constraints of low-budget inference, preventing the model from assuming that coarse tokens only represent flat backgrounds. ***Content-adaptive layout***(probability $1 - a - b$): The standard quadtree $Q(H(x), \tau + \delta)$ is applied using the actual image entropy map, optimally routing tokens based on true visual complexity. For both the random and content-adaptive quadtree layouts, the global offset δ is randomly sampled for each training image. This ensures the model learns to operate across a continuous range of token counts, scaling from roughly 25% up to 100% of the original dense grid.

Sampling with Dynamic Layout Estimation. During inference, the image's entropy is unknown before it is generated, so the layout is estimated as the image forms. Let i index the sampling steps and $\hat{x}_i$ be the model's prediction of the clean image at step i. The first n_w warm-up steps run on the uniform grid, $L_i = L_{\text{uniform}}$, which lets the model set the global composition of the image. After that, the layout is the entropy quadtree of the current prediction, $L_i = Q\big(H(\hat{x}_{i'}), \tau + \delta^*(\hat{x}_{i'}, B)\big)$, with the thresholds shifted by the offset δ^* that budget mode finds for this prediction and the budget B. The layout is rebuilt only every m steps: $i' = n_w + m\lfloor (i - n_w)/m \rfloor$ is the last rebuild step at or before i, so the m steps from i' on share one layout.

4 Experiments

4.1 Implementation Details

We adapt seven released pixel-space diffusion transformers: JiT-B/16 and JiT-L/16 at 256^2 and JiT-B/32 and JiT-L/32 at 512^2 (Li & He, 2026), MiniT2I-B/16 and MiniT2I-L/16 (Wang et al., 2026b) at 512^2, and PixelDiT-T2I (Yu et al., 2026) at 1024^2, plus the one-step pixel MeanFlow pMF-L/16 (Lu et al., 2026) at 256^2. Each is fine-tuned once with the layout augmentation (with $a = 0.15$, $b = 0.1$) using the objective described in Section 3.5 for different budget. The added modules are zero-initialised, so training starts exactly from the released model (see App. C.2–C.4).

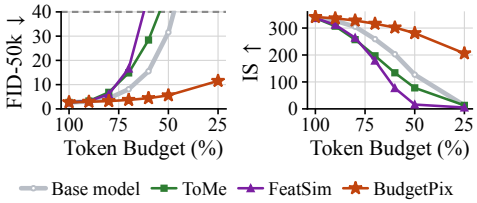

Figure 7: **Class-conditional generation on ImageNet, JiT-L/32 at 512^2.** BudgetPix provides a better trade-off across different compute-budgets. IS denotes Inception Score, FID is cut at 40.

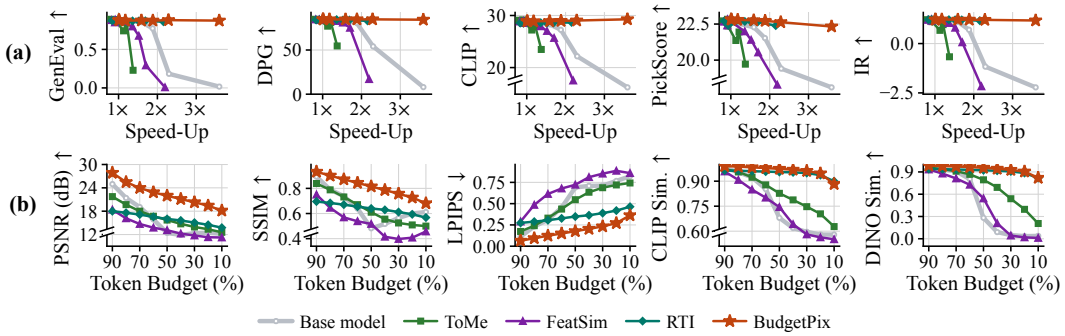

Figure 8: **Text-to-image results, MiniT2I-L/16 at 512^2. (a)** Each metric plotted against the wall-clock speed-up over the released model (DPG: DPG-Bench; IR: ImageReward denote metrics). **(b)** Faithfulness to the method's own full-budget image plotted against the token budget (Sim.: CLIP or DINOv2 feature similarity).

4.2 Evaluation settings

Baselines. We compare BudgetPix against the released checkpoints driven to the same budget through our sampler and three token-reduction methods on those weights: ToMe (Bolya et al., 2023), the feature-similarity merge of ToMeSD (Bolya & Hoffman, 2023) applied once, down to the budget (*FeatSim*), and RTI (Zamfir et al., 2026), a concurrent adapter with region tokens, run from its released checkpoint. Every method shares prompts or labels, initial noise, solver, step count and attention arithmetic; only the budget and the reduction rule differ. See App. C.5 for the protocol.

Evaluation dataset and metrics. We use FID-50k (Heusel et al., 2017) and Inception score (Salimans et al., 2016) for class-conditional quality, GenEval (Ghosh et al., 2023) and DPG-Bench (Hu et al., 2024) for prompt adherence, and CLIP (Radford et al., 2021), PickScore (Kirstain et al., 2023) and ImageReward (Xu et al., 2023) for preference on PartiPrompts (Yu et al., 2022). We also measure faithfulness, whether a reduced-budget output is still *the same image*, with paired PSNR, SSIM (Wang et al., 2004), LPIPS (Zhang et al., 2018) and CLIP and DINOv2 (Oquab et al., 2024) similarity against each method's own output at full 100% budget. Cost is wall-clock time; see App. C.7 for the timing.

4.3 Analysis

Class-conditional generation at 256^2 and 512^2. On JiT-L/32 at 512^2 (Figure 7), the FID of ours starts at that of the released model and rises only slowly as the budget falls, while the base model and both merging baselines degrade sharply: at half 60-50% budgets their FID is several times ours, and ToMe and FeatSim do worse than simply running the released weights at the reduced budget. The Inception score shows the same ordering: ours keeps most of its dense value even at a 60% budget, where the others collapse. Additional quantitative and qualitative results for JiT-L/16 at 256^2 are provided in App. B.1 and A.2.

Faithfulness across the budget. We measure faithfulness, that is, how much a method's image at a reduced budget similar to the image the same method produces at their full budget. In Figure 8(b), ours stays closest to its dense image on nearly every metric and budget, while the base model and both merging baselines lose the image by half the grid. These gaps can be attributed to the fact that our layout merges the flattest regions first and the refiner restores their texture, whereas the merging baselines group tokens by feature similarity across the whole frame and the base model receives coarse tokens it was never trained on. See App. B.2 for the base model and ours on JiT.

Table 1: **Text-to-image generation** with PixelDiT-T2I at 1024^2. BudgetPix consistently outperform base model across different compute budgets.

metric	method	100%	90%	80%	70%	60%	50%	25%
GenEval ↑	Base model	0.721	0.722	0.724	0.725	0.708	0.674	0.323
	BudgetPix	**0.741**	**0.740**	**0.741**	**0.738**	**0.746**	**0.743**	**0.725**
DPG-Bench ↑	Base model	84.8	84.6	84.2	84.0	83.3	81.8	55.1
	BudgetPix	**85.2**	**85.2**	**85.0**	**85.0**	**84.6**	**84.6**	**83.7**

Text-to-image generation at 512^2 and 1024^2. On MiniT2I-L (Figure 8(a); values in Table 5) GenEval and DPG-Bench stay flat across the compute budgets while the base model collapses to near

zero, and the three preference metrics follow suit; PixelDiT at 1024^2 is similar, if milder (Table 1). Figure 6 shows one subject under every method at every budget; see App. A.1 for six more, and App. A.2–A.8 for ours across the compute budgets.

User study and VLM-as-Judge. The paired metrics say how far an image moved, not when a reader would notice. In a user study (Figure 9), 50 people per form compare images on 20 GenEval prompts, and in VLM-as-Judge a vision–language model does the same on 100. Shown a method's own dense image and a grid of its budgets, the budget can be cut by 52 % for ours before human raters first see a difference, against at most 38 % for the baselines, and by 36 % against at most 21 % before the VLM judge does (Figure 9 a). When shown the full-budget image and two budget-reduced candidates from two different methods, human raters find ours more faithful than RTI at all four budgets asked, and the VLM judge also favors ours than every baseline below 80 % (Figure 9 b). See App. C.6 for both protocols.

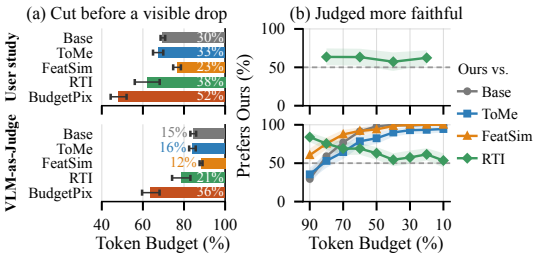

Figure 9: **User study (top) and VLM-as-Judge (bottom), MiniT2I-L/16** at 512^2. Left: How many percentage from the compute budget can be cut without a noticeable difference for human and VLM judges. Right: Head-to-head comparison of BudgetPix against baselines by human and VLM judges.

Wall-clock and speed-up. Figure 8(a) plots every text-to-image metric against the measured wall-clock speed-up over the released model: ours holds its scores across the whole range, the base model starts at the same value but collapses fast. The merging baselines buy less speed, since the merge itself costs time. See App. B.4 for the other base models.

Ablation study. Table 2 switches the two added modules on in turn. The aggregator does most of the work, since it is what lets a coarse token carry its region's content. The refiner takes roughly a further third off, and its contribution grows as the budget falls, since it acts only inside coarse regions. Neither costs anything at the dense grid, where both are inert by construction. See App. D for other ablations and App. D.6 for the entropy quadtree against random and oracle layouts.

Table 2: **Ablation on JiT-B, FID-50k.** BudgetPix with both Patch Aggregator and Refiner works best across different compute budgets.

	Model	100%	90%	80%	70%	60%	50%	25%
512^2	Base Model	4.12	5.00	6.86	12.24	24.79	60.61	190.71
	+ Patch Aggregator	3.96	4.52	5.26	6.67	8.56	11.58	25.07
	+ Patch Refiner	**3.94**	**4.35**	**4.80**	**5.57**	**6.59**	**8.30**	**16.71**
256^2	Base Model	3.61	4.24	5.84	10.67	23.46	63.94	145.46
	+ Patch Aggregator	**3.57**	3.92	4.34	5.14	6.17	7.84	16.90
	+ Patch Refiner	3.60	**3.89**	**4.25**	**4.89**	**5.73**	**7.01**	**13.05**

One-step generation with MeanFlow. Ours cuts the cost of each step, not their number, so it complements other inference-efficiency methods such as distillation. Figure 10 shows our results on one-step (1-NFE) pixel MeanFlow (pMF-L/16; Lu et al., 2026), fine-tuned on layouts from training images of each class: the released weights collapse from 80 %; ours stays below 4 FID to half the grid (see App. B.6 and A.5).

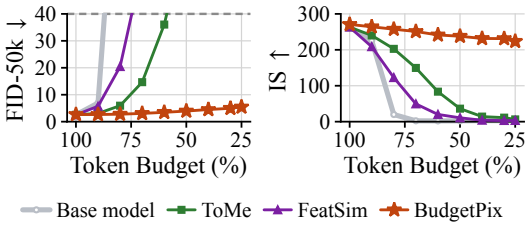

Figure 10: **BudgetPix on one-step pixel MeanFlow pMF-L/16.** BudgetPix provides a great quality-efficiency trade-off. FID cut at 40.

5 Conclusion

BudgetPix shows that the token count of a released pixel-space diffusion transformer does not have to be fixed at pretraining: after one fine-tuning run, a single checkpoint serves budgets from the dense grid to a small fraction of it and degrades gracefully where token-merging baselines collapse, even at 1 NFE (Figure 10). Its limitations are that the layout is derived from the model's intermediate predictions, so it can miss detail they do not yet show, and that full-resolution stages, such as PixelDiT's pixel stage, cap the speed-up. Learning where to place tokens, not deriving it from entropy, is a natural next step.

AI USE STATEMENT

We used AI assistants based on large language models to polish the text and its grammar and to help with coding. The VLM-as-Judge study uses a vision–language model for evaluation (see App. C.6). The authors have verified all AI-produced output and take full responsibility for the content of this paper.

ETHICS STATEMENT

This work lowers the inference cost of pixel-space diffusion transformers that are already public. It adds no new generation capability, but cheaper generation also makes misuse, such as misleading or harmful images, cheaper; the licences and safeguards of the released models apply to their adapted versions. We use only public checkpoints, datasets and benchmarks (ImageNet, CC12M, PD12M, the 120K alignment mix, GenEval, DPG-Bench and PartiPrompts) under their licences. The user study collected anonymous answers through online forms, with no names, e-mail addresses or sign-in, and showed only generated images of GenEval prompts.

REPRODUCIBILITY STATEMENT

All parents are public checkpoints (Section 4.1). See App. C for what is needed to rerun our experiments: the training objective, the zero-initialisation check, every fine-tuning and sampling setting (Tables 18 and 19), the evaluation protocol, the user study and VLM-as-Judge protocols with the forms and the judge's prompts, and the wall-clock measurement conditions. See App. B for every value behind the plots and App. D for the ablations. We will release the code and the fine-tuned checkpoints.

REFERENCES

Sotiris Anagnostidis, Gregor Bachmann, Yeongmin Kim, Jonas Kohler, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Albert Pumarola, Ali Thabet, and Edgar Schönfeld. Flexidit: Your diffusion transformer can easily generate high-quality samples with less compute. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 28316–28326. IEEE, 2025.

Lucas Beyer, Pavel Izmailov, Alexander Kolesnikov, Mathilde Caron, Simon Kornblith, Xiaohua Zhai, Matthias Minderer, Michael Tschannen, Ibrahim Alabdulmohsin, and Filip Pavetic. Flexivit: One model for all patch sizes. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14496–14506. IEEE, 2023.

Daniel Bolya and Judy Hoffman. Token merging for fast stable diffusion. In *2023 IEEE/CVF conference on computer vision and pattern recognition workshops (CVPRW)*, pp. 4599–4603. IEEE, 2023.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=JroZRaRw7Eu.

Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12m: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3557–3567. IEEE, 2021.

Shoufa Chen, Chongjian Ge, Shilong Zhang, Peize Sun, and Ping Luo. Pixelflow: Pixel-space generative models with flow. *arXiv preprint arXiv:2504.07963*, 2025.

Rohan Choudhury, JungEun Kim, Jinhyung Park, Eunho Yang, Laszlo A. Jeni, and Kris Kitani. Faster vision transformers with adaptive patches. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=SzoowJtd14.

Katherine Crowson, Stefan Andreas Baumann, Alex Birch, Tanishq Mathew Abraham, Daniel Z Kaplan, and Enrico Shippole. Scalable high-resolution pixel-space image synthesis with hourglass diffusion transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 9550–9575. PMLR, 21–27 Jul 2024. URL https://proceedings.mlr.press/v235/crowson24a.html.

Mostafa Dehghani, Basil Mustafa, Josip Djolonga, Jonathan Heek, Matthias Minderer, Mathilde Caron, Andreas Steiner, Joan Puigcerver, Robert Geirhos, Ibrahim M Alabdulmohsin, et al. Patch n'pack: Navit, a vision transformer for any aspect ratio and resolution. *Advances in Neural Information Processing Systems*, 36:2252–2274, 2023.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *Advances in Neural Information Processing Systems*, 38:75460–75482, 2026.

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.

Jakob Drachmann Havtorn, Amélie Royer, Tijmen Blankevoort, and Babak Ehteshami Bejnordi. Msvit: Dynamic mixed-scale tokenization for vision transformers. In *2023 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pp. 838–848. IEEE, 2023.

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30, 2017.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*, 2021. URL https://openreview.net/forum?id=qw8AKxfYbI.

Jonathan Ho, Chitwan Saharia, William Chan, David J Fleet, Mohammad Norouzi, and Tim Salimans. Cascaded diffusion models for high fidelity image generation. *Journal of Machine Learning Research*, 23(47):1–33, 2022.

Emiel Hoogeboom, Jonathan Heek, and Tim Salimans. simple diffusion: End-to-end diffusion for high resolution images. In *International Conference on Machine Learning*, pp. 13213–13232. PMLR, 2023.

Emiel Hoogeboom, Thomas Mensink, Jonathan Heek, Kay Lamerigts, Ruiqi Gao, and Tim Salimans. Simpler diffusion: 1.5 fid on imagenet512 with pixel-space diffusion. In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 18062–18071. IEEE, 2025.

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*, 2024.

Minchul Kim, Dingqiang Ye, Yiyang Su, Feng Liu, and Xiaoming Liu. Sapiensid: Foundation for human recognition. In *CVPR*, 2025.

Yuval Kirstain, Adam Polyak, Uriel Singer, Shahbuland Matiana, Joe Penna, and Omer Levy. Pick-a-pic: An open dataset of user preferences for text-to-image generation. *Advances in neural information processing systems*, 36:36652–36663, 2023.

Jaihyun Lew, Soohyuk Jang, Jaehoon Lee, Seungryong Yoo, Eunji Kim, Saehyung Lee, Jisoo Mok, Siwon Kim, and Sungroh Yoon. Superpixel tokenization for vision transformers: Preserving semantic integrity in visual tokens. *arXiv preprint arXiv:2412.04680*, 2024.

Hui Li, Baoyou Chen, Jiaye Li, Jingdong Wang, and Siyu Zhu. Pyramid patchification flow for visual generation. In *International Conference on Learning Representations*, volume 2026, pp. 11656–11676, 2026.

Tianhong Li and Kaiming He. Back to basics: Let denoising generative models denoise. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 36115–36125, 2026.

Youwei Liang, Chongjian GE, Zhan Tong, Yibing Song, Jue Wang, and Pengtao Xie. EVit: Expediting vision transformers via token reorganizations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=BjyvwnXXVn_.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.

Ziming Liu, Yifan Yang, Chengruidong Zhang, Yiqi Zhang, Lili Qiu, Yang You, and Yuqing Yang. Region-adaptive sampling for diffusion transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 2346–2356, 2026.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. Dpm-solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025.

Yiyang Lu, Susie Lu, Qiao Sun, Hanhong Zhao, Zhicheng Jiang, Xianbang Wang, Tianhong Li, Zhengyang Geng, and Kaiming He. One-step latent-free image generation with pixel mean flows. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=WUK8JIeetF.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research*, 2024. ISSN 2835-8856. URL https://openreview.net/forum?id=a68SUt6zFt. Featured Certification.

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

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*, pp. 8748–8763. PmLR, 2021.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. *Advances in neural information processing systems*, 34:13937–13949, 2021.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Tomer Ronen, Omer Levy, and Avram Golbert. Vision transformers with mixed-resolution tokenization. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 4613–4622. IEEE, 2023.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in neural information processing systems*, 29, 2016.

Junhong Shen, Kushal Tirumala, Michihiro Yasunaga, Ishan Misra, Luke Zettlemoyer, Lili Yu, and Chunting Zhou. Cat: Content-adaptive image tokenization. *Advances in Neural Information Processing Systems*, 38:712–742, 2026.

Shuai Wang, Ziteng Gao, Chenhui Zhu, Weilin Huang, and Limin Wang. Pixnerd: Pixel neural field diffusion. In *International Conference on Learning Representations*, volume 2026, pp. 43559–43580, 2026a.

Xianbang Wang, Hanhong Zhao, Yiyang Lu, Kangyang Zhou, Linrui Ma, and Kaiming He. Minit2i: A minimalist baseline for text-to-image generation, 2026b. URL https://peppaking8.github.io/#/post/minit2i.

Yulin Wang, Rui Huang, Shiji Song, Zeyi Huang, and Gao Huang. Not all images are worth 16x16 words: Dynamic transformers for efficient image recognition. *Advances in neural information processing systems*, 34:11960–11973, 2021.

Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612, 2004.

Haoyu Wu, Jingyi Xu, Hieu Le, and Dimitris Samaras. Importance-based token merging for efficient image and video generation. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4983–4995. IEEE, 2025.

Jiazheng Xu, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong. Imagereward: Learning and evaluating human preferences for text-to-image generation. *Advances in Neural Information Processing Systems*, 36:15903–15935, 2023.

Hongxu Yin, Arash Vahdat, Jose M Alvarez, Arun Mallya, Jan Kautz, and Pavlo Molchanov. A-vit: Adaptive tokens for efficient vision transformer. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10799–10808. IEEE, 2022.

Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, Ben Hutchinson, Wei Han, Zarana Parekh, Xin Li, Han Zhang, Jason Baldridge, and Yonghui Wu. Scaling autoregressive models for content-rich text-to-image generation. *Transactions on Machine Learning Research*, 2022. ISSN 2835-8856. URL https://openreview.net/forum?id=AFDcYJKhND. Featured Certification.

Yongsheng Yu, Wei Xiong, Weili Nie, Yichen Sheng, Shiqiu Liu, and Jiebo Luo. Pixeldit: Pixel diffusion transformers for image generation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 14273–14282, 2026.

Eduard Zamfir, Christian Reisswig, Zongwei Wu, Yongqin Xian, and Radu Timofte. Elastic token compression for pixel-space diffusion transformers. *arXiv preprint arXiv:2608.29281*, 2026.

Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *2018 IEEE/CVF conference on computer vision and pattern recognition*, pp. 586–595. IEEE, 2018.

Chang Zou, Xuyang Liu, Ting Liu, Siteng Huang, and Linfeng Zhang. Accelerating diffusion transformers with token-wise feature caching. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=yYZbZGo4ei.

SUPPLEMENTARY MATERIAL

CONTENTS OF THE SUPPLEMENTARY MATERIAL

A Additional qualitative results across the budget **16**

A.1 Baseline Comparison (MiniT2I-L/16, 512^2) . . . 16

A.2 BudgetPix (JiT-L/32, 512^2) . . . 19

A.3 BudgetPix (MiniT2I-L/16, 512^2) . . . 21

A.4 BudgetPix (PixelDiT-T2I, 1024^2) . . . 23

A.5 BudgetPix (pMF-L/16, 256^2, 1 NFE) . . . 25

A.6 BudgetPix layouts (JiT-L/32, 512^2) . . . 28

A.7 BudgetPix layouts (MiniT2I-L/16, 512^2) . . . 30

A.8 BudgetPix layouts (PixelDiT-T2I, 1024^2) . . . 32

B Results in full **34**

B.1 Class-conditional generation (JiT) . . . 34

B.2 Faithfulness to the dense image (JiT) . . . 35

B.3 Text-to-image generation (MiniT2I) . . . 36

B.4 Wall-clock speed-up (every parent) . . . 37

B.5 Budget mode against threshold mode (JiT-L/32) . . . 41

B.6 One-step generation (pixel MeanFlow) . . . 42

C Details of the method and the experiments **43**

C.1 Architecture of the added modules . . . 43

C.2 Training objective . . . 44

C.3 Zero-initialisation check . . . 45

C.4 Fine-tuning and sampling settings . . . 46

C.5 Evaluation protocol . . . 48

C.6 User study and VLM-as-Judge . . . 49

C.7 Wall-clock measurement . . . 52

D Ablations on JiT-B **54**

D.1 Reference runs (development JiT-B) . . . 54

D.2 A parallel pixel-stream decoder (development JiT-B/32) . . . 54

D.3 Sampling schedule and step count (development JiT-B/32) . . . 54

D.4 Guidance scale along the budget (development JiT-B/32) . . . 54

D.5 Upper bound on a better decoder (development JiT-B/32) . . . 55

D.6 Layout and sampler settings (final JiT-B/32) . . . 55

D.7 Sensitivity to the entropy thresholds (final JiT-B/32) . . . 56

D.8 Layout evolution during sampling (final JiT-B/32) . . . 56

D.9 Pooled budget across a batch (final JiT-B/32) . . . 56

A ADDITIONAL QUALITATIVE RESULTS ACROSS THE BUDGET

A.1 BASELINE COMPARISON (MINIT2I-L/16, 512^2)

Figures 11–16 show the comparison with the baselines on MiniT2I-L/16 at 512^2, one prompt per figure. Each row is a method (the base model, ToMe, FeatSim, RTI and ours), and each column a token budget, from 100 % to 10 %.

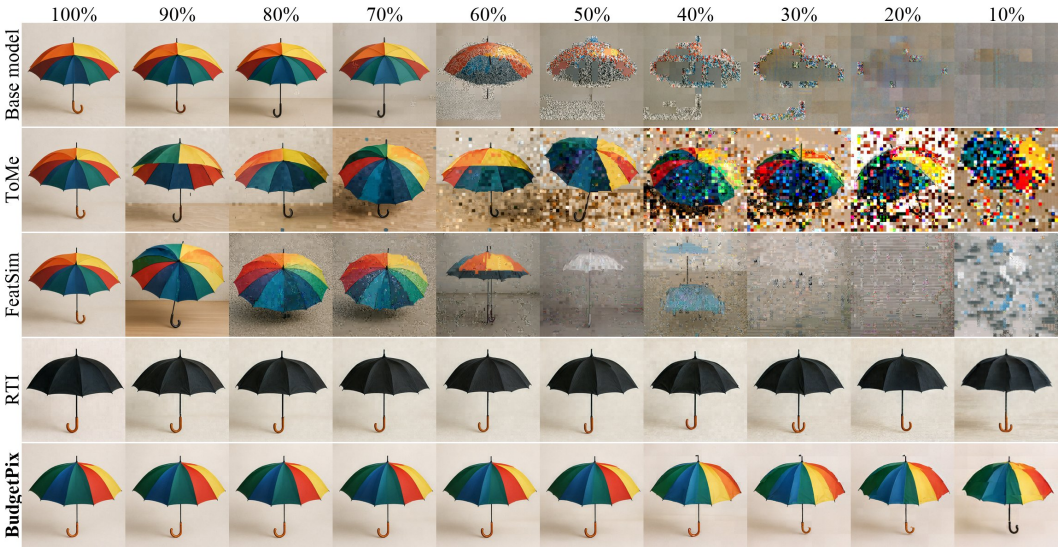

Figure 11: **Every method across the budget: an umbrella.**

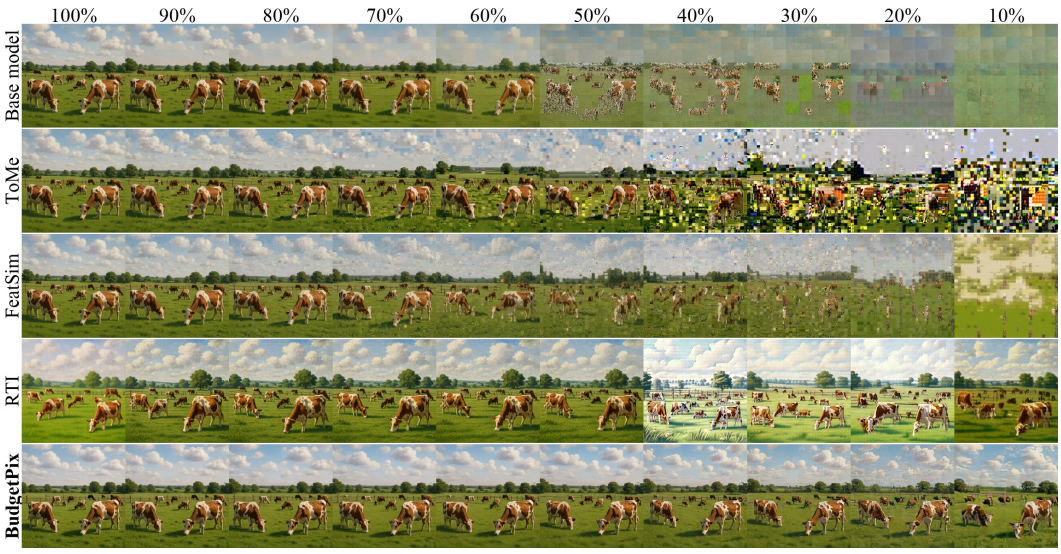

Figure 12: **Every method across the budget: a herd of cows.**

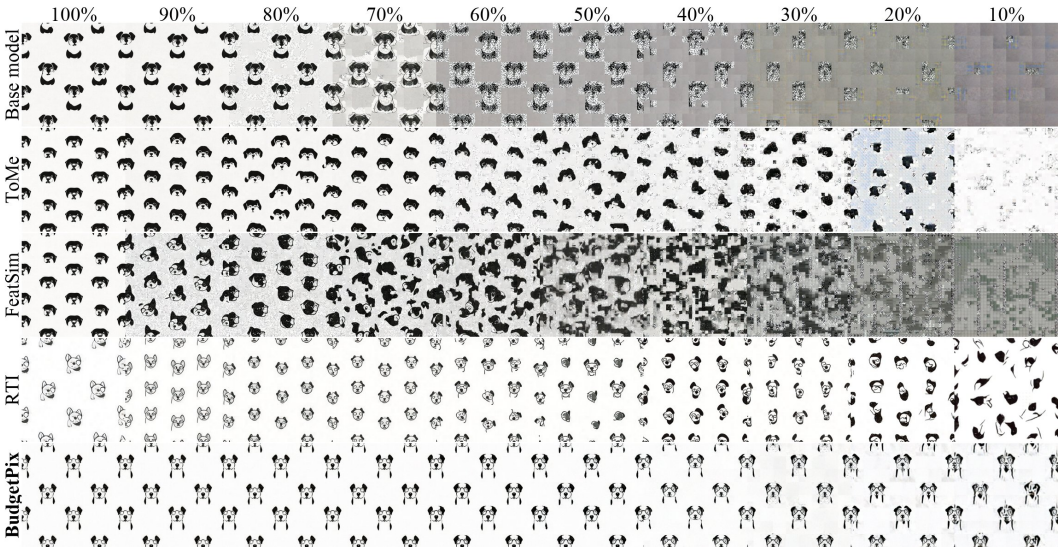

Figure 13: **Every method across the budget: a seamless vector pattern.**

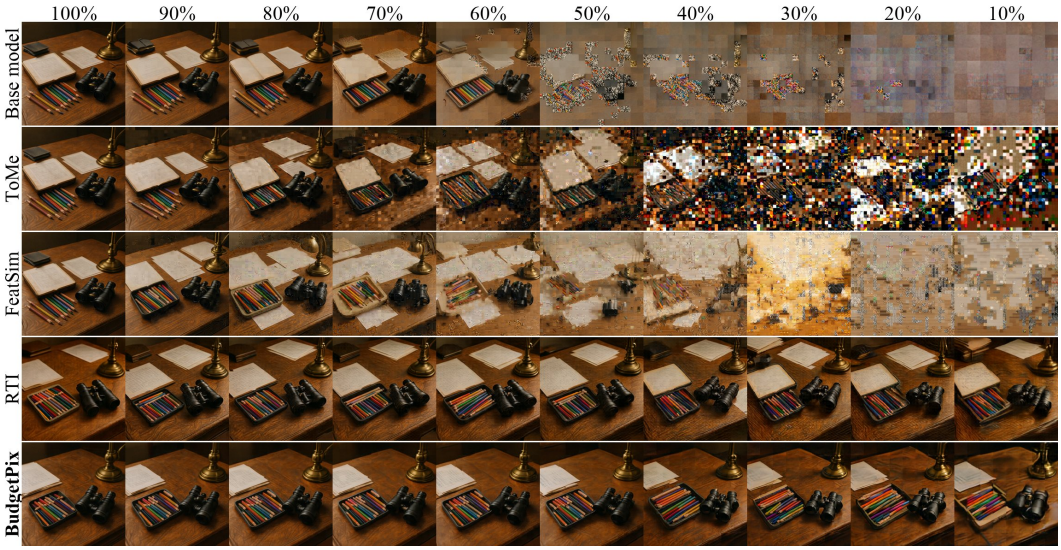

Figure 14: **Every method across the budget: a pencil case and binoculars.**

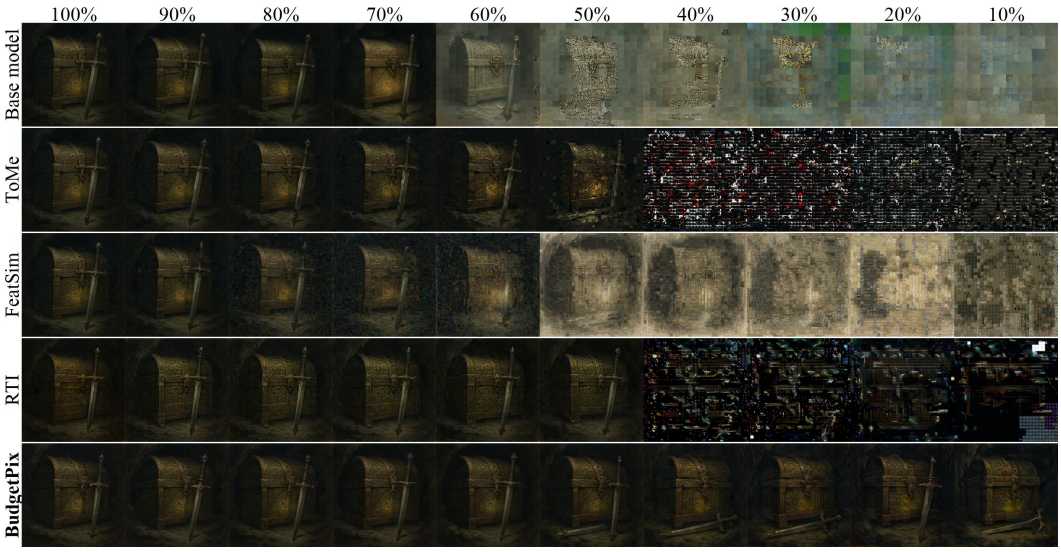

Figure 15: **Every method across the budget: an ornate treasure chest.**

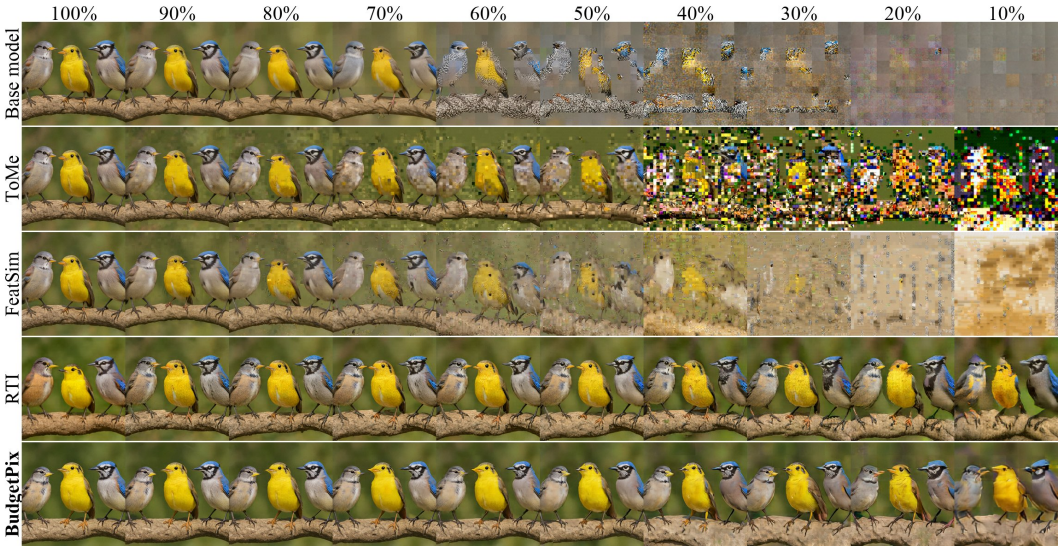

Figure 16: **Every method across the budget: three birds on a branch.**

A.2 BUDGETPIX (JIT-L/32, 512^2)

Figures 17 and 18 show additional qualitative results on JiT-L/32 at 512^2. Each row is one ImageNet class, and each column a token budget, from 100 % to 10 %.

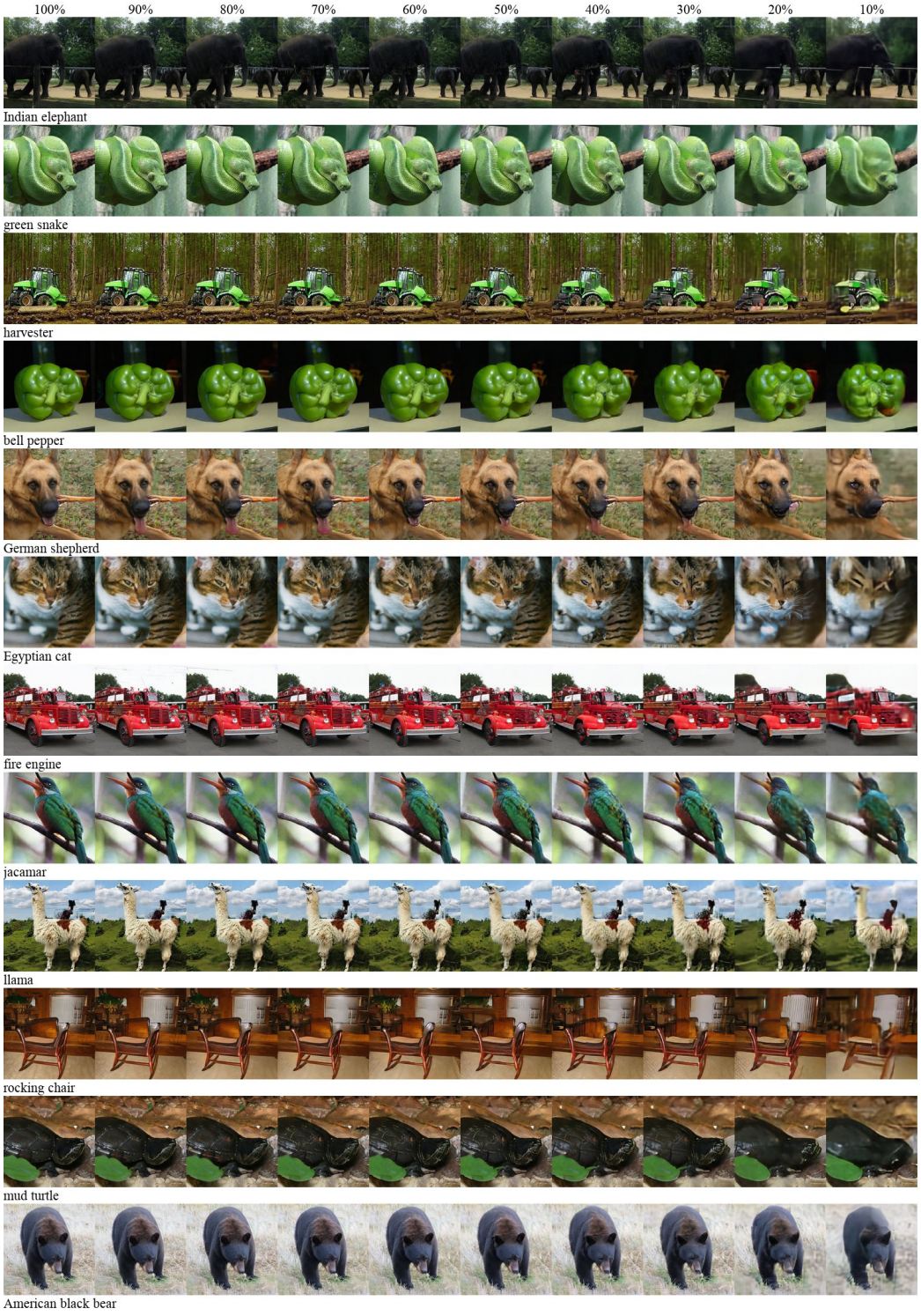

Figure 17: **BudgetPix on JiT-L/32 at 512^2, one ImageNet class per row.**

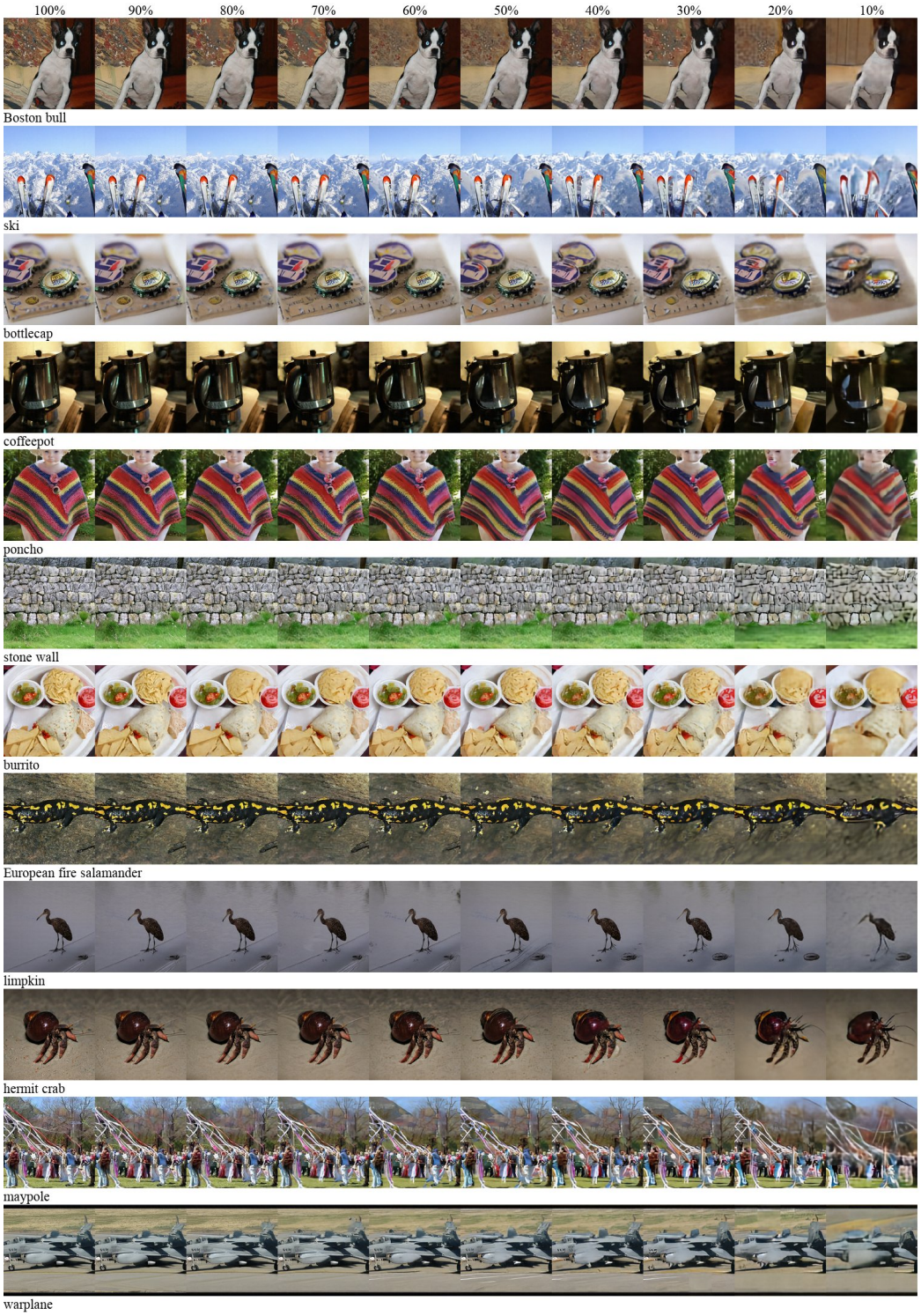

Figure 18: **BudgetPix on JiT-L/32 at 512^2, one ImageNet class per row (continued).**

A.3 BUDGETPIX (MINIT2I-L/16, 512^2)

Figures 19 and 20 show additional qualitative results on MiniT2I-L/16 at 512^2. Each row is one prompt, and each column a token budget, from 100 % to 10 %.

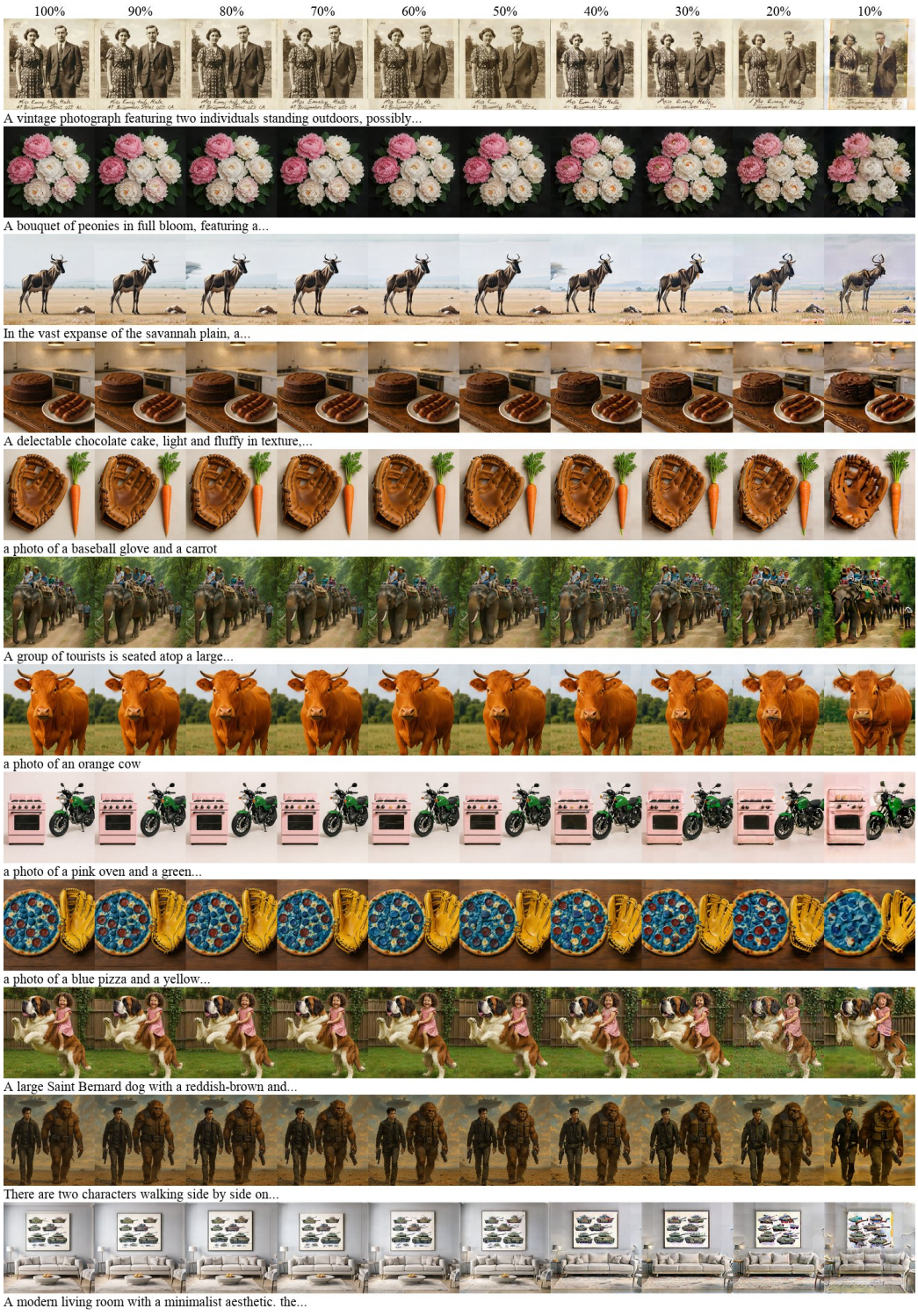

Figure 19: **BudgetPix on MiniT2I-L/16 at 512^2, one prompt per row.**

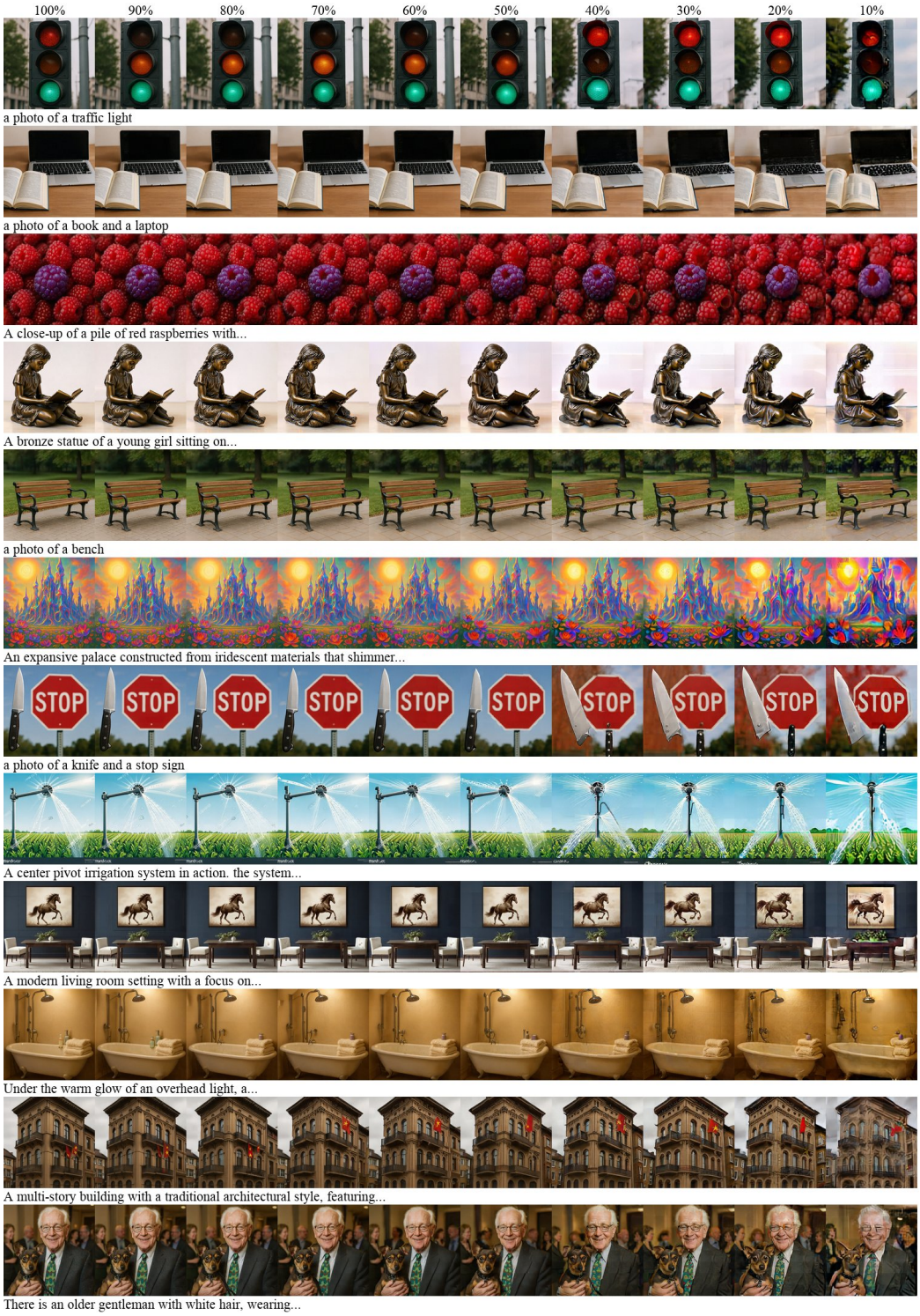

Figure 20: **BudgetPix on MiniT2I-L/16 at 512^2, one prompt per row (continued).**

A.4 BUDGETPIX (PIXELDIT-T2I, 1024^2)

Figures 21 and 22 show additional qualitative results on PixelDiT-T2I at 1024^2. Each row is one prompt, and each column a token budget, from 100 % to 10 %.



Figure 21: **BudgetPix on PixelDiT-T2I at 1024^2, one prompt per row.**



Figure 22: **BudgetPix on PixelDiT-T2I at 1024^2, one prompt per row (continued).**

A.5 BUDGETPIX (PMF-L/16, 256^2, 1 NFE)

Figures 23–25 show additional qualitative results on the one-step pMF-L/16 at 256^2. Each row is one ImageNet class, and each column a token budget, from 100 % to 25 %; Figure 23 first shows the released model at 100 % and the base model at 50 % (the released weights with our layouts, no training).



Figure 23: **One-step samples, pMF-L/16 at 256^2, one ImageNet class per row.**

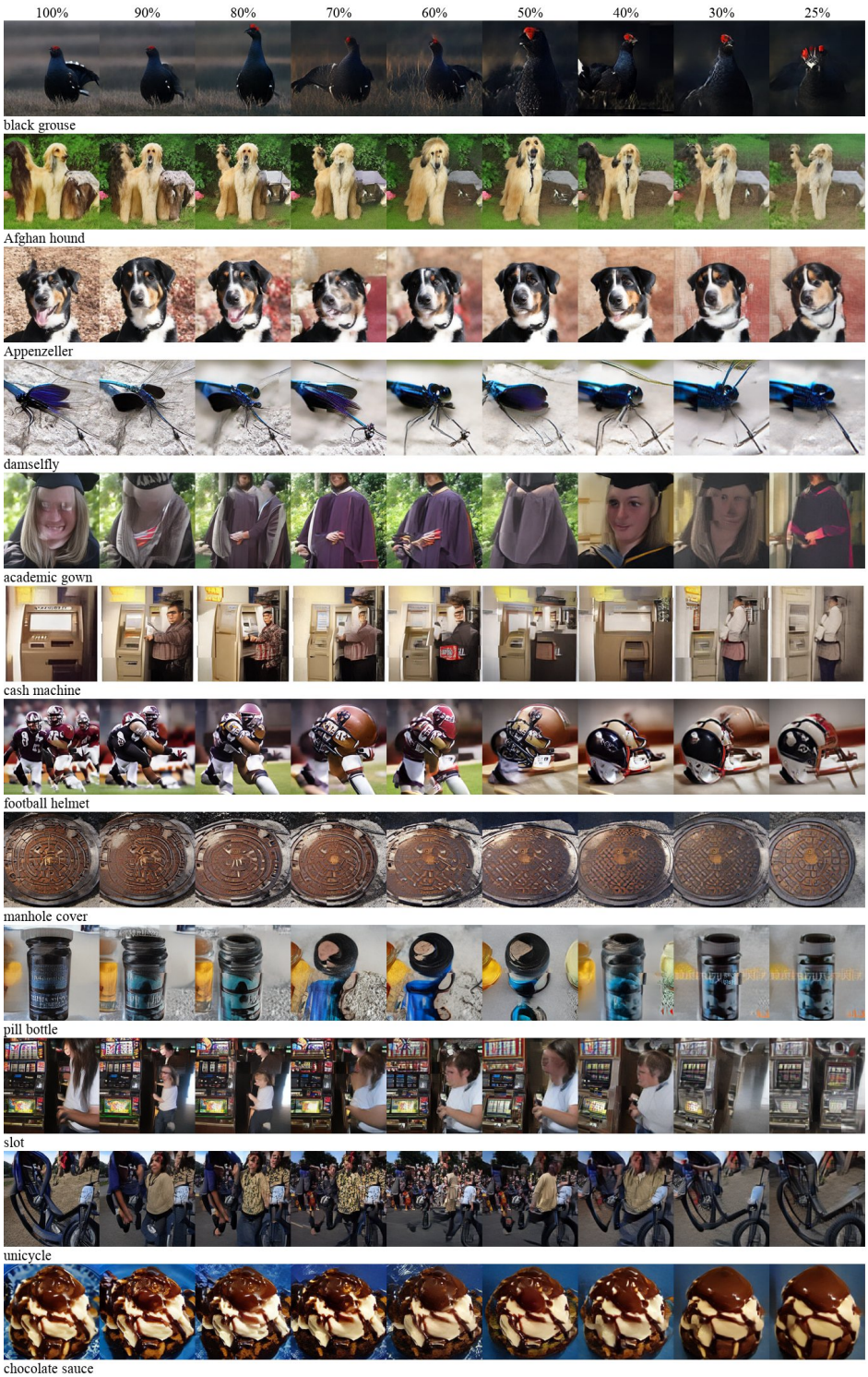

Figure 24: **BudgetPix on pMF-L/16 at 256^2, one step, one ImageNet class per row.**

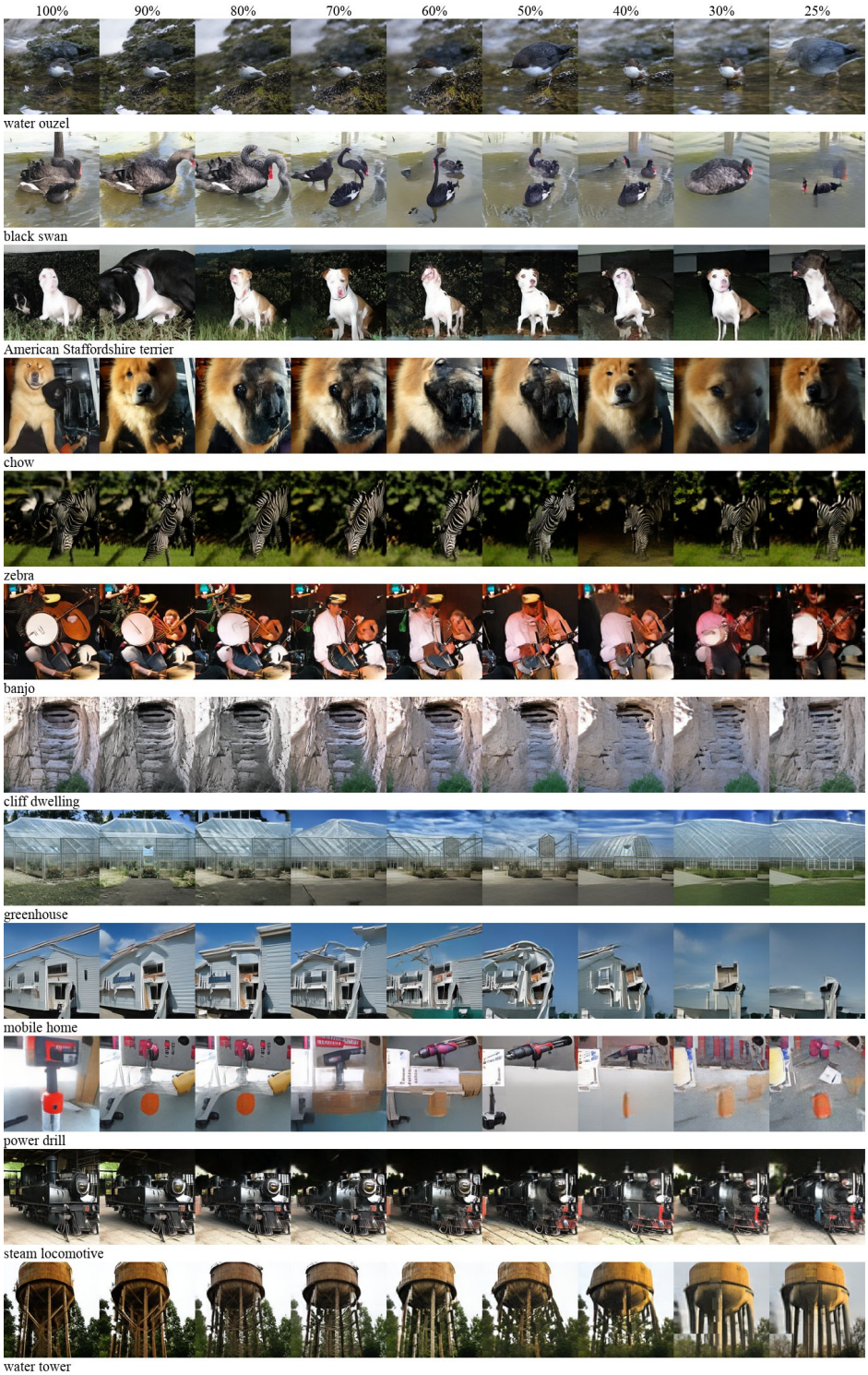

Figure 25: **BudgetPix on pMF-L/16 at 256^2, one step, one ImageNet class per row (continued).**

A.6 BUDGETPIX LAYOUTS (JIT-L/32, 512^2)

Figures 26 and 27 show additional qualitative results with their layouts on JiT-L/32 at 512^2. Each row is one ImageNet class; the columns are the samples at 100, 50 and 25 % of the grid, then the same images with their layouts drawn over them (each token outlined in the colour of its size).

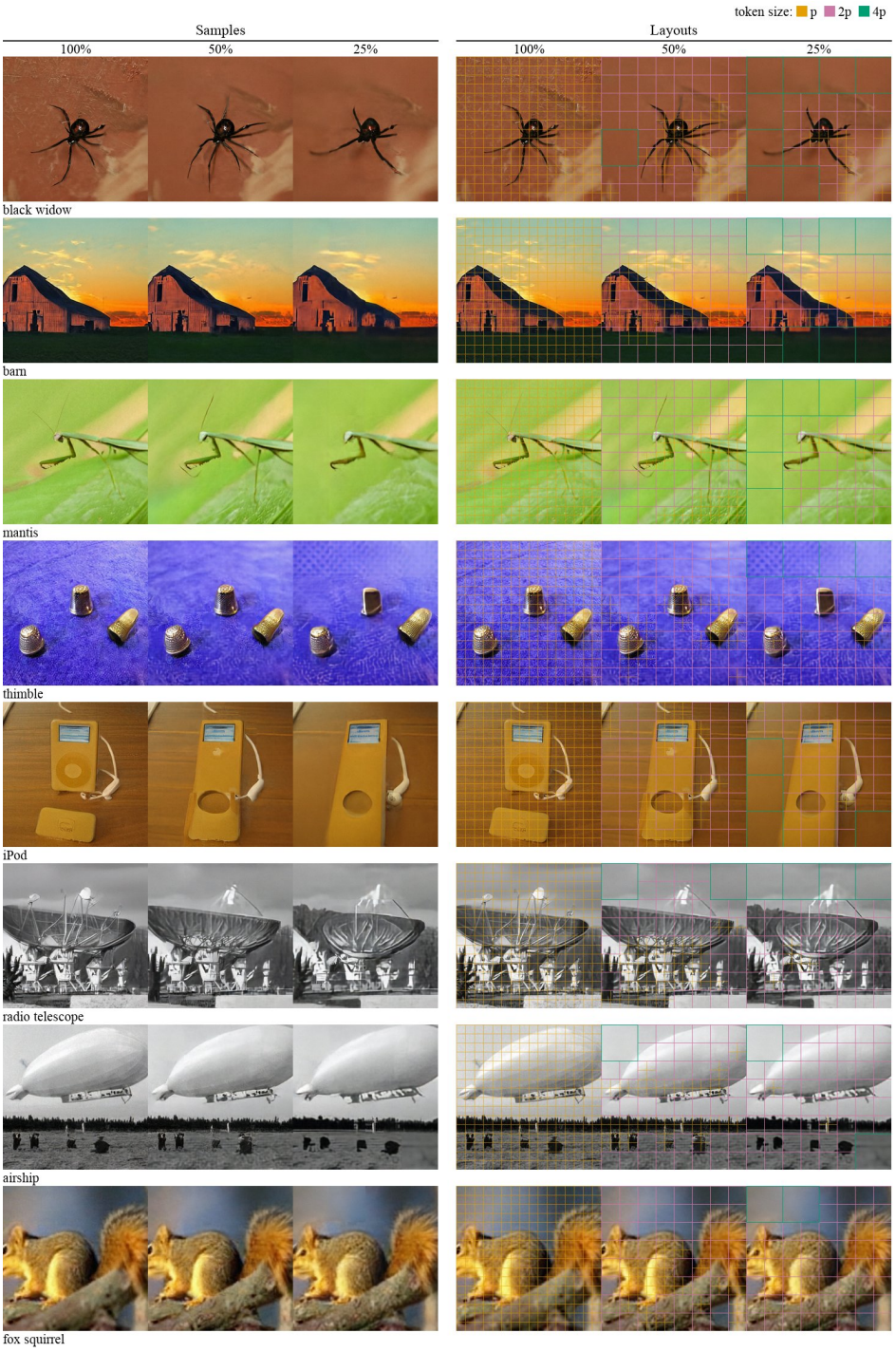

Figure 26: **BudgetPix on JiT-L/32 at 512^2: samples and their layouts, one ImageNet class per row.**

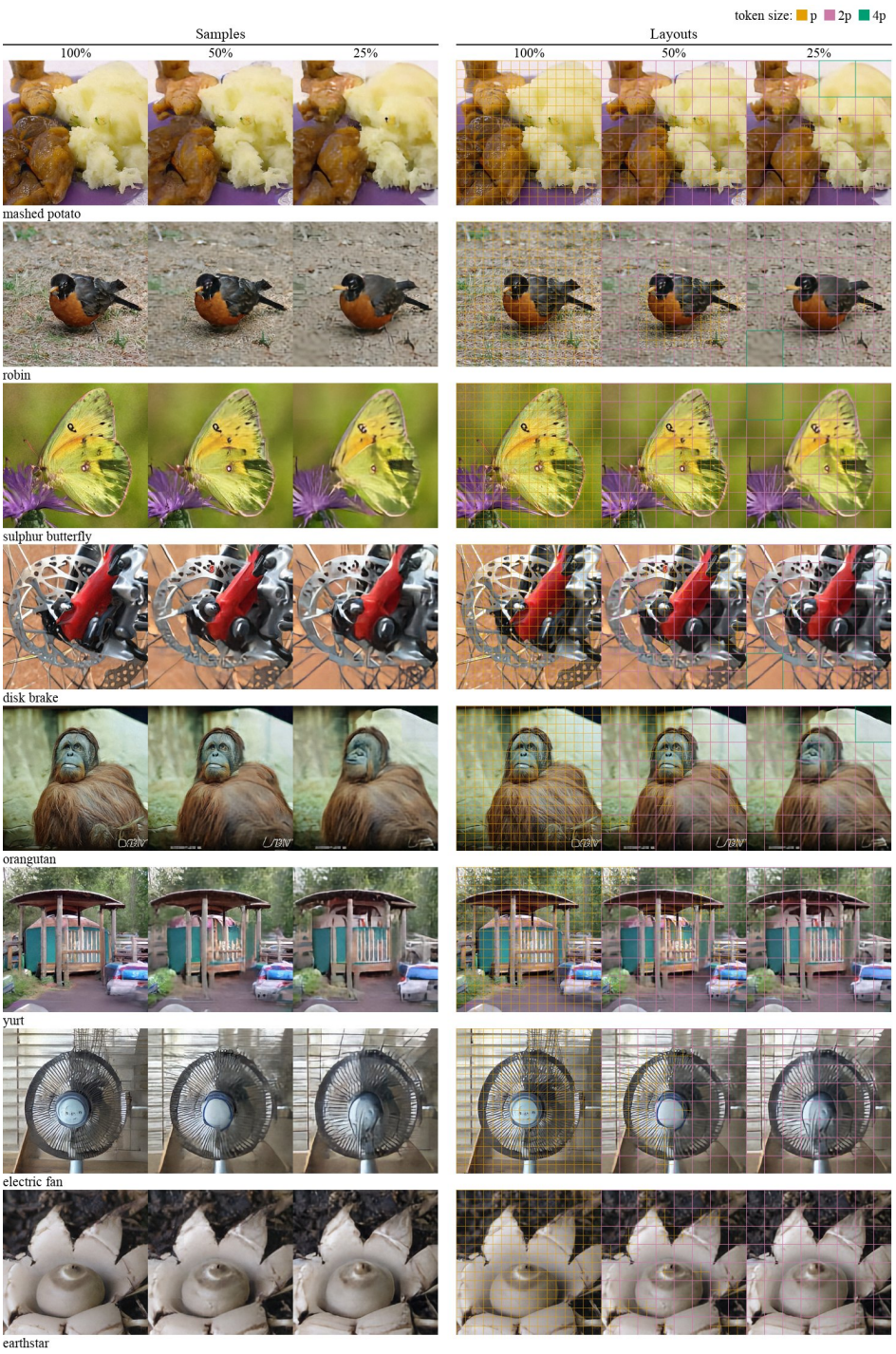

Figure 27: **BudgetPix on JiT-L/32 at 512^2: samples and their layouts, one ImageNet class per row (continued).**

A.7 BUDGETPIX LAYOUTS (MINIT2I-L/16, 512^2)

Figures 28 and 29 show additional qualitative results with their layouts on MiniT2I-L/16 at 512^2. Each row is one prompt; the columns are the samples at 100, 50 and 25 % of the grid, then the same images with their layouts drawn over them (each token outlined in the colour of its size).

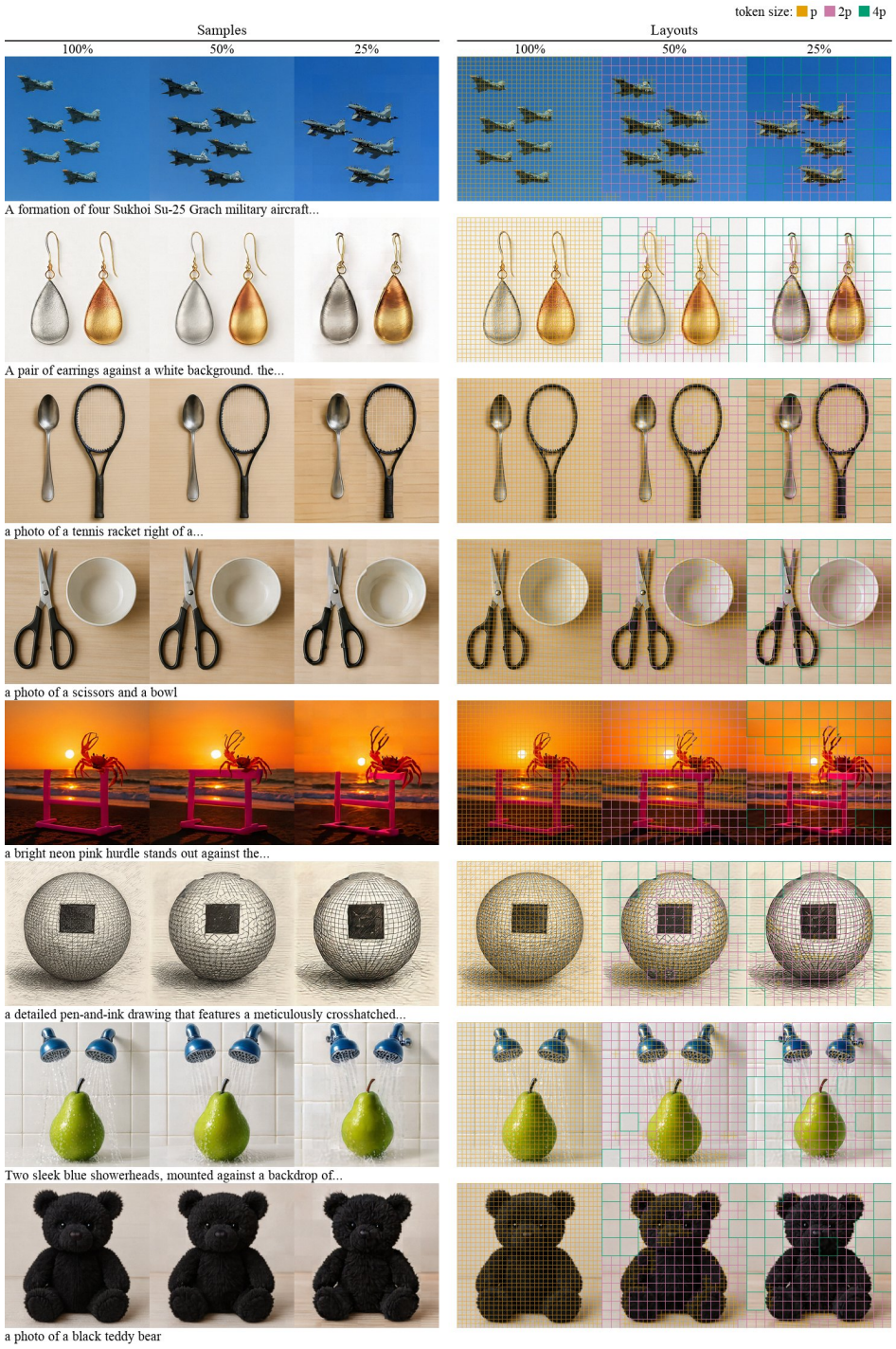

Figure 28: **BudgetPix on MiniT2I-L/16 at 512^2: samples and their layouts, one prompt per row.**

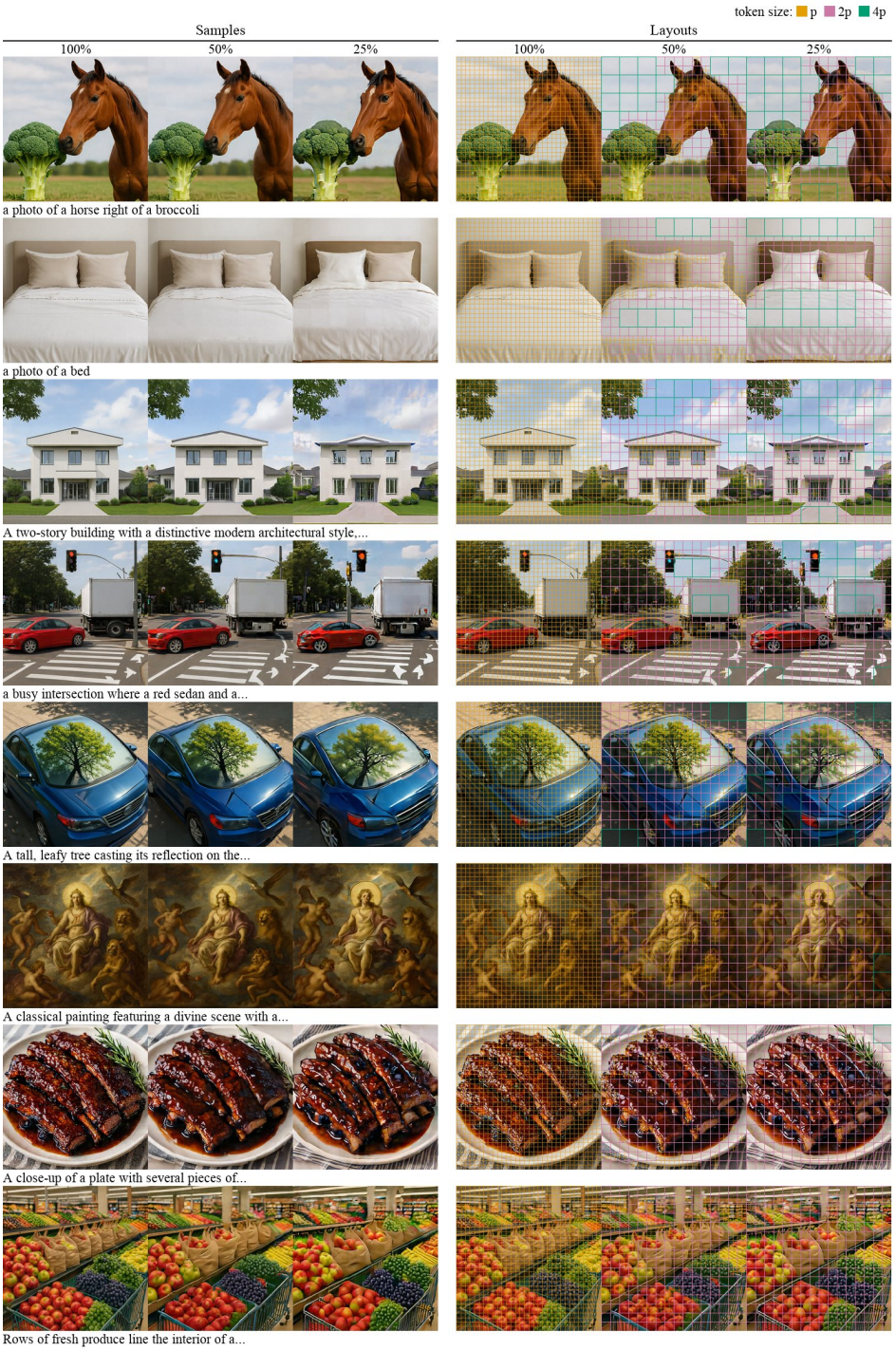

Figure 29: **BudgetPix on MiniT2I-L/16 at 512^2: samples and their layouts, one prompt per row (continued).**

A.8 BUDGETPIX LAYOUTS (PIXELDIT-T2I, 1024^2)

Figures 30 and 31 show additional qualitative results with their layouts on PixelDiT-T2I at 1024^2. Each row is one prompt; the columns are the samples at 100, 50 and 25 % of the grid, then the same images with their layouts drawn over them (each token outlined in the colour of its size).

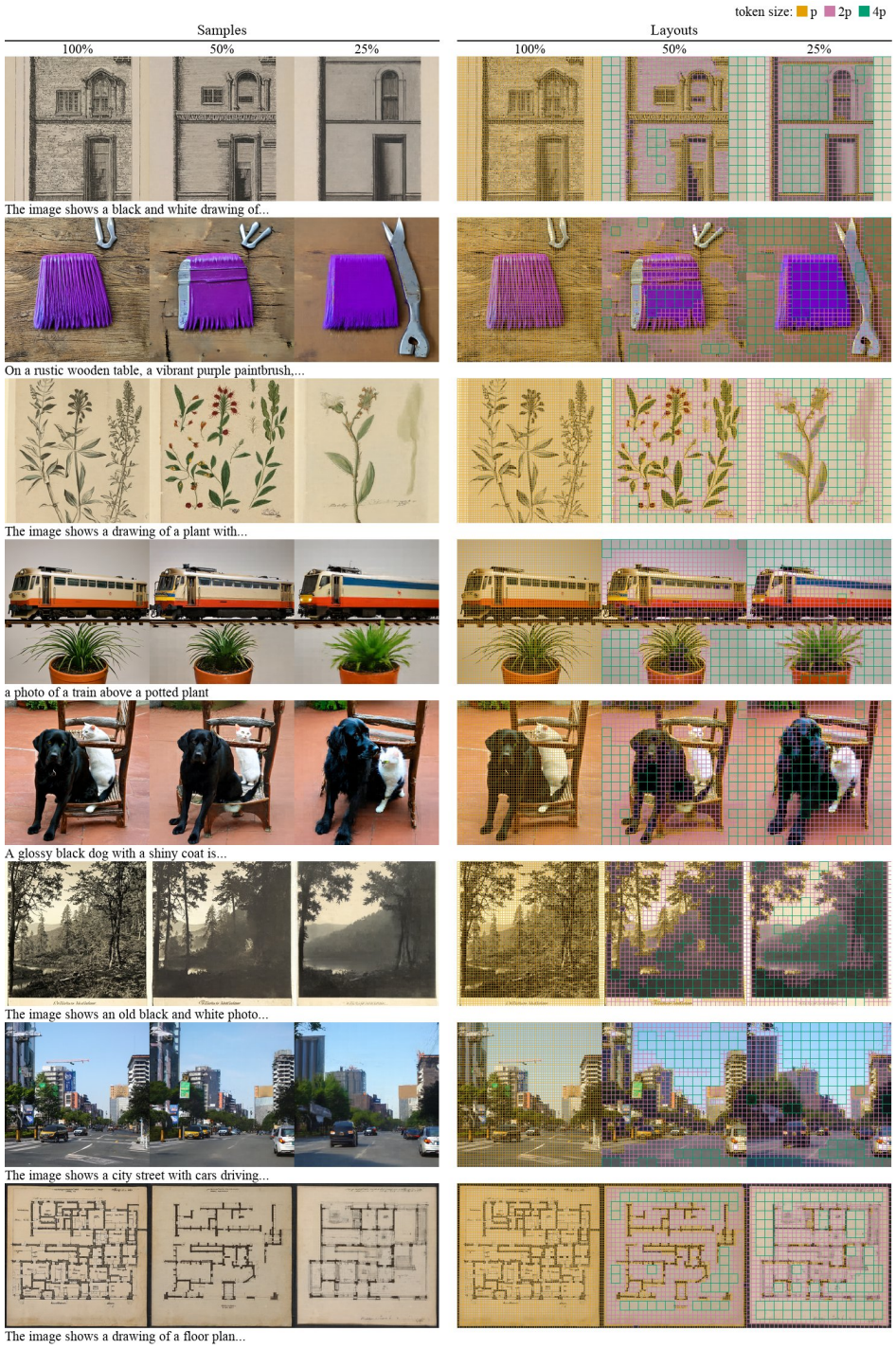

Figure 30: **BudgetPix on PixelDiT-T2I at 1024^2: samples and their layouts, one prompt per row.**

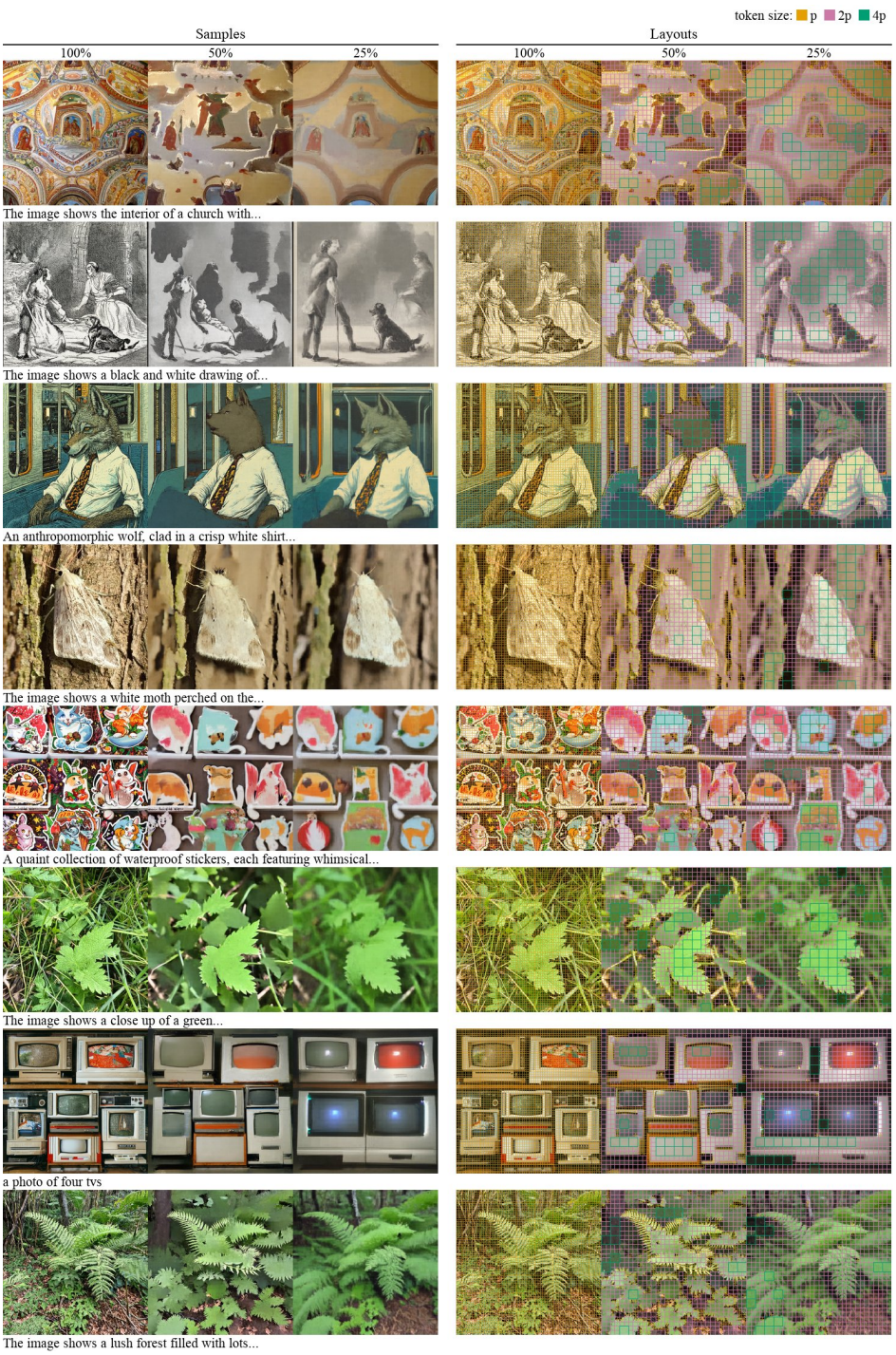

Figure 31: **BudgetPix on PixelDiT-T2I at 1024^2: samples and their layouts, one prompt per row (continued).**

B RESULTS IN FULL

The tables and figures behind Section 4.3, one subsection per result.

B.1 CLASS-CONDITIONAL GENERATION (JIT)

FID-50k and Inception score (IS) on ImageNet (see App. C.5) for the four JiT parents, JiT-L above JiT-B: Table 3 gives every value, and Figure 32 plots them with FID cut at 40.

Table 3: **Class-conditional generation on ImageNet, JiT-L and JiT-B: FID-50k and Inception score.**

		FID-50k ↓							Inception score ↑						
parent	method	100%	90%	80%	70%	60%	50%	25%	100%	90%	80%	70%	60%	50%	25%
JiT-L/32, 512^2	Base model	**2.66**	3.07	4.34	8.09	15.51	31.57	107.51	337	327	305	259	203	127	13
	ToMe	**2.66**	3.25	6.82	14.83	28.40	49.16	129.69	337	307	257	197	134	78	13
	FeatSim	**2.66**	3.20	6.04	16.62	47.19	110.60	172.31	337	312	264	179	77	16	5
	BudgetPix	2.71	**2.94**	**3.20**	**3.69**	**4.41**	**5.57**	**11.54**	**341**	**336**	**328**	**316**	**302**	**281**	**206**
JiT-L/16, 256^2	Base model	2.71	3.01	4.44	9.07	18.09	35.95	116.16	328	318	294	246	188	115	12
	ToMe	2.71	2.99	5.71	18.43	41.58	70.13	159.39	328	309	264	173	92	45	7
	FeatSim	2.71	3.45	7.77	22.56	56.21	120.23	188.34	328	298	241	150	61	13	4
	BudgetPix	**2.56**	**2.70**	**2.93**	**3.33**	**3.87**	**4.75**	**8.90**	**331**	**328**	**322**	**312**	**300**	**284**	**221**
JiT-B/32, 512^2	Base model	4.12	5.00	6.86	12.24	24.79	60.61	190.71	273	261	238	189	123	47	4
	ToMe	4.11	5.25	8.81	16.18	34.07	57.40	147.26	274	253	213	167	104	60	10
	FeatSim	4.11	5.48	10.55	26.88	66.38	140.23	192.26	274	248	196	120	47	10	4
	BudgetPix	**3.94**	**4.35**	**4.80**	**5.57**	**6.59**	**8.30**	**16.71**	**287**	**281**	**271**	**259**	**244**	**225**	**157**
JiT-B/16, 256^2	Base model	3.61	4.24	5.84	10.67	23.46	63.94	145.46	271	261	241	197	131	46	5
	ToMe	**3.60**	4.47	6.75	11.19	21.08	36.20	104.98	271	253	223	186	131	83	15
	FeatSim	**3.60**	4.85	8.72	20.12	49.63	115.89	166.16	271	247	202	131	56	11	5
	BudgetPix	**3.60**	**3.89**	**4.25**	**4.89**	**5.73**	**7.01**	**13.05**	**281**	**276**	**271**	**261**	**248**	**231**	**171**

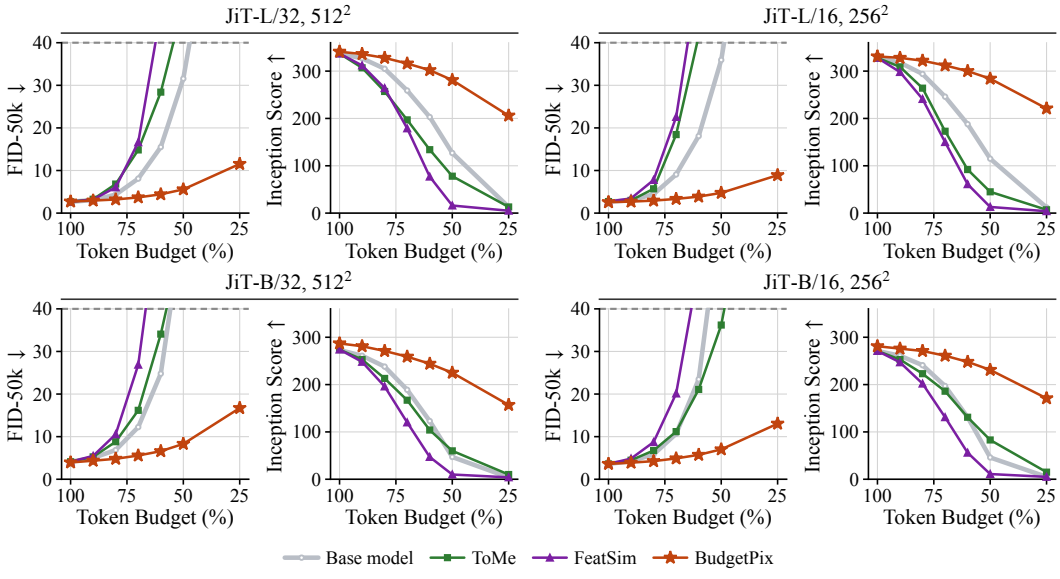

Figure 32: **Class-conditional generation on ImageNet, JiT-L and JiT-B.**

B.2 FAITHFULNESS TO THE DENSE IMAGE (JIT)

As in Figure 8(b), each image at a reduced budget is scored against the same model's image at the dense grid, from the same label and noise (see App. C.5): Figure 33 for JiT-L and JiT-B, and Table 4 for JiT-B, which adds CLIP similarity and splits PSNR between the pixels the layout merged and those it kept. A break (//) marks an axis that does not start at zero.

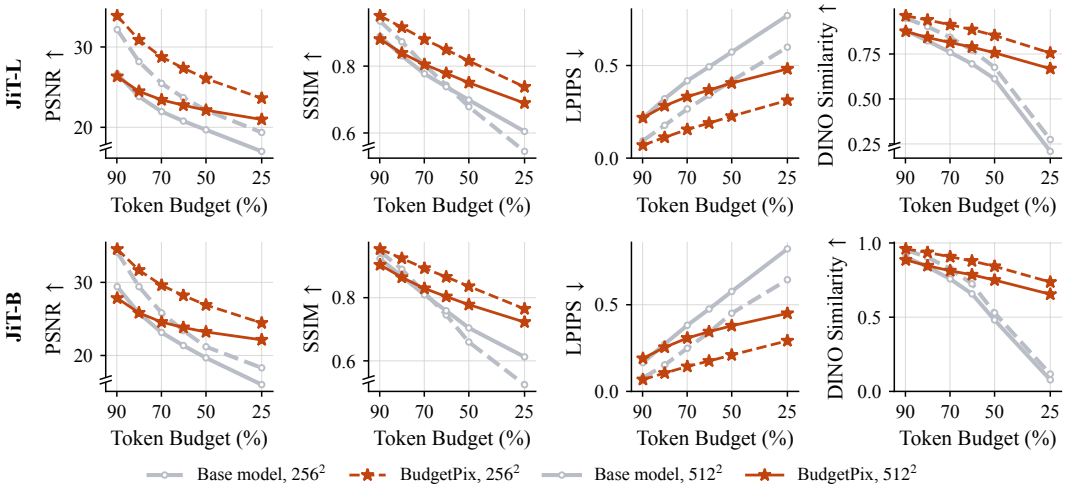

Figure 33: **Paired metrics against the token budget, JiT-L and JiT-B.**

Table 4: **Faithfulness to the dense image, JiT-B: paired metrics.**

		JiT-B/32, 512^2						JiT-B/16, 256^2					
metric	method	90%	80%	70%	60%	50%	25%	90%	80%	70%	60%	50%	25%
PSNR ↑	Base model	29.4	25.8	23.1	21.4	19.7	16.0	34.1	29.4	25.8	23.4	21.2	18.3
	BudgetPix	27.8	25.8	24.6	23.8	23.2	22.1	34.5	31.7	29.6	28.2	26.9	24.4
PSNR, merged pixels ↑	Base model	28.5	25.7	23.4	21.7	20.0	16.1	30.5	27.4	24.8	22.9	21.2	18.4
	BudgetPix	28.6	26.8	25.5	24.6	23.9	22.3	32.0	30.1	28.6	27.6	26.7	24.5
PSNR, kept pixels ↑	Base model	30.1	26.4	23.5	21.5	19.6	15.2	35.9	31.2	27.4	24.6	21.7	17.3
	BudgetPix	28.0	25.9	24.5	23.6	22.7	21.0	35.8	33.1	31.1	29.6	28.1	24.6
SSIM ↑	Base model	0.924	0.867	0.809	0.758	0.704	0.613	0.946	0.890	0.816	0.745	0.660	0.525
	BudgetPix	0.904	0.864	0.829	0.803	0.778	0.723	0.954	0.924	0.893	0.866	0.836	0.764
LPIPS ↓	Base model	0.166	0.273	0.382	0.475	0.576	0.822	0.077	0.153	0.249	0.340	0.451	0.644
	BudgetPix	0.190	0.254	0.308	0.345	0.379	0.450	0.068	0.106	0.144	0.175	0.210	0.292
CLIP similarity ↑	Base model	0.953	0.922	0.882	0.833	0.753	0.589	0.972	0.943	0.901	0.851	0.770	0.614
	BudgetPix	0.947	0.931	0.917	0.905	0.893	0.863	0.977	0.966	0.953	0.940	0.925	0.884
DINOv2 similarity ↑	Base model	0.907	0.839	0.758	0.657	0.481	0.078	0.954	0.903	0.827	0.725	0.529	0.117
	BudgetPix	0.885	0.847	0.813	0.786	0.750	0.654	0.960	0.936	0.907	0.879	0.843	0.737

B.3 TEXT-TO-IMAGE GENERATION (MINIT2I)

Table 5 gives the values behind Figures 8(a) and 34 by token budget, for MiniT2I-B/16 and MiniT2I-L/16 at 512^2. PixelDiT at 1024^2 is complete in Table 1.

Table 5: **Text-to-image generation, MiniT2I-B/16 and MiniT2I-L/16 at 512^2.**

		MiniT2I-B/16							MiniT2I-L/16						
metric	method	100%	90%	80%	70%	60%	50%	25%	100%	90%	80%	70%	60%	50%	25%
GenEval ↑	Base model	0.876	**0.884**	0.872	0.865	0.506	0.040	0.006	0.882	0.877	0.868	0.854	0.773	0.183	0.017
	ToMe	**0.877**	0.878	0.867	0.861	0.841	0.779	0.283	**0.886**	**0.891**	**0.886**	0.873	0.845	0.741	0.229
	FeatSim	**0.877**	0.882	0.871	0.865	0.830	0.610	0.106	**0.886**	0.880	0.850	0.798	0.676	0.292	0.007
	RTI	0.850	0.864	0.870	0.869	0.870	0.868	0.856	0.863	0.873	0.867	0.870	0.868	0.871	0.859
	BudgetPix	**0.877**	**0.884**	**0.875**	**0.879**	**0.875**	**0.871**	**0.866**	0.880	0.880	0.875	**0.875**	**0.874**	**0.880**	**0.874**
DPG-Bench ↑	Base model	84.2	83.7	83.7	82.8	74.3	23.2	6.8	84.7	84.4	83.6	83.0	80.6	54.1	8.0
	ToMe	84.1	**84.5**	84.3	84.0	82.9	80.4	62.8	84.5	**84.8**	84.3	83.4	81.2	77.0	54.6
	FeatSim	84.1	84.3	84.2	84.2	83.4	80.9	57.8	84.5	84.6	84.0	82.8	81.0	75.1	17.0
	RTI	82.3	83.8	83.8	84.0	83.8	83.7	82.5	82.9	84.0	84.1	84.4	83.8	83.9	83.0
	BudgetPix	**84.3**	84.3	**84.5**	**84.4**	**84.1**	**84.4**	**83.8**	**84.8**	84.7	**84.7**	**84.6**	**84.7**	**84.7**	**84.3**
CLIP ↑	Base model	28.21	28.12	28.03	27.50	23.39	17.86	15.99	28.88	28.81	28.59	28.30	27.25	22.17	16.20
	ToMe	**28.61**	**28.68**	28.60	28.35	28.01	27.38	24.48	**28.95**	**29.09**	**29.14**	28.80	28.17	27.21	23.48
	FeatSim	**28.61**	28.67	**28.72**	**28.72**	**28.60**	27.61	23.08	**28.95**	28.84	28.55	27.92	27.02	25.68	17.51
	RTI	28.35	28.10	28.17	28.18	28.22	**28.21**	**28.29**	28.71	28.55	28.52	28.61	28.54	28.57	28.67
	BudgetPix	28.11	28.12	28.07	28.09	28.06	28.06	27.95	28.87	28.91	28.89	**28.92**	**28.96**	**29.03**	**29.27**
PickScore ↑	Base model	**22.77**	**22.72**	**22.63**	22.29	20.34	18.66	18.28	22.82	22.72	22.54	22.28	21.53	19.41	18.11
	ToMe	22.51	22.43	22.36	22.19	21.94	21.56	20.01	**22.83**	22.72	22.61	22.37	21.95	21.36	19.72
	FeatSim	22.51	22.47	22.39	22.26	21.98	21.36	19.65	**22.83**	22.66	22.40	21.99	21.40	20.53	18.29
	RTI	22.09	22.37	22.37	22.36	22.33	22.30	**22.07**	22.41	22.71	22.69	22.68	22.66	**22.64**	**22.39**
	BudgetPix	22.50	22.49	22.46	**22.43**	**22.38**	**22.31**	22.00	22.82	**22.81**	**22.79**	**22.76**	**22.72**	22.63	22.33
ImageReward ↑	Base model	**1.16**	1.14	1.12	0.96	-0.45	-1.95	-2.23	**1.23**	1.20	1.17	1.10	0.70	-1.17	-2.22
	ToMe	1.15	1.14	1.12	1.06	0.97	0.77	-0.38	**1.23**	1.21	1.19	1.11	0.96	0.67	-0.67
	FeatSim	1.15	1.15	1.12	1.08	0.99	0.67	-0.91	**1.23**	1.20	1.15	1.04	0.80	0.06	-2.17
	RTI	1.03	1.12	1.12	1.12	1.11	1.10	1.07	1.14	1.21	1.21	**1.21**	1.20	**1.20**	1.16
	BudgetPix	**1.16**	**1.17**	**1.16**	**1.16**	**1.14**	**1.14**	**1.09**	1.21	**1.22**	**1.22**	**1.21**	**1.21**	**1.20**	**1.17**

B.4 WALL-CLOCK SPEED-UP (EVERY PARENT)

Speed-up is measured wall-clock time over the released model at its dense grid (see App. C.7). Figure 34 is Figure 8(a) for MiniT2I-B/16 (DPG: DPG-Bench; IR: ImageReward). Figure 35 (dashed: the released model, 1×) and Table 6 give the speed-up along the budget, Figure 36 quality against it (FID cut at 40), and Tables 7–13 the seconds per image and the speed-up of every method. PixelDiT was timed for the base model and ours only. A break (//) marks an axis that does not start at zero.

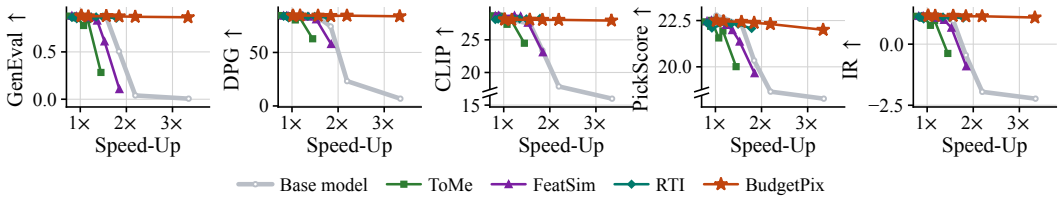

Figure 34: **Five metrics against the measured wall-clock speed-up, MiniT2I-B/16 at 512^2.**

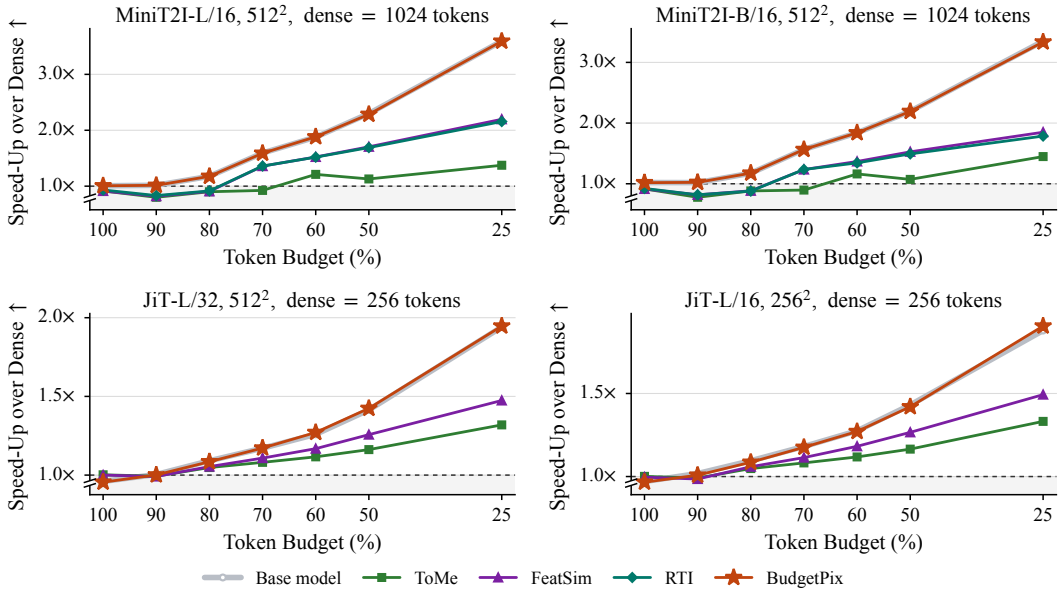

Figure 35: **Wall-clock speed-up against the token budget, four parents.**

Table 6: **Wall-clock speed-up over the released model, every parent.**

parent	method	100%	90%	80%	70%	60%	50%	25%
MiniT2I-L/16, 512^2	Base model	1.01×	1.02×	1.18×	1.59×	1.88×	2.30×	3.59×
	ToMe	0.91×	0.80×	0.90×	0.92×	1.21×	1.13×	1.37×
	FeatSim	0.91×	0.82×	0.91×	1.36×	1.52×	1.70×	2.20×
	RTI	0.93×	0.83×	0.92×	1.36×	1.52×	1.69×	2.16×
	BudgetPix	1.01×	1.01×	1.18×	1.59×	1.88×	2.29×	3.59×
MiniT2I-B/16, 512^2	Base model	1.02×	1.02×	1.17×	1.57×	1.84×	2.19×	3.35×
	ToMe	0.92×	0.78×	0.88×	0.90×	1.16×	1.07×	1.45×
	FeatSim	0.91×	0.82×	0.88×	1.23×	1.37×	1.53×	1.85×
	RTI	0.92×	0.82×	0.88×	1.24×	1.34×	1.49×	1.79×
	BudgetPix	1.02×	1.02×	1.18×	1.56×	1.84×	2.19×	3.34×
PixelDiT-T2I, 1024^2	Base model	0.99×	0.79×	0.88×	0.99×	1.12×	1.29×	1.94×
	BudgetPix	1.00×	0.79×	0.88×	1.00×	1.12×	1.29×	1.95×
JiT-L/32, 512^2	Base model	0.96×	1.00×	1.09×	1.17×	1.26×	1.41×	1.94×
	ToMe	1.00×	0.99×	1.05×	1.08×	1.12×	1.16×	1.32×
	FeatSim	1.00×	0.99×	1.05×	1.11×	1.17×	1.26×	1.47×
	BudgetPix	0.95×	1.00×	1.08×	1.17×	1.27×	1.42×	1.95×
JiT-B/32, 512^2	Base model	0.95×	0.92×	0.96×	1.02×	1.08×	1.15×	1.39×
	ToMe	1.00×	0.92×	0.96×	0.99×	1.02×	1.07×	1.20×
	FeatSim	1.01×	0.97×	1.01×	1.05×	1.09×	1.15×	1.28×
	BudgetPix	0.94×	0.92×	0.97×	1.00×	1.06×	1.14×	1.36×
JiT-L/16, 256^2	Base model	0.97×	1.02×	1.09×	1.18×	1.27×	1.43×	1.88×
	ToMe	1.00×	0.99×	1.05×	1.08×	1.12×	1.17×	1.33×
	FeatSim	1.00×	0.98×	1.06×	1.11×	1.18×	1.27×	1.49×
	BudgetPix	0.97×	1.01×	1.09×	1.18×	1.27×	1.42×	1.91×
JiT-B/16, 256^2	Base model	0.97×	0.98×	1.03×	1.11×	1.19×	1.32×	1.63×
	ToMe	0.99×	0.90×	0.96×	0.98×	1.04×	1.06×	1.21×
	FeatSim	1.00×	0.96×	1.01×	1.04×	1.09×	1.16×	1.28×
	BudgetPix	0.95×	0.96×	1.02×	1.09×	1.18×	1.28×	1.58×

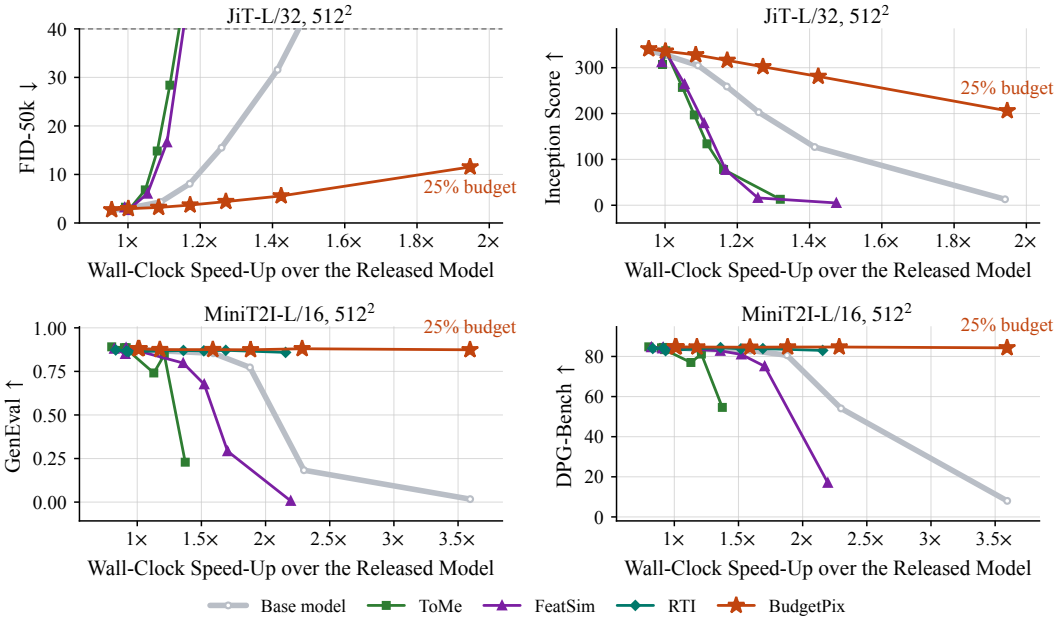

Figure 36: **Quality against measured speed-up, JiT-L/32 and MiniT2I-L/16 at 512^2.**

Table 7: **Wall-clock per image and speed-up, MiniT2I-L/16 at 512^2.**

measure	method	100%	90%	80%	70%	60%	50%	25%
seconds per image	Base model	4.912	4.870	4.205	3.105	2.634	2.151	1.376
	ToMe	5.413	6.171	5.493	5.353	4.082	4.382	3.600
	FeatSim	5.419	6.023	5.452	3.643	3.249	2.903	2.252
	RTI	5.305	5.947	5.402	3.636	3.257	2.926	2.294
	BudgetPix	4.913	4.872	4.206	3.113	2.626	2.164	1.376
speed-up	Base model	1.01×	1.02×	1.18×	1.59×	1.88×	2.30×	3.59×
	ToMe	0.91×	0.80×	0.90×	0.92×	1.21×	1.13×	1.37×
	FeatSim	0.91×	0.82×	0.91×	1.36×	1.52×	1.70×	2.20×
	RTI	0.93×	0.83×	0.92×	1.36×	1.52×	1.69×	2.16×
	BudgetPix	1.01×	1.01×	1.18×	1.59×	1.88×	2.29×	3.59×

Table 8: **Wall-clock per image and speed-up, MiniT2I-B/16 at 512^2.**

measure	method	100%	90%	80%	70%	60%	50%	25%
seconds per image	Base model	1.782	1.784	1.549	1.161	0.988	0.828	0.543
	ToMe	1.986	2.340	2.061	2.029	1.564	1.695	1.255
	FeatSim	1.988	2.222	2.057	1.472	1.329	1.191	0.983
	RTI	1.970	2.223	2.067	1.471	1.353	1.218	1.018
	BudgetPix	1.785	1.776	1.547	1.161	0.988	0.829	0.545
speed-up	Base model	1.02×	1.02×	1.17×	1.57×	1.84×	2.19×	3.35×
	ToMe	0.92×	0.78×	0.88×	0.90×	1.16×	1.07×	1.45×
	FeatSim	0.91×	0.82×	0.88×	1.23×	1.37×	1.53×	1.85×
	RTI	0.92×	0.82×	0.88×	1.24×	1.34×	1.49×	1.79×
	BudgetPix	1.02×	1.02×	1.18×	1.56×	1.84×	2.19×	3.34×

Table 9: **Wall-clock per image and speed-up, PixelDiT-T2I at 1024^2.**

measure	method	100%	90%	80%	70%	60%	50%	25%
seconds per image	Base model	2.992	3.760	3.383	2.983	2.652	2.306	1.527
	BudgetPix	2.974	3.754	3.378	2.973	2.651	2.307	1.526
speed-up	Base model	0.99×	0.79×	0.88×	0.99×	1.12×	1.29×	1.94×
	BudgetPix	1.00×	0.79×	0.88×	1.00×	1.12×	1.29×	1.95×

Table 10: **Wall-clock per image and speed-up, JiT-L/32 at 512^2.**

measure	method	100%	90%	80%	70%	60%	50%	25%
seconds per image	Base model	0.394	0.376	0.346	0.323	0.300	0.267	0.195
	ToMe	0.377	0.381	0.361	0.350	0.339	0.325	0.287
	FeatSim	0.378	0.381	0.359	0.341	0.324	0.301	0.256
	BudgetPix	0.396	0.377	0.348	0.322	0.297	0.265	0.194
speed-up	Base model	0.96×	1.00×	1.09×	1.17×	1.26×	1.41×	1.94×
	ToMe	1.00×	0.99×	1.05×	1.08×	1.12×	1.16×	1.32×
	FeatSim	1.00×	0.99×	1.05×	1.11×	1.17×	1.26×	1.47×
	BudgetPix	0.95×	1.00×	1.08×	1.17×	1.27×	1.42×	1.95×

Table 11: **Wall-clock per image and speed-up, JiT-B/32 at 512^2.**

measure	method	100%	90%	80%	70%	60%	50%	25%
seconds per image	Base model	0.131	0.134	0.129	0.122	0.115	0.108	0.089
	ToMe	0.124	0.134	0.129	0.125	0.122	0.116	0.103
	FeatSim	0.122	0.127	0.123	0.118	0.113	0.108	0.097
	BudgetPix	0.132	0.135	0.128	0.123	0.116	0.109	0.091
speed-up	Base model	0.95×	0.92×	0.96×	1.02×	1.08×	1.15×	1.39×
	ToMe	1.00×	0.92×	0.96×	0.99×	1.02×	1.07×	1.20×
	FeatSim	1.01×	0.97×	1.01×	1.05×	1.09×	1.15×	1.28×
	BudgetPix	0.94×	0.92×	0.97×	1.00×	1.06×	1.14×	1.36×

Table 12: **Wall-clock per image and speed-up, JiT-L/16 at 256^2.**

measure	method	100%	90%	80%	70%	60%	50%	25%
seconds per image	Base model	0.383	0.365	0.339	0.314	0.291	0.259	0.197
	ToMe	0.371	0.374	0.354	0.343	0.332	0.319	0.279
	FeatSim	0.372	0.377	0.350	0.333	0.314	0.293	0.249
	BudgetPix	0.384	0.368	0.342	0.315	0.292	0.261	0.195
speed-up	Base model	0.97×	1.02×	1.09×	1.18×	1.27×	1.43×	1.88×
	ToMe	1.00×	0.99×	1.05×	1.08×	1.12×	1.17×	1.33×
	FeatSim	1.00×	0.98×	1.06×	1.11×	1.18×	1.27×	1.49×
	BudgetPix	0.97×	1.01×	1.09×	1.18×	1.27×	1.42×	1.91×

Table 13: **Wall-clock per image and speed-up, JiT-B/16 at 256^2.**

measure	method	100%	90%	80%	70%	60%	50%	25%
seconds per image	Base model	0.121	0.119	0.113	0.106	0.098	0.089	0.072
	ToMe	0.118	0.130	0.122	0.119	0.113	0.111	0.097
	FeatSim	0.117	0.122	0.116	0.112	0.107	0.101	0.091
	BudgetPix	0.123	0.122	0.115	0.108	0.099	0.091	0.074
speed-up	Base model	0.97×	0.98×	1.03×	1.11×	1.19×	1.32×	1.63×
	ToMe	0.99×	0.90×	0.96×	0.98×	1.04×	1.06×	1.21×
	FeatSim	1.00×	0.96×	1.01×	1.04×	1.09×	1.16×	1.28×
	BudgetPix	0.95×	0.96×	1.02×	1.09×	1.18×	1.28×	1.58×

B.5 BUDGET MODE AGAINST THRESHOLD MODE (JIT-L/32)

In budget mode δ is bisected so that every image gets at most the requested token count; in threshold mode δ is fixed and each image takes the count its content gives. Threshold mode sweeps δ from -1 to $+4$ on JiT-L/32 at 512^2, with the images, noise and protocol of Table 3; budget mode is that table's points, with 40, 30, 20 and 10 % added. Figure 37 draws both down to a fifth of the grid, with bars of one standard deviation of the token count across images, and Table 14 lists every point. Read at the same mean token count the two modes lie on nearly one curve. What threshold mode gives up is the per-image guarantee, which is why every other number in the paper is in budget mode; see App. D.9 for a pooled budget. Figure 38 shows the layout as the budget falls for a penguin before a plain wall (top) and a textured toyshop (bottom).

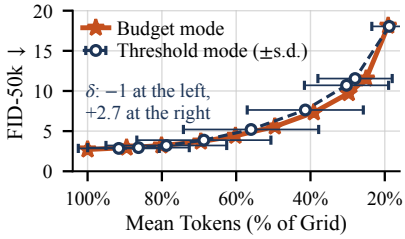

Figure 37: **Budget mode against threshold mode, FID-50k on JiT-L/32 at 512^2.**

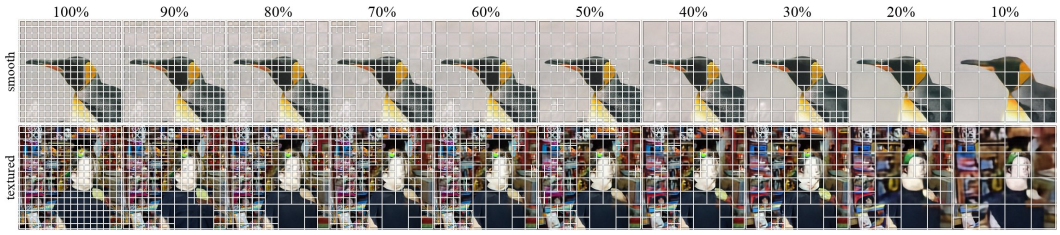

Figure 38: **The layout as the budget falls, JiT-L/32 at 512^2, budget mode.**

Table 14: **Budget mode against threshold mode, JiT-L/32 at 512^2: every point of Figure 37.**

budget mode, budget	100%	90%	80%	70%	60%	50%	40%	30%	25%	20%	10%
tokens (% of grid)	100.0	89.5	80.1	69.5	60.2	49.6	39.1	29.7	25.0	19.1	9.8
FID-50k ↓	2.71	2.94	3.20	3.69	4.41	5.57	7.33	9.73	11.54	18.12	44.62
Inception score ↑	341	336	328	316	302	281	256	228	206	164	76
threshold mode, δ	-1	-0.5	0	$+0.5$	$+1$	$+1.5$	$+1.9$	$+2$	$+2.7$	$+3$	$+4$
tokens (% of grid)	91.6	86.3	78.8	68.7	56.0	41.4	30.3	28.0	18.8	17.1	9.2
s.d. across images (points)	10.9	13.7	16.2	18.0	18.2	15.6	11.3	10.0	4.7	4.9	3.5
FID-50k ↓	2.85	2.95	3.21	3.88	5.21	7.63	10.71	11.55	18.04	21.33	47.13
Inception score ↑	337	334	322	308	285	251	215	206	158	140	64

B.6 ONE-STEP GENERATION (PIXEL MEANFLOW)

Table 15 completes Figure 10; see App. A.5 for samples. The parent is the one-step pixel MeanFlow model (Lu et al., 2026) pMF-L/16, class-conditional on ImageNet at 256^2 (256 tokens; token scales of 16, 32 and 64 pixels), fine-tuned with its own objective (see App. C.4). One step leaves no trajectory to read a layout from, so each image takes the entropy quadtree of one of 20 training images of its class, picked by the image index and bisected to the budget; guidance is the released model's own at every budget. FID-50k follows pMF, 50 images per class against JiT's reference statistics at 256^2 (the released L/16 reads 2.54 here and 2.52 in its paper). The base model is the released weights on our layouts, without training; ToMe and FeatSim are the JiT implementations ported to pMF.

Table 15: **One-step generation, pMF-L/16 at 256^2: FID-50k and Inception score.**

metric	method	100%	90%	80%	70%	60%	50%	40%	30%	25%
	tokens	*256*	*230*	*205*	*179*	*154*	*128*	*102*	*77*	*64*
FID-50k ↓	Base model	**2.54**	6.91	118.76	242.62	257.20	244.95	236.82	253.49	233.64
	ToMe	**2.54**	3.06	5.93	14.66	36.04	70.62	109.40	122.59	149.41
	FeatSim	**2.54**	5.71	20.44	57.37	100.26	134.35	180.26	194.43	192.94
	BudgetPix	2.65	**2.65**	**2.77**	**3.06**	**3.46**	**3.94**	**4.50**	**4.96**	**5.44**
Inception score ↑	Base model	263	226	20	2	2	2	3	3	3
	ToMe	263	240	203	150	84	36	14	11	6
	FeatSim	263	209	123	49	20	11	4	3	3
	BudgetPix	**271**	**264**	**257**	**251**	**241**	**238**	**232**	**231**	**224**

C Details of the method and the experiments

Details that Sections 3 and 4 leave out.

C.1 Architecture of the added modules

Table 16 lists how each family embeds and decodes a coarse token of side np ($n = 2, 4$; p is the released patch and D the backbone width), as built in the code; Sections 3.3 and 3.4 describe the modules. Base tokens ($n = 1$) use the released embedding and head unchanged, and each coarse scale has its own Patch Aggregator. PixelDiT's released pixel stage decodes every pixel at every budget, which sets its cost floor, so instead of a pixel-space refiner, CellRefine corrects each cell's conditioning of that stage.

Table 16: **Architecture of the added modules, every family.**

	JiT 256^2 *and* 512^2	pMF-L/16 256^2	MiniT2I 512^2	PixelDiT-T2I 1024^2
width D	768 (B), 1024 (L)	1024	768 (B), 1248 (L)	1536
cell embedding (released)	$p \times p$ convolution to 128 channels, then 1×1 convolution to D			linear map of the flattened $16 \times 16 \times 3$ cell
downscale path	released embedding of the patch resized to $p \times p$	none		
split path: scale mixer	n^2 cell embeddings + learned positions; 1×1 conv to 256, GELU, depthwise $n \times n$ conv, 1×1 conv, GELU, linear to D	n^2 cell embeddings + learned positions; 1×1 conv to 256, depthwise $n \times n$ conv, linear to D; no activation		—
coarse token	downscale path + scale mixer + scale embedding	mean of the n^2 cell embeddings + scale mixer		
position	sin–cos embedding and RoPE at the token centre	mean of the covered cells' learned positions; RoPE at the token centre	sin–cos embedding and RoPE at the token centre	RoPE at the token centre
upscale path	released head's $p \times p$ prediction, bilinearly upsampled to $np \times np$			the token's state copied to its n^2 cells
refiner tokens	4×4-pixel pieces of the $p \times p$ prediction: 16 ($p = 16$) or 64 ($p = 32$)	16 pieces of 4×4 pixels		one learned slot per cell (4 or 16)
refiner blocks	2 transformer blocks, width 64, 4 heads, adaLN on the token's final hidden state only			
refiner output	linear to $4 \cdot 4 \cdot 3\, n^2$ values per piece, pixel shuffle by n: a residual on the upscale path			linear to a correction of each cell's 1536-d conditioning, then the released pixel stage
one refiner per	coarse scale	coarse scale and branch (u; v in training)	coarse scale	coarse scale
zero-initialised	last layer of the mixer, scale embedding, refiner adaLN and output	last layer of the mixer, refiner adaLN and output		
added parameters	2.39M (B/16), 2.40M (B/32), 3.06M (L/16, L/32)	4.77M	2.26M (B), 3.50M (L)	4.37M

C.2 TRAINING OBJECTIVE

The fine-tuning objective of Section 3.5, $w(t)\|x - \hat{x}\|_2^2$, keeps the weighting each parent was pre-trained with, so no comparison gains from a change of loss. The parents regress velocity; with $z_t = t\,x + (1-t)\,\varepsilon$, the target and predicted velocities are $(x - z_t)$ and $(\hat{x} - z_t)$ over the same factor, so

$$\mathcal{L}(\theta) = \mathbb{E}\Big\| \frac{x - z_t}{\max(1-t,\, t_\epsilon)} - \frac{\hat{x} - z_t}{\max(1-t,\, t_\epsilon)} \Big\|_2^2 = \mathbb{E}\Big[\frac{1}{\max(1-t,\, t_\epsilon)^2}\, \|x - \hat{x}\|_2^2 \Big], \tag{3}$$

and $w(t) = \max(1-t,\, t_\epsilon)^{-2}$, where each parent's own clip t_ϵ keeps the weight finite as $t \to 1$. PixelDiT regresses $\varepsilon - x$ at noise level $\sigma = 1 - t$: the same objective with $w = \sigma^{-2}$, and its own sampling of σ (logit-normal, shift 4.0).

C.3 ZERO-INITIALISATION CHECK

Every added path starts at zero, so training starts from the released model (Section 4.1). Table 17 checks this on the seven parents: max $|\Delta|$ is the largest absolute difference from the released model's output on the same inputs, and FID is dense-grid FID-50k on ImageNet for JiT and held-out FID for MiniT2I and PixelDiT (see App. C.5).

Table 17: **Zero-initialisation check, every parent.**

quantity	JiT-B/16 256^2	JiT-L/16 256^2	JiT-B/32 512^2	JiT-L/32 512^2	MiniT2I-B/16 512^2	MiniT2I-L/16 512^2	PixelDiT-T2I 1024^2
max $\|\Delta\|$ at initialisation	1.9×10^{-6}	1.6×10^{-6}	2.5×10^{-6}	1.9×10^{-6}	2.0×10^{-6}	4.5×10^{-6}	9.6×10^{-4}
FID, released model	3.61	2.71	4.12	2.66	24.6	24.8	43.7
FID, adapted model at step 0	3.61	2.71	4.12	2.66	24.6	24.8	43.7
FID, adapted model after training	3.60	2.56	3.94	2.71	17.1	22.9	36.2

C.4 FINE-TUNING AND SAMPLING SETTINGS

Tables 18 and 19 give each parent's fine-tuning recipe, read from its run, and the sampler behind its benchmark scores. Every parent starts from its released checkpoint (for JiT, its EMA weights), all weights are fine-tuned, and one checkpoint per parent serves every budget. MiniT2I-L's 120K mix is BLIP3o-60k (58,859 images), DALL-E 3 (19,016) and ShareGPT-4o-Image (41,355), with 1,024 held out. The held-out FIDs use 50 steps (see App. C.5), and the wall-clock runs the settings of Table 21.

Shared by the seven parents of Section 4.1. *Layout:* the entropy quadtree on a 512-bin luma histogram (entropy in $[0, 9]$ bits), with $\tau_{2p} = 6.0$ and $\tau_{4p} = 4.0$ bits; the offset $\delta \sim \mathcal{U}(0, 2)$ is drawn per image and step; $a = 0.15$ of the images train on the dense grid, and $b = 0.10$ of the rest (8.5 % of all images) on the random layout, whose cell entropies are drawn uniformly in $[0, 9]$ bits (Section 3.5). No image augmentation: a centre crop and a resize fix the geometry. *Optimisation:* AdamW without weight decay, a linear warm-up then a constant learning rate, the added modules at $8\times$ the backbone rate, EMA 0.9999 in fp32 (the evaluated weights), bf16 autocast with fp32 master weights, seed 0, four GPUs. *Sampling:* the first n_w steps run on the dense grid, and the layout is recomputed every $m = 5$ steps.

Table 18: **Fine-tuning and sampling settings, JiT parents (ImageNet-1k).**

setting	JiT-B/16 256^2	JiT-B/32 512^2	JiT-L/16 256^2	JiT-L/32 512^2
patch / token scales	16 / 16, 32, 64	32 / 32, 64, 128	16 / 16, 32, 64	32 / 32, 64, 128
parameters, base + added	131.3M + 2.39M	133.4M + 2.40M	459.1M + 3.06M	461.8M + 3.06M
steps / epochs / images seen	142,987 / 100 / 128.1M		71,494 / 50 / 64.1M	
global batch (per GPU × GPUs × accum.)	896 ($112 \times 4 \times 2$)			896 ($56 \times 4 \times 4$)
learning rate, backbone / added	3.5×10^{-5} / 2.8×10^{-4}			
learning-rate warm-up	1k steps			
optimiser	AdamW $(0.9, 0.95)$, clip 3.0			
noise scale	1.0	2.0	1.0	2.0
timestep t	logit-normal$(-0.8, 0.8)$			
label dropout	0.1			
sampler, guidance	50 Heun, 3.0 on $t \in [0.1, 1]$			
dense warm-up steps n_w	10	5	10	5
FID samples	50k, class-balanced, against training-set statistics			

Table 19: **Fine-tuning and sampling settings, text-to-image parents.**

setting	MiniT2I-B/16	MiniT2I-L/16	PixelDiT-T2I 1024^2
	512^2		
patch / token scales	16 / 16, 32, 64		
parameters, base + added	258.1M + 2.26M	911.8M + 3.50M	1302.5M + 4.37M
training data	CC12M, LLaVA recaptions, 1M images	CC12M (1M), then the 120K mix	PD12M, 200K images
steps / images seen	100k / 25.6M	40k (24k CC12M, then 16k mix) / 10.2M	30k / 0.96M
global batch (per GPU × GPUs × accum.)	256 ($16 \times 4 \times 4$)	256 ($4 \times 4 \times 16$)	32 ($4 \times 4 \times 2$)
learning rate, backbone / added	10^{-5} / 8×10^{-5}		2×10^{-5} / 1.6×10^{-4}
learning-rate warm-up	1k steps		2k steps
optimiser	AdamW $(0.9, 0.95)$, $\epsilon\ 10^{-8}$, clip 3.0		AdamW $(0.9, 0.999)$, $\epsilon\ 10^{-15}$, clip 0.1
diffusion law	x-prediction, noise scale 2.0, $t \sim$ logit-normal$(-0.8, 0.8)$		flow matching, velocity target, shift 4.0, $\sigma \sim$ logit-normal$(0, 1)$
caption dropout	0.1, attention mask zeroed		0.1, null embedding
text encoder (frozen)	flan-T5-large, 256 tokens		Gemma-2-2B-it, 300 tokens
memory	none		gradient checkpointing, EMA on CPU
sampler, guidance	100 Euler, 5.0		25 DPM-Solver++(2M) (Lu et al., 2025), 4.5
dense warm-up steps n_w	5		
held-out set	5k CC12M prompts		2k PD12M prompts

Pixel MeanFlow. pMF-L/16 (see App. B.6) is fine-tuned with pMF's own objective, ported from its code (the MeanFlow identity with its forward-mode derivative, x-prediction with the guidance in the target, the auxiliary v-loss, and LPIPS and ConvNeXt-V2 perceptual terms for $t < 0.8$, in pMF's time, where $t = 1$ is pure noise), with one layout per image held fixed across the step's network calls. Training layouts come from the clean image: the dense grid with probability 0.30, otherwise the entropy quadtree at $\delta \sim \mathcal{U}(0, 2)$, with random entropy maps for 10 % of those (7 % of all images). A merged token is the mean of its base-patch embeddings plus a zero-initialised mixer, and each coarse scale has a zero-initialised pixel refiner on each output branch (u, and v in training; 4.77M added parameters in all), so training starts from the released model. AdamW $(0.9, 0.95)$ without weight decay, learning rate 10^{-5} ($8\times$ on the added modules), 500 warm-up steps then constant, gradient clipping at 20, EMA 0.9995. It trains for 64,000 steps at a global batch of 91 (7 GPUs $\times$ 13; 52 on 4 GPUs for steps 40,000–58,000), 5.1M images, with the learning rate cooled linearly to zero over the last 8,000. Sampling keeps the released guidance, 7.0 on $t \in [0.2, 0.7]$.

C.5 EVALUATION PROTOCOL

Methods compared. The *released model* is the parent as shipped, at its dense grid; the *base model* is the released weights run at the same budget through our sampler, without training. ToMe merges tokens by bipartite soft matching on the image keys before RoPE, averaged over heads (size-weighted, with proportional attention and the same r in every block; condition tokens, text or JiT's in-context class tokens, are never merged). FeatSim makes one similarity merge down to B tokens early in the network (block 3 on MiniT2I). Both wrap the released weights, keep the sampler on the dense grid, and on MiniT2I write the merged tokens back to the dense grid for the last two blocks. RTI runs its released adapter (40k steps), which exists for MiniT2I-B and MiniT2I-L only. On PixelDiT ours is compared with the base model. Every method of a parent shares the prompts or labels, the noise, the solver, the steps, the guidance and the attention arithmetic.

Class-conditional. FID and Inception score follow JiT: Inception-v3 from torch-fidelity on uint8 images at native resolution, against JiT's ImageNet-1k (Deng et al., 2009) training-set statistics at 256^2 and 512^2; the Inception score uses 10 splits. FID-50k uses 50 images per class; the ablations (see App. D) use FID-10k, 10 per class. Each FID is a single run.

Text-to-image. GenEval (553 prompts, 4 images each) counts an image as correct only if every element of its prompt is rendered, averaged over six tasks; DPG-Bench (1,065 prompts, 4 images each) is the mean mPLUG-large answer; CLIP score (ViT-L/14), PickScore and ImageReward are means over the 1,632 PartiPrompts, one image each. Held-out FID, used at the dense grid, compares 5k generated with 5k held-out real images for CC12M (Changpinyo et al., 2021) and 2k with 2k for PD12M, at native resolution, with 50 steps at guidance 3.0 (MiniT2I-B), 5.0 (MiniT2I-L) and 2.75 (PixelDiT).

Paired metrics. Each image at a reduced budget is scored against the same method's image at the dense grid, from the same prompt or label and noise, by PSNR, SSIM, LPIPS and the cosine similarity of CLIP and DINOv2 embeddings. On MiniT2I-L each cell is the mean over the 553 GenEval prompts, one image each, with its 95 % interval; on JiT it is the mean over 2,000 class-balanced images at guidance 3.0 and 50 steps, and PSNR is also split between the pixels the layout merged into coarse tokens and those it kept.

C.6 User study and VLM-as-Judge

Images. Both studies use the images of Figure 8(b): MiniT2I-L/16 at 512^2, five methods at ten budgets, and the released model. Image j comes from prompt j and noise seeded by j, so two images in one comparison differ by method and budget alone. The prompts are 100 GenEval prompts drawn at random (22 colours, 21 two-object, 20 colour-attribution, 14 position, 13 single-object, 10 counting); the user study uses 20 of them, chosen from the released model's images alone. Both studies are scored alike: "no visible difference" counts as 5 %, a tie as one half, and 95 % intervals are bootstrapped over prompts. Table 20 gives every number of Figure 9, with the intervals in brackets.

User study. Nine anonymous online forms of 20 questions, one image per question, in random order; each form counts its first 50 participants. The instructions ask for faithfulness only, not to prefer a sharper or prettier image that has changed what the reference shows.

User study: where the drop is first seen (Figure 9(a)). Five forms, one per method, show the method's own image at the dense grid above its budgets from 90 to 10 %, and ask for the first budget with a visible difference. Figure 39 shows the start of one of them and its first question.

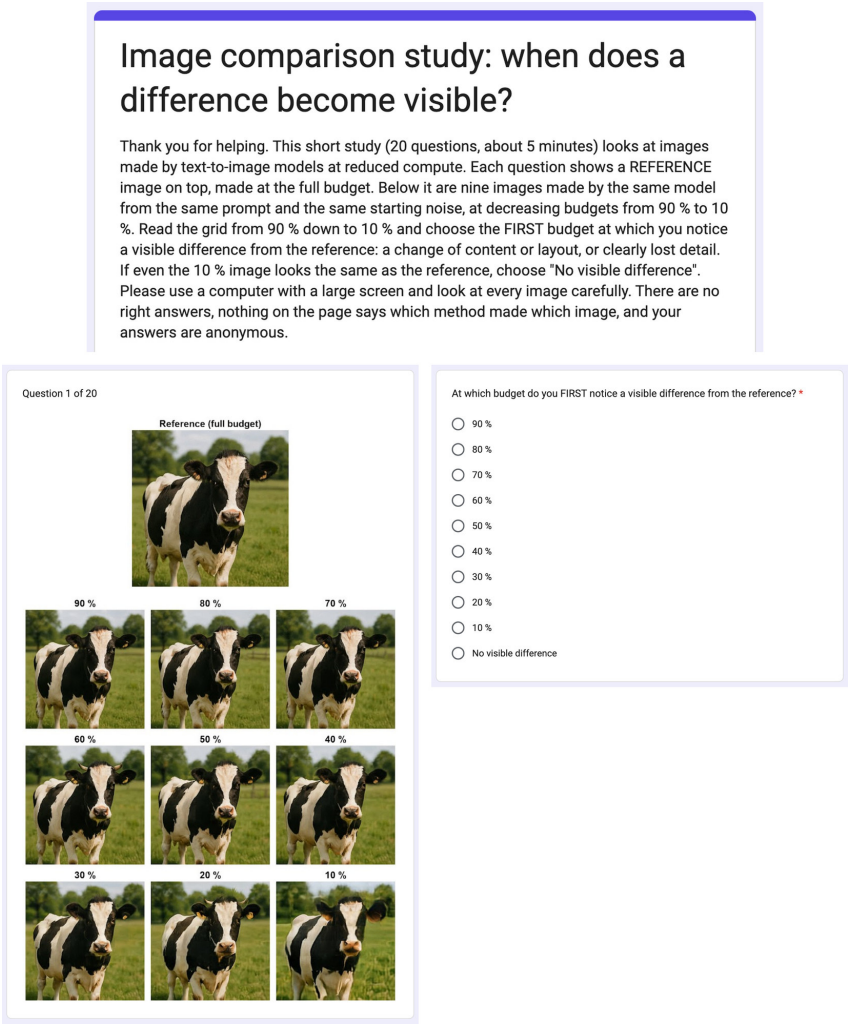

Figure 39: **User-study form on where a difference is first seen: its start and first question.**

User study: which is more faithful (Figure 9(b)). Four forms, one per budget (80, 60, 40 and 20 %), show the released image above ours and RTI as A and B, ours on the left in half the questions,

and ask which is more faithful, or "equally faithful". Figure 40 shows the start of one of them and its first question.

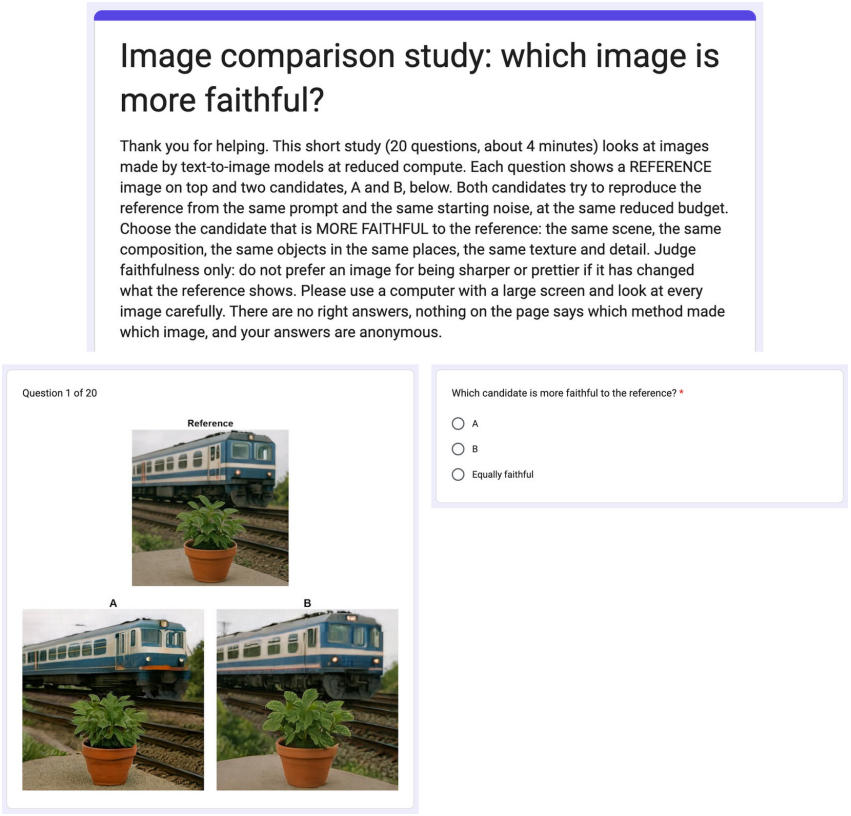

Figure 40: **User-study form on which image is more faithful: its start and first question.**

User study: reading the numbers. Participants chose A in 48 to 54 % of the answers that picked a side, so position did not drive them. Dropping the 17 who gave one answer to 18 or more of their 20 questions moves no cut by more than 2.4 points and no share at all.

VLM-as-Judge. The judge is Gemini 3.8 Flash at temperature 0, thinking level low, one image set per call, answering with a single number or letter.

VLM-as-Judge: where the drop is first seen (Figure 9(a)). The judge sees a method's own image at the dense grid, then one picture tiling its ten budgets in order, after this text:

> The first image is a REFERENCE, generated by this same method at its full token budget from the prompt: "⟨*prompt*⟩"
>
> The second image is a grid of ten images from the same prompt and the same starting noise, generated by the SAME method at decreasing token budgets. The budgets are printed above each tile and run from 100% (top left) down to 10% (bottom right), in order; the 100% tile is the reference itself.
>
> Reading the grid in order from 100% downwards, at which budget do you FIRST notice a visible difference from the reference (a change of content, layout, or clearly lost detail)? If even the 10% tile is indistinguishable from the reference, answer 0.
>
> Reply with exactly one number from this list and nothing else: 100, 90, 80, 70, 60, 50, 40, 30, 20, 10, 0.

With the method's own dense image as reference, the answer is what the budget costs, not what the method already changed at 100 %. One call per method and prompt: 500 judgments.

VLM-as-Judge: which is more faithful (Figure 9(b)). The judge sees the released image, then ours and one baseline at the same budget, after this text:

> The first image is a REFERENCE, generated by an image model at its full token budget from the prompt: "⟨*prompt*⟩"
>
> Two candidate images follow, A then B. Each was generated by a different method at a reduced token budget from the same prompt and the same starting noise, trying to reproduce the reference.
>
> Which candidate is MORE FAITHFUL to the reference: the same scene, the same composition, the same objects in the same places, the same texture and detail? Judge faithfulness to the reference only. Do not reward an image for being sharper or more pleasing than the reference if it has changed what the reference shows.
>
> Reply with exactly one character and nothing else: A, B, or T if the two are equally faithful.

The three images are resized and JPEG-encoded identically. Each pair is judged in both left/right orders, as independent calls: 7,193 judgments over four baselines, nine budgets and 100 prompts.

VLM-as-Judge: reading the numbers. Position bias, the share of pairs answered on the same side in both orders, is 8.7 %, below the 15 % rule of thumb. The judge never answered T.

Table 20: **User study and VLM-as-Judge, MiniT2I-L/16 at 512^2: the numbers of Figure 9.**

cut before a difference is seen (%)	Base model	ToMe	FeatSim	RTI	BudgetPix
people, 20 prompts	30.3 [29.2, 31.4]	32.5 [30.1, 35.1]	23.3 [21.5, 25.3]	38.0 [31.8, 44.0]	**51.8** [48.0, 55.7]
judge, the same 20 prompts	18.5 [14.5, 22.5]	20.0 [16.5, 24.0]	12.5 [11.0, 14.5]	24.0 [14.5, 35.0]	**43.5** [35.0, 51.0]
judge, all 100 prompts	15.4 [14.1, 16.9]	15.9 [14.3, 17.5]	11.7 [11.0, 12.4]	21.1 [16.9, 25.8]	**36.2** [31.9, 40.4]

prefers ours to RTI (%)	80%	60%	40%	20%
people, 20 prompts	63.5 [52.7, 74.7]	63.4 [52.2, 74.2]	57.2 [45.3, 69.5]	62.3 [52.8, 72.2]
answers for ours / RTI / equal	54 / 27 / 19	56 / 29 / 16	50 / 36 / 14	54 / 29 / 17
judge, the same 20 prompts	77.5 [60.0, 95.0]	72.5 [55.0, 87.5]	50.0 [30.0, 70.0]	75.0 [57.5, 90.0]
judge, all 100 prompts	75.5 [67.5, 83.0]	69.0 [61.0, 76.5]	54.5 [46.0, 63.0]	61.5 [53.5, 69.5]

C.7 WALL-CLOCK MEASUREMENT

Timing. Every cell was timed on one H100, with the overhead of the measurement rather than of the method removed: per-forward GPU-to-CPU synchronisations, layout work repeated on every forward, and kernel-launch cost, with one CUDA graph per layout. The images are bit-identical to those of each method's released code; ours and the base model also resample merged tokens with two fp32 matrix products (identical images on MiniT2I, $\sim$55 dB PSNR on JiT). Speed-up is over the released model at its dense grid, timed the same way. Table 21 lists the conditions, which every method of a parent shares.

Two checks. ToMe, FeatSim and RTI are not deterministic: two runs of their released code agree only to 20, 27 and 34 dB PSNR, so for them the images match up to this noise. PixelDiT does not replay correctly as a CUDA graph (23 dB against its own output), so it is timed without graphs, with bit-identical images.

Table 21: **Conditions of the wall-clock measurements, every parent.**

setting	JiT-B/32, L/32 (512^2)	JiT-B/16, L/16 (256^2)	MiniT2I-B/16	MiniT2I-L/16	PixelDiT-T2I
resolution	512×512	256×256	512×512		1024×1024
dense grid	256 tokens (32 px)	256 tokens (16 px)	1024 tokens (16 px)		4096 tokens (16 px)
token scales of the layout	32 / 64 / 128 px	16 / 32 / 64 px			
budgets 100, 90, 80, 70, 60, 50, 25 %	256, 230, 205, 179, 154, 128, 64 tokens		1024, 922, 819, 717, 614, 512, 256 tokens		4096, 3686, 3277, 2867, 2458, 2048, 1024 tokens
conditioning	class label, random		text, pre-encoded		
batch (images per sampler call)	50		32	20	8
warm-up, discarded	one call per cell for the released model and BudgetPix, one per process for the others				
timed calls per cell	released, BudgetPix, all of JiT-B/32: 3 (median); others: 2	released, BudgetPix: 3 (median); others: 2	released, BudgetPix: 2 (median); others: 1		released, BudgetPix: 2 (median); base model: 1
solver, steps	Heun, 50 (last step Euler)		Euler, 100		DPM-Solver++(2M), 50
guidance scale	3.0 on $t \in [0.1, 1]$		3.0	5.0	4.5
network forwards per step	4 (2 Heun stages $\times$ cond./uncond.)		2 (cond., uncond.)		
dense warm-up steps n_w	5, +1 forward (21 of 199 forwards)	10, +1 forward (41 of 199 forwards)	5, +1 forward for the $\hat{x}$ estimate		5, +1 forward
layout	built once, then frozen				
numerics	bf16 weights, bf16 autocast		bf16 weights, bf16 autocast; RTI in pure bf16, as it ships		bf16 weights, bf16 autocast
attention arithmetic (all methods)	the model's own		released: einsum + softmax		fused SDPA, masked when a reduced budget pads the layout
methods timed	released, base model, BudgetPix, ToMe, FeatSim		released, base model, BudgetPix, ToMe, FeatSim, RTI		released, base model, BudgetPix

Measurement protocol.

- Hardware: one machine with 8$\times$ NVIDIA H100 80GB (driver 590.48), PyTorch 2.5.1 with CUDA 12.4, one Python environment for every method, and one cell per GPU.
- Timed: the sampler call only, from noise to the final image tensor, with `torch.cuda.synchronize()` before and after it. Model loading, text encoding (prompts are pre-encoded), scoring and image writing are off the clock.
- Warm-up: the first sampler call is discarded, once per cell for the released model and ours and once per process for the others (autotuning is off). Repeats agreed within about 0.5 %.
- Released model, the speed-up denominator: its own forward at the dense grid, in the same sampler loop. The base model's 100 % cell shows what our wrapper costs: 2 % faster to 6 % slower than the released model.

- Ours: the first n_w steps on the dense grid (10 for JiT at 256^2, 5 otherwise, as in the quality runs), one forward for $\hat{x}$, then the layout bisected to the budget, kept to the end.
- RTI (MiniT2I only) is timed around its own sampler, with its fused attention routed through the released einsum and softmax that every other method runs; with its fused kernel it measured $1.8\times$ faster than the released model with no token removed.
- The 90 % cells were timed at 922 tokens on MiniT2I and 3686 on PixelDiT; the quality runs use 911 and 3645. Realised token counts never exceed the budget and fall at most a few tokens short of it.

D ABLATIONS ON JIT-B

Each ablation changes one factor at a time on JiT-B, on the development model (see App. D.1–D.5) or on the final JiT-B/32 (see App. D.6–D.9). All are FID-10k (see App. C.5), not comparable with the FID-50k of the main tables. The reported results use none of the budget-tuned schedules or guidance scales found here. The component ablation is Table 2.

D.1 REFERENCE RUNS (DEVELOPMENT JIT-B)

Table 22 is the reference for the development runs (see App. D.2–D.4): the 50-epoch model on JiT-B/32 with the default sampler (uniform schedule, 50 steps, guidance 3.0). JiT-B/16 had no adapted model yet, so only its base model is shown.

Table 22: **Reference runs, JiT-B: FID-10k.**

parent	method	100%	90%	80%	70%	60%	50%	40%	30%
	tokens	*256*	*230*	*205*	*179*	*154*	*128*	*102*	*77*
JiT-B/32, 512^2	Base model	6.53	7.44	9.45	14.96	27.61	63.91	118.04	156.99
	BudgetPix, 50 epochs	**6.47**	**6.93**	**7.36**	**8.33**	**9.36**	**11.18**	**13.89**	**17.33**
JiT-B/16, 256^2	Base model	6.02	6.67	8.30	13.26	26.34	67.38	116.44	152.40

D.2 A PARALLEL PIXEL-STREAM DECODER (DEVELOPMENT JIT-B/32)

Before the refiner, the decoder was a pixel stream running in parallel with the token stream. Table 23 follows it from 10 to 50 epochs.

Table 23: **Pixel-stream decoder across training length, JiT-B/32 at 512^2: FID-10k.**

parent	training	100%	90%	80%	70%	60%	50%	40%	30%
	tokens	*256*	*230*	*205*	*179*	*154*	*128*	*102*	*77*
JiT-B/32, 512^2	10 epochs	7.48	8.16	9.33	11.30	13.87	18.19	23.95	29.94
	20 epochs	7.45	8.15	9.10	10.91	13.08	16.27	21.80	28.29
	30 epochs	7.40	8.05	9.05	10.72	12.43	15.95	20.50	26.78
	50 epochs	7.33	8.01	8.94	10.36	12.15	15.17	19.41	24.96

D.3 SAMPLING SCHEDULE AND STEP COUNT (DEVELOPMENT JIT-B/32)

Table 24 varies the timestep schedule and the step count at 128 tokens (half the grid), guidance 6.0.

Table 24: **Timestep schedule against step count, JiT-B/32 at 512^2: FID-10k.**

parent	schedule	10 steps	15 steps	25 steps	50 steps
JiT-B/32, 512^2	cosine	10.02	9.15	9.01	8.82
	shift-3	11.71	11.21	10.61	10.10
	uniform	12.14	11.88	10.42	9.10

D.4 GUIDANCE SCALE ALONG THE BUDGET (DEVELOPMENT JIT-B/32)

Table 25 sweeps guidance from 3.0 to 6.0 along the budget, with the cosine schedule at 50 steps; the last row is the default sampler (uniform schedule, guidance 3.0).

Table 25: **Guidance scale along the budget, JiT-B/32 at 512^2: FID-10k.**

parent	sampler	90%	80%	70%	60%	40%
	tokens	*230*	*205*	*179*	*154*	*102*
JiT-B/32, 512^2	cosine, guidance 3.0	7.19	7.75	8.75	10.17	14.50
	cosine, guidance 4.0	7.21	7.31	7.62	8.37	11.14
	cosine, guidance 5.0	8.03	7.87	7.92	8.17	9.96
	cosine, guidance 6.0	8.93	8.55	8.46	8.45	9.66
	uniform, guidance 3.0 (default)	6.93	7.36	8.33	9.36	13.89

D.5 Upper bound on a better decoder (development JiT-B/32)

Two arms change only the decoded output of the 50-epoch JiT-B/32 (Table 22). Oracle-fine gives the coarse regions their ground-truth fine detail, an upper bound on any better decoder; downsample-lanczos resamples every output to 256^2 and back, removing all finer detail.

Table 26: **Upper bound on a better decoder, JiT-B/32 at 512^2: FID-10k.**

parent	arm	80%	50%	30%
	tokens	*205*	*128*	*77*
	guidance	*3.0*	*6.0*	*6.0*
JiT-B/32, 512^2	reference	7.36	9.10	11.69
	downsample-lanczos	7.42	8.92	11.14
	oracle-fine	7.38	9.34	11.56

D.6 Layout and sampler settings (final JiT-B/32)

Table 27 varies one setting at a time on the final JiT-B/32 (512^2, 100 epochs), from the sampler of every reported result (*default*: 50 Heun steps, guidance 3.0, five dense warm-up steps, a layout rebuilt every five steps), on the same 10k images and noise. Random placement is the entropy quadtree on permuted entropy maps, which keeps the token count; the oracle is the quadtree of a 50-step dense draft from the same noise. The noise-seed rows repeat the default with shifted noise.

Table 28 varies guidance and the timestep schedule the same way. Time shift 3.0 spends more steps at high noise, the Karras schedule more at low noise, cosine more at both ends, and two-phase only 8 of the 50 at $t \geq 0.7$; 50 Euler steps make half the network evaluations of the others.

Table 27: **Layout and sampler settings, JiT-B/32 at 512^2: FID-10k.**

setting	value	50% *128*	25% *64*
layout signal	entropy quadtree (default)	**10.99**	19.79
	random placement	19.51	23.41
	oracle, 50-step draft	11.30	**19.58**
dense warm-up steps	0	12.44	20.41
	2	11.20	20.33
	5 (default)	10.99	19.79
	10	10.36	18.09
	20	**9.68**	**15.73**
re-layout interval	every step	**10.96**	19.76
	every 5 steps (default)	10.99	19.79
	every 10 steps	11.00	**19.70**
	every 25 steps	11.04	19.73
	once, after the warm-up	**10.96**	19.76
Heun steps	25	**10.88**	**18.02**
	50 (default)	10.99	19.79
	100	10.92	19.92
noise seed	0 (default)	10.99	19.79
	1	10.81	19.36
	2	10.74	19.38

Table 28: **Guidance scale and timestep schedule, JiT-B/32 at 512^2: FID-10k.**

setting	value	50% *128*	25% *64*
guidance	2.0	19.09	31.73
	2.5	13.77	24.27
	3.0 (default)	10.99	19.79
	4.0	8.96	15.13
	5.0	**8.67**	13.50
	6.0	8.91	**13.20**
	7.0	9.30	13.67
schedule	uniform, Heun (default)	10.99	19.79
	time shift 3.0	14.54	24.01
	Karras, $\rho = 7$	11.50	**17.72**
	cosine	11.65	20.40
	two-phase	11.13	19.48
	Euler, 100 steps	**10.72**	19.01
	Euler, 50 steps	11.03	18.46

D.7 Sensitivity to the entropy thresholds (final JiT-B/32)

The thresholds $\tau_{2p} = 6.0$ and $\tau_{4p} = 4.0$ bits are taken from APT (Choudhury et al., 2026), not tuned. Table 29 shifts τ_{2p} by $+\kappa$ and τ_{4p} by $-\kappa$ and then bisects δ to the budget, so the count stays near 128 tokens (124 at $\kappa = -1.5$) and mainly the split between $2p$ and $4p$ tokens moves; a negative κ favours $4p$. The model, sampler and images are those of Table 27.

Table 29: **Entropy-threshold skew, JiT-B/32 at 512^2: FID-10k and Inception score at 128 tokens.**

skew κ	-1.5	-1	-0.5	0 (default)	$+0.5$	$+1$	$+1.5$
FID-10k ↓	15.79	13.02	11.42	10.99	**10.95**	**10.95**	10.97
Inception score ↑	152	165	173	175	176	**177**	**177**

D.8 Layout evolution during sampling (final JiT-B/32)

Eight ImageNet classes from fixed noise, 50 Heun steps at guidance 3.0 with the layout rebuilt at every step, at 128 and 64 tokens, with no warm-up (the first layout is read off the pure-noise estimate) or with the default five dense steps. Figure 41 shows one sample's estimate and layout at 128 tokens (left; top without warm-up, bottom with it) and, over the eight images, the share of the image whose token size still differs from the final layout (right; grey band: the default's warm-up).

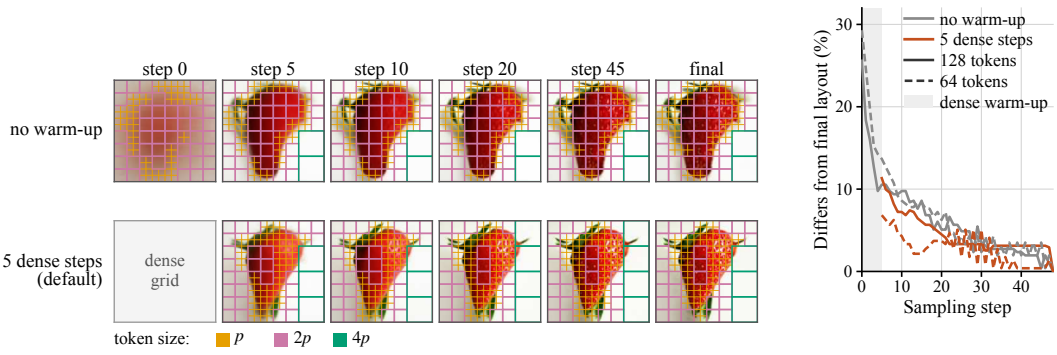

Figure 41: **Layout evolution during sampling, JiT-B/32 at 512^2.**

D.9 Pooled budget across a batch (final JiT-B/32)

Every reported result gives each image the same budget. The pooled arm instead sets one offset per batch, so the batch meets the budget on average and each image takes what its content asks for (16 to 253 tokens at a mean of 128); the fixed arm is the default. Table 30 gives FID-10k as in Table 27,

and PSNR (mean, with 10th / 90th percentiles) and LPIPS of 2,000 images against the dense image of the same noise, with the base model at the same token count as a control. Figure 42 shows eight samples at a mean of 128 tokens, ordered by the tokens the pool gave them (above each column): the dense image, the pooled and fixed arms, and the base model.

Table 30: **Pooled against fixed budget, JiT-B/32 at 512^2: FID-10k and paired fidelity.**

metric	arm	70% *179*	50% *128*	25% *64*
FID-10k ↓	fixed (default)	8.20	10.99	**19.79**
	pooled	**7.79**	**10.82**	20.39
PSNR ↑	fixed (default)	24.57 (20.86 / 28.40)	23.22 (19.83 / 26.84)	22.14 (18.96 / 25.59)
	pooled	26.35 (22.05 / 29.54)	23.49 (20.47 / 26.56)	22.08 (19.18 / 25.27)
	base model	20.58 (17.44 / 24.17)	18.68 (15.95 / 21.59)	16.02 (13.75 / 18.52)
LPIPS ↓	fixed (default)	0.308	0.379	**0.450**
	pooled	**0.271**	**0.370**	0.457
	base model	0.487	0.639	0.877

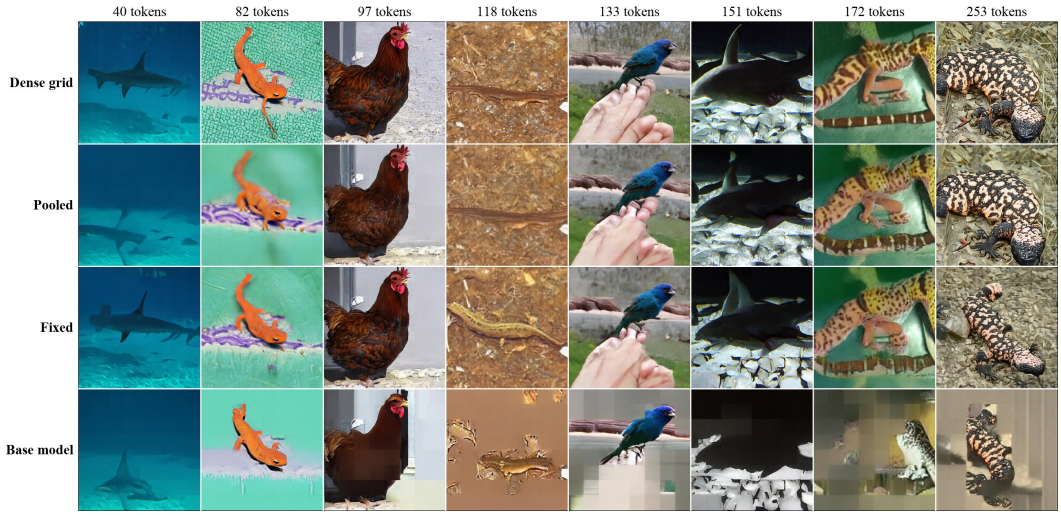

Figure 42: **Pooled budget, JiT-B/32 at 512^2, a mean of 128 tokens.**