STRUCTURE OVER PIXELS: LEARNING VARIABLE-LENGTH VISUAL PROGRAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Discrete visual tokenizers map images to ordered sequences of tokens, providing a natural representation for structural scene descriptions. Most use a fixed sequence length, while adaptive methods often require post-hoc search or choose among a small set of rates that control the length. We propose STROP, a discrete tokenizer that learns both a visual program and its image-dependent active length. A length head is trained with a four-phase curriculum using local rate–distortion probes against frozen DINOv3 features, then predicts the active prefix in a single forward pass. At a matched rate of about 250 nominal bits per crop, the adaptive model improves segmentation over a separately trained fixed-length baseline on four benchmarks (by 1.6–3.1 mIoU), and it also beats a fixed $K=32$ baseline that uses more bits. STROP programs also yield higher segmentation mIoU than FlexTok, One-D-Piece, and ALIT at similar or higher rates, under the same readout architecture and training protocol. STROP therefore learns useful per-image sequence lengths without post-hoc search or a predefined set of compression rates.

1 INTRODUCTION

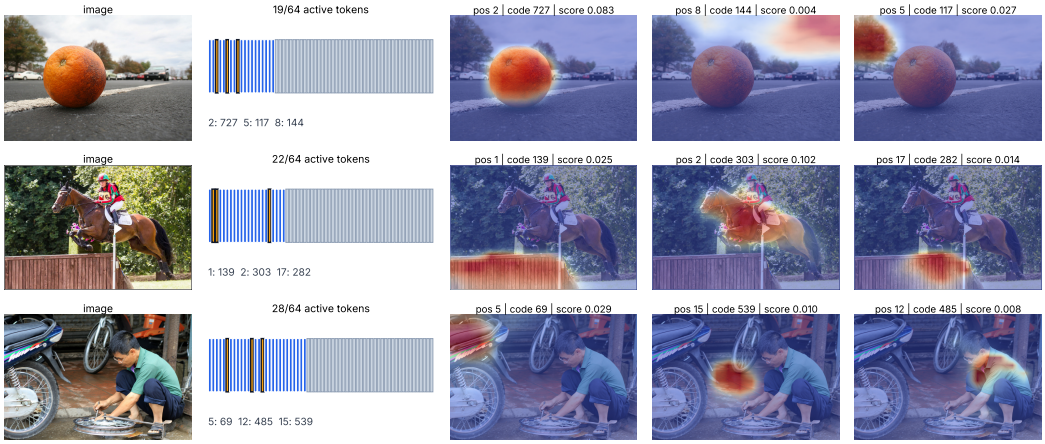


Figure 1: **Images as variable-length discrete programs.** Each image is encoded as a sequence of discrete tokens whose length the model chooses per image. For each highlighted position, we erase the token and reinterpret the remaining sequence. The attribution panels show the resulting change in the DINO-aligned patch field. Red regions show that removing a token changes a spatially localized part of the patch field. Heatmaps are normalized per panel; scores are raw attribution magnitudes.

Deep neural networks prove unmatched at most computer vision (CV) tasks, mainly thanks to efficient representation learning. Their learned representations remain however verbose, distributed, convoluted, and thus hard to inspect and validate in use cases that require robust scene interpretation, like robotics, planning, autonomous driving or medical diagnosis, which require models that reason about scenes in terms of objects, their properties, and their spatial relations. Yet such structural visual descriptions remain hard to learn, especially when supervision is scarce. Object-centric learning (Lo-

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.

catello et al., 2020; Singh et al., 2022; Seitzer et al., 2023) and neural program induction (Reed & de Freitas, 2016) both pursue such descriptions.

Discrete visual tokenizers (DVTs) (Jia et al., 2025) offer a promising framework for forming structural descriptions, since they already operate with quasi-symbols: a scene is mapped to an ordered sequence of discrete tokens from a fixed vocabulary, which can be usually interpreted back into the spatial feature field. The model learns to generate such sequences without object labels, hand-designed predicates, or a-priori helper structures like scene graphs. Enforcing such latent representations facilitates inspection in search of signs of code reuse, symbol grounding, or compositionality.

Most DVTs operate at constant sequence length. VQ-VAE and VQGAN-style models produce a fixed spatial grid of tokens, and recent 1D tokenizers replace that grid with a fixed number of latent slots (van den Oord et al., 2017; Esser et al., 2021; Yu et al., 2024b). A smooth sky patch and a crowded street are mapped to different sequences, but both receive the same length (sometimes referred to as token *budget*). This sits uneasily with cognitive postulates like Gestalt principles and Occam's razor, not to mention the formalisms of minimum description length and Kolmogorov complexity (KC), which ties the complexity of an artifact to the length of the shortest program that generates it (Kolmogorov, 1965; Li & Vitányi, 2008). Uncomputable in general, KC nonetheless makes the right demand: adding an object to a scene should come at a cost. The sequence length should correlate with the 'amount' of visual structure in the observed scene – this is the first step, if not a prerequisite, for forming principled structural representations.

In this study, we call a sequence of tokens a *program* in a deliberately thin sense. The codebook acts as a fixed instruction set, but the interpreter learns what each instruction does, and no grammar restricts how instructions combine. Neural program induction, by contrast, usually starts from predefined operations (Reed & de Freitas, 2016). *Incremental interpretability* is our only requirement, i.e. every prefix of a program is a valid input to the same interpreter. Some researchers train decoders to tolerate prefixes; others use inference-time search, recurrent allocation, routing among fixed rates, or information-theoretic scores of input compressibility (Bachmann et al., 2025; Miwa et al., 2025; Yan et al., 2025; Duggal et al., 2025a; Shen et al., 2025; Ye et al., 2025). Closest to us is KARL (Duggal et al., 2025b), which predicts a per-image length in one forward pass and reads it as an estimate of complexity, though conditioned on a pixel-space reconstruction quality supplied at inference rather than on the tokenizer's own economy. These results suggest that length control should be tightly linked with visual tokenization, rather than being delegated to some external mechanism.

Another prevailing characteristic in DVT research is the emphasis on fidelity of reconstruction. Indeed, previous works suggest that training DVTs with pixel-level reconstruction loss serves compression well. However, it can waste budget on capturing textural characteristics that carry little semantic or geometric structure. Feature-distillation tokenizers like BEiT v2 (Peng et al., 2022), DINOSAUR (Seitzer et al., 2023), REPA (Yu et al., 2025), and MAETok (Chen et al., 2025a) show that guiding training with teacher features and representation-aligned objectives, rather than just reconstruction loss, can produce useful latent spaces and object-sensitive structure.

Motivated by the above findings, we propose STROP[1], a DVT architecture built from two transformers. A generator writes a sequence of tokens from a visual vocabulary, and an interpreter reads it back into a patch feature field. A learned length head decides how much of the sequence to expose. Each token costs a fixed number of bits, so this length sets the *rate* of the encoding, which the model trades against the distortion of the interpreted features. This gives three concrete contributions: **(i) Variable-length visual programs.** We supervise per-image program length with local rate-distortion probes of the trained tokenizer and use a prefix curriculum – random truncation first, predicted prefixes later. **(ii) A simple generator-interpreter tokenizer.** Two encoder-only transformers with learned queries write and read causally ordered visual tokens. DINOv3 features provide the only supervision. **(iii) Evaluation of representation and structure.** We compare adaptive programs with fixed-length tokenizers and public adaptive tokenizers, ablate the training design, measure downstream dense prediction and analyze where tokens act in the image and how program length follows scene content.

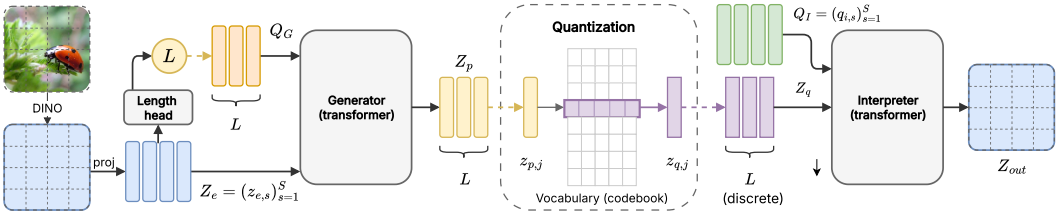


Figure 2: STROP architecture overview.

2 The proposed STROP architecture

The architecture is shown in Figure 2. Given an input image $x \in \mathbb{R}^{3\times H\times H}$, we seek a compact discrete program $Z_q = (z_{q_1}, \ldots, z_{q_L})$ from which a DINO-aligned feature field can be recovered, where L varies by image and $\mathcal{V} = \{v_i \in \mathbb{R}^{d_c}\}_{i=1}^{V}$ is a visual vocabulary of size V. The model comprises a frozen *visual encoder* producing patch-wise features, a *program generator* G that maps those features into a sequence of tokens, a vector-quantization bottleneck, and an *interpreter* I that expands the program back to a spatial feature grid. An auxiliary *length head* predicts L from the encoder features, enabling variable-length programs at inference without autoregressive sampling.

Encoder. We extract token-level features with a frozen DINOv3 ViT encoder (Siméoni et al., 2025). For patch size p, it produces $S = (H/p)^2$ tokens. A layer norm followed by a linear projection maps every token from d_{enc} to a shared model dimension d, yielding source features $Z_e \in \mathbb{R}^{S\times d}$.

Program generator. To map the patch-based visual features into a sequence of tokens describing the scene, we adopt an encoder-only transformer that processes Z_e with an appended sequence Q_G of K learnable query tokens. The concatenated $[Z_e;\, Q_G] \in \mathbb{R}^{(S+K)\times d}$ is processed by standard transformer encoder layers. A causal mask prevents the kth query token from attending to query tokens with $k' > k$, and the source tokens from attending to any query, ensuring that no future-query information leaks through intermediate source representations. After encoding, only the K query outputs are retained as the pre-quantized, raw *program* $Z_p \in \mathbb{R}^{K\times d}$.

Vector quantization. A linear projection maps each token of Z_p into the codebook space $\mathbb{R}^{K\times d_c}$ ($d_c \ll d$), where an V-entry vocabulary assigns discrete indices via nearest-neighbor lookup using Euclidean distance. The quantized tokens are projected back to d dimensions, yielding $Z_q \in \mathbb{R}^{K\times d}$.

Length head. A lightweight MLP h_ϕ predicts the program length directly from the encoder features: $\hat{L} = \lceil K\sigma\big(h_\phi(\text{pool}(Z_e))\big)\rceil$, where pool denotes mean pooling over the spatial dimension and σ is the sigmoid function. At inference, $\hat{L}$ is rounded to the nearest integer. Positions $k > \hat{L}$ in Z_q are masked from the interpreter. Training of this head is detailed in Section 2.2.

Interpreter. The interpreter mirrors the generator in mapping Z_q truncated to $\hat{L}$ tokens back to patch-level feature tokens. Padding is used only in training, where selected prefixes are padded for batching and masked so that the interpreter cannot attend to padded positions; no padded or zeroed program positions are passed at inference. The interpreter updates $(H/p)^2$ learnable grid queries by cross-attending to the retained tokens, treating the selected program as the key-value memory. The resulting grid-query outputs are projected with a 1×1 convolution and reshaped into a feature map $Z_{out} \in \mathbb{R}^{(H/p)\times(H/p)\times d}$ to obtain the patch-feature field.

Relation to feature autoencoders. STROP reconstructs frozen DINO features, so it can be viewed as a discrete feature autoencoder. A standard feature autoencoder uses a fixed-size latent representation. STROP instead produces a causally ordered sequence and predicts how much of it to retain. Any prefix is a valid input to the interpreter, and the predicted length controls how much information passes through the bottleneck. The variable-length bottleneck and the vocabulary are learned together.

[1]Our code is available at: https://anonymous.4open.science/r/structure-over-pixels.

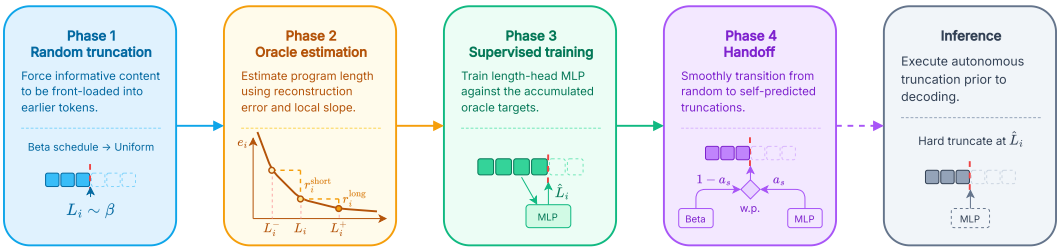


Figure 3: Training curriculum for adaptive truncation. **Phase 1:** Random Beta truncation encourages front-loading of information. **Phase 2:** Oracle lengths are estimated from the feature reconstruction error and the local slope of the rate–distortion curve. **Phase 3:** An MLP length head is supervised on these oracle targets. **Phase 4:** A probabilistic handoff smoothly transitions from random to MLP-predicted lengths. **Inference:** Hard truncation relies solely on the learned MLP. Details on the curriculum can be found in Appendix A.

2.1 Training objective

The loss function is a composite of the following objectives.
Latent alignment $\mathcal{L}_{\text{lat}}$. Given the interpreter-generated patches Z_{out} and the frozen DINO patches Z_e, we use $\mathcal{L}_{\text{lat}} = 1 - \cos(Z_{out}, Z_e) + \text{MSE}(Z_{out}, Z_e)$, where the cosine term preserves the teacher-feature geometry and MSE stabilizes scale.
Commitment loss $\mathcal{L}_{\text{commit}}$. The standard VQ regularization penalty $\mathcal{L}_{\text{commit}} = \lambda_{\text{q}} \|Z_p - \text{sg}[Z_q]\|^2$ keeps pre-quantized embeddings close to their selected codebook entries.
Diversity loss $\mathcal{L}_{\text{div}}$. The variance and covariance terms of VICReg (Bardes et al., 2022), applied to the pre-quantization program vectors Z_p: a hinge keeps each embedding dimension's variance from collapsing, and an off-diagonal covariance penalty decorrelates dimensions (c=0.04), together giving the codebook distinct directions to specialize into. We omit VICReg's invariance term, which requires paired views we do not have. Warmed up over 40k steps with weight λ_{div}=0.3.
The full training objective is thus: $\mathcal{L} = \mathcal{L}_{\text{lat}} + \mathcal{L}_{\text{commit}} + \mathcal{L}_{\text{div}} + \mathcal{L}_{\text{len}}$, where $\mathcal{L}_{\text{len}}$ (Sec. 2.2) is activated in Phase 3 of the curriculum. The weights for each term are given in Table 5.
Codebook design. The vocabulary consists of V=1024 entries of d_c=16 dimensions, updated via exponential moving average (EMA, γ=0.95) with ℓ_2-normalized codes. The low code dimension combined with ℓ_2 normalization places all entries on a hypersphere, stabilizing nearest-neighbor assignment and preventing dead codes from drifting far away from the active manifold.

2.2 Adaptive program-length curriculum

The generator produces K quantized tokens for each sample x. The interpreter uses only a prefix of length $L_i \in \{1, \ldots, K\}$ and masks positions $k > L_i$. We train the length head with the four-phase curriculum shown in Fig. 3. Appendix A provides more details on the schedule.
Phase 1: Random truncation. We train the model on randomly truncated programs. The length distribution favors longer prefixes, and the generator learns to place useful information in the early tokens. The length head remains inactive.
Phase 2: Oracle target estimation. We continue random truncation and estimate a target length for each sample from its feature reconstruction error and from the local rate–distortion slope, using slightly shorter and longer prefixes. These targets are collected but do not yet affect truncation.
Phase 3: Supervised length-head training. We train the length head on the targets estimated in Phase 2. Prefix lengths are still sampled independently of its predictions.
Phase 4: Handoff to predicted lengths. We gradually replace the sampled lengths with the predictions from the length head. After the handoff, all programs use predicted lengths, while the head continues to receive supervision. Appendix B analyzes sensitivity to the handoff timing. At inference, we round $\hat{L}_i$, clip it to $[1, K]$, and pass only the resulting prefix to the interpreter.

3 Related work

Discrete visual tokenizers. VQ-VAE introduced discrete visual representations based on a continuous encoder and learned codebook (van den Oord et al., 2017; Razavi et al., 2019), while VQGAN added

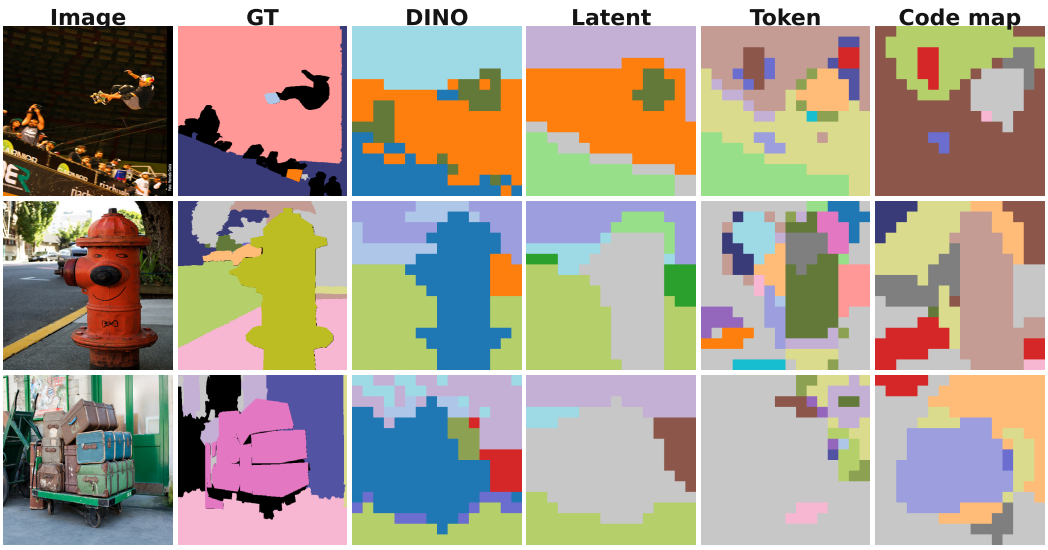


Figure 4: Unsupervised readouts on COCO-Stuff-27. DINO, Latent, and Token show k-means regions obtained from DINO teacher patch features, STROP latent patch features, and token-erasure attribution vectors, respectively. Colors denote arbitrary cluster IDs and are not semantic labels. Code map shows a many-to-one mapping from program code IDs to semantic classes.

adversarial and perceptual losses (Esser et al., 2021). Later work developed masked and next-scale prediction (Chang et al., 2022; Tian et al., 2024), alternative quantizers (Mentzer et al., 2024; Lee et al., 2022), and a more stable straight-through estimator (Fifty et al., 2025). LFQ in MAGVIT-v2 replaces the learned codebook with binarized latent dimensions (Yu et al., 2024a); BSQ first normalizes the latents onto a hypersphere (Zhao et al., 2025). Continuous and product-quantized alternatives also shorten autoregressive sequences (Chen et al., 2025b; Li et al., 2024b). These methods retain a fixed token count determined by the spatial grid.

Variable-length tokenization and adaptive computation. TiTok showed that images can be represented by a short sequence of 1D tokens (Yu et al., 2024b). FlexTok trains a rectified-flow decoder with nested dropout (Bachmann et al., 2025), and One-D-Piece applies tail-token dropout to a discrete tokenizer (Miwa et al., 2025). ALIT adds latents through recurrent rollouts (Duggal et al., 2025a); ElasticTok selects the length using a reconstruction-quality threshold (Yan et al., 2025); CAT routes images among three compression ratios (Shen et al., 2025); and InfoTok uses an ELBO-based router (Ye et al., 2025).

KARL predicts per-token halting in one pass, conditioned on a reconstruction-loss target supplied at inference (Duggal et al., 2025b). STROP also predicts length in one pass. Its supervision comes from the local rate–distortion slope measured against frozen teacher features, so inference does not require an external quality target. KARL uses token sets, while STROP a causally ordered prefix.

Reducing tokens inside a fixed encoder. DynamicViT prunes input tokens through predictors inserted at several transformer layers (Rao et al., 2021), while ToMe merges similar tokens without additional training (Bolya et al., 2023). Foveated inference instead reduces the input resolution away from fixation (Harari et al., 2025). These methods remove, merge, or subsample input tokens. STROP produces a sequence of learned codebook entries that is not a subset of the input.

Feature-space reconstruction. I-JEPA predicts continuous, fixed-size spatial representations rather than pixels (Assran et al., 2023). BEiT v2 distills a CLIP teacher into discrete codes (Peng et al., 2022), while RCG learns a diffusion prior over self-supervised representations (Li et al., 2024a). REPA aligns diffusion-transformer states with frozen DINOv2 features (Yu et al., 2025), and REPA-E extends this signal through the VAE tokenizer (Leng et al., 2025). Related tokenizers regularize VAE latents toward vision-foundation-model features (Yao & Wang, 2025), use masked feature prediction (Chen et al., 2025a), or formulate training as latent denoising (Yang et al., 2025). STROP instead distills frozen DINOv3 features into a discrete sequence whose length varies by image.

Table 1: Rate-matched comparison with fixed-length tokenizers. **Segmentation mIoU** (%, ↑) on four benchmarks. All rows use a frozen DINO-S encoder, the STROP-S generator/interpreter and COCO training; each fixed-length baseline is trained separately with K tokens and no length head. Bits are nominal index bits per image: 10 bits per token, plus a 6-bit length field for STROP. The $K{=}25$ baseline is rate-matched to STROP via $K{=}\lfloor \bar{L} + 6/\log_2 V + 0.5 \rfloor$, where $\bar{L}$ is the mean predicted program length on COCO val. mIoU values are means over 3 tokenizer seeds, each averaged over 3 probe seeds; probes are trained on the full training split at $256{\times}256$ for 40k updates. Latent probes observe the interpreter's spatial latent field with a $1{\times}1$ head; program probes process the active program token vectors z_q with a lightweight transformer (3.8M parameters). DINO-S probes the frozen teacher features that STROP learns to reproduce, as a reference without compression. Best values among the $\approx$250-bit tokenizers in bold.

Model	Bits	Representation	VOC	COCO-Stuff-27	ADE20K	Citysc.
Teacher (frozen DINO-S)	—	teacher	80.3	46.2	42.1	56.7
Fixed-length $K{=}25$	250	latent	66.3	39.4	25.0	33.9
		program	59.5	37.7	23.4	35.7
Fixed-length $K{=}32$	320	latent	67.0	39.5	25.8	34.9
		program	60.1	37.9	23.7	35.9
Fixed-length $K{=}64$	640	latent	73.0	43.4	32.3	42.0
		program	67.6	42.2	29.5	42.9
STROP (ours, adaptive)	$\approx$250	latent	**68.6**	**41.2**	**28.0**	36.8
		program	61.5	39.3	25.3	**37.8**

Compositional and object-centric representations. Slot Attention uses iterative competitive attention to form object-centric representations (Locatello et al., 2020). SLATE decodes slots autoregressively over discrete tokens (Singh et al., 2022), and DINOSAUR reconstructs self-supervised features to obtain slots on real images (Seitzer et al., 2023). SlotDiffusion combines slots with latent diffusion (Wu et al., 2023b), while SlotFormer extends object-centric representations to video (Wu et al., 2023a). Wen et al. Wen et al. (2025) impose a PCA-like order on 1D tokens, and NEO learns latent programs executed by a shared transition model (Baek et al., 2026). STROP uses a causally ordered prefix rather than a permutation-invariant set, which makes program length meaningful.

Generator–interpreter framing. Our terminology follows Neural Programmer-Interpreters (Reed & de Freitas, 2016). The architecture uses learned queries to compress and reconstruct spatial representations, as in DETR (Carion et al., 2020), Perceiver (Jaegle et al., 2021), and Perceiver IO (Jaegle et al., 2022). STROP adds a variable-length prefix with end-to-end length supervision.

4 Results

4.1 Experimental setup

We train five model variants by sweeping the visual encoder's (DINOv3 ViT-S/B/L (Siméoni et al., 2025)) and generator/interpreter's capacity (small STROP-S, medium STROP-M). The encoder is always frozen; trainable components comprise the generator, interpreter, quantizer projections, and length head. Unless otherwise stated, the generators produce programs of length $K{=}64$ with 2D patch and learned prefix positional encodings, a vector quantizer with $V{=}1024$ codes of dimension $d_c{=}16$, ℓ_2-normalized code vectors, EMA updates with $\gamma{=}0.95$, and commitment weight $\lambda_{\text{q}}{=}1.0$. STROP-S uses $d_{\text{model}} = 256$, 8 layers, and an FFN width of 1024, while STROP-M uses $d_{\text{model}} = 512$, 10 layers, and an FFN width of 2048. The interpreter mirrors the generator in every variant, sharing d_{model}, depth, number of heads (always 8), and feed-forward width. Table 6 in Appendix D lists all configurations with parameter budgets and training cost. Unless otherwise stated, training engages curriculum learning of the length head (Sec. 2.2, Appendix A), parametrized according to Table 4: Phase 1 lasts 100k and 200k steps for STROP-S and STROP-M respectively, and each of the subsequent three phases lasts 50k steps.

Table 2: Comparison with public adaptive tokenizers at several token budgets. **Program mIoU** (%, ↑) on COCO-Stuff-27 and VOC. Every tokenizer is frozen and read out with the same probe, a transformer with two encoder and two decoder layers over the discrete tokens, trained on the full training split for 40k updates. Bits are nominal index bits per 256×256 image; for STROP, Tokens is the mean predicted program length and the rate includes the 6-bit length field. Values are means over 3 probe seeds, except rows marked † (one probe seed). FlexTok (Bachmann et al., 2025), One-D-Piece (Miwa et al., 2025) and ALIT (Duggal et al., 2025a) are off-the-shelf checkpoints without STROP training, so this is a system-level comparison: training data, teachers and objectives differ. Stuff-27 denotes COCO-Stuff-27. The best value in each column in bold.

Tokenizer	Model	Tokens	Bits	Stuff-27	VOC
FlexTok	D12-D12-IN1k	16	256	32.7	55.6
FlexTok	D12-D12-IN1k†	64	1024	36.3	61.4
FlexTok	D12-D12-IN1k	256	4096	35.4	61.3
One-D-Piece	S-256	21	252	18.6	29.2
One-D-Piece	S-256	256	3072	21.3	31.7
ALIT	small-IN1k	32	384	16.6	23.4
ALIT	small-IN1k	64	768	16.5	21.8
DINO-S	STROP-S (ours), forced prefix†	16	160	35.5	56.6
DINO-S	STROP-S (ours), adaptive	≈24.4	≈250	39.7	62.6
DINO-B	STROP-S (ours), adaptive	≈22.8	≈234	**42.3**	**66.9**

Datasets. All five variants are trained on both ImageNet-1k (Deng et al., 2009) and COCO 2017 (Lin et al., 2014) at 256×256 resolution. We additionally train on CLEVR (Johnson et al., 2016) at 128×128 with reduced program length ($K=32$) and smaller codebooks ($V \in \{32, 64, 128\}$) to study codebook sizing and program structure on a controlled scene domain (Appendix L).
Optimization. All variants share the same optimizer and schedule: AdamW with peak learning rate 10^{-3} (length head: 3×10^{-4}), batch size 32, and 5×10^{5} total steps. The learning rate follows a warmup-hold-cosine schedule (30k warmup, hold to 100k, cosine decay to 10^{-4}). The rate-matched comparison and the ablations (Tables 1 and 3) use the COCO-trained DINO-S/STROP-S configuration with three tokenizer seeds, each read out with three probe seeds. All other results come from a single checkpoint per configuration.

4.2 Evaluation protocol

We freeze STROP before every evaluation and consider three complementary protocols. First, single-token and paired-token erasures test whether program edits have localized effects and whether tokens interact beyond their individual contributions (Fig. 1; Appendix 11). Second, code-use statistics and unsupervised region readouts measure vocabulary reuse and the spatial organization of the generated programs (Fig. 4). Third, we train lightweight supervised readouts to compare the information retained in frozen DINOv3 features, the interpreted STROP field, and the active quantized program.

The downstream evaluation covers semantic segmentation on PASCAL VOC 2012, COCO-Stuff-27, ADE20K, and Cityscapes; multi-label classification on VOC and COCO-Stuff-27; and monocular depth estimation on NYUv2 (Fig. 12 in Appendix K). For semantic segmentation, all frozen representations are evaluated using the same data pipeline and optimization schedule. The readout architectures differ only where needed to accommodate spatial feature maps and token sequences. Appendix E provides the metric definitions, readout architectures, and complete training and evaluation settings.

4.3 Adaptive versus fixed-length programs

Table 1 compares STROP with fixed-length tokenizers trained separately with the same encoder, generator, codebook, and data. At about 250 bits per image, STROP beats the fixed $K=25$ model on all four benchmarks and for both readouts, by 1.6–3.1 percent points on mIoU. It also beats the

Table 3: Ablations of the STROP design. Segmentation mIoU (%) of the adaptive STROP of Table 1 (first row) and the change in mIoU (points) when one factor is changed and the model is retrained (DINO-S / STROP-S, COCO training). Red/blue: worse/better than STROP, saturating at 10 points. Evaluation as in Table 1 (3 tokenizer seeds, each averaged over 3 probe seeds); Mean is the average over the four benchmarks. Stuff-27 denotes COCO-Stuff-27. Measured rates on COCO val are ≈250 bits per image for STROP, 250 for the spatial grid, ≈248 for the noncausal generator, ≈244 for uniform truncation, ≈225 for V=512 and ≈276 for V=2048; rates of the other variants were not measured. †One of the three seeds collapses (3.0 mean latent mIoU, against 32.5 and 32.8 for the others). ‡2048 implies 11 instead of 10 bit cost (≈276 vs. 250 bits per image); at an equal bit budget, the larger codebook lowers teacher cosine by 0.010 on average, and no seed improves.

	Latent					Program				
Variant	VOC	Stuff-27	ADE20K	Citysc.	Mean	VOC	Stuff-27	ADE20K	Citysc.	Mean
STROP (adaptive)	68.6	41.2	28.0	36.8	43.7	61.5	39.3	25.3	37.8	41.0
Objective										
+ pixel gradients	−10.3	−5.3	−8.7	−7.4	−7.9	−11.7	−7.2	−8.1	−7.2	−8.6
cosine loss only	−2.9	−1.7	−2.7	−2.5	−2.4	−2.5	−1.8	−1.9	−2.3	−2.1
MSE loss only†	−30.5	−18.2	−16.1	−18.7	−20.9	−27.4	−17.2	−13.5	−17.6	−18.9
Ordering										
noncausal generator	−3.6	−1.4	−2.5	−3.3	−2.7	−2.8	−1.6	−1.9	−3.5	−2.5
spatial 5×5 grid	−0.2	−0.1	−0.1	0.0	−0.1	0.0	−0.3	+0.5	+0.9	+0.3
Curriculum										
joint from start	−4.6	−3.1	−4.2	−5.1	−4.2	−3.7	−3.6	−3.1	−4.5	−3.7
uniform truncation	−1.3	−1.1	−1.6	−1.7	−1.4	−0.6	−1.0	−0.8	−1.8	−1.1
abrupt handoff	+0.1	0.0	−0.2	−0.5	−0.1	−0.1	0.0	+0.2	−0.1	0.0
no oracle warm-up	+0.1	+0.2	0.0	0.0	+0.1	−0.3	+0.1	+0.7	−0.5	0.0
Vocabulary										
V=512	−1.7	−1.1	−1.5	−2.2	−1.6	−1.5	−1.1	−1.3	−1.8	−1.4
V=2048‡	+0.8	+0.5	+1.1	+1.9	+1.1	+0.4	+0.7	+1.2	+1.4	+0.9

fixed K=32 model everywhere with 22% fewer bits. Only the fixed K=64 model scores higher, at 2.6× the rate. Forcing the same adaptive checkpoint to longer prefixes does not help: its predicted length matches or exceeds every forced length up to 64 codes (Fig. 9). At a fixed average budget, the improvements therefore come from spending more codes on some images and fewer on others.

What drives the gain. Table 3 retrains the model with one factor changed at a time. Letting pixel-reconstruction gradients reach the generator costs the most, 7.9 mIoU for the latent readout and 8.6 for the program readout on average. Training all curriculum phases jointly from the start costs 4.2/3.7 points and a noncausal generator 2.7/2.5. Sampling Phase-1 truncation lengths uniformly, with no bias toward long prefixes, costs 1.4/1.1, while the handoff style and the oracle warm-up change little. The latent loss needs both of its terms: cosine alone loses 2.4/2.1 points, and with MSE alone one of three seeds collapses. A 5×5 spatial grid with the same 25 codes performs on par with STROP, so we do not claim that the sequential layout is better in itself. At a fixed length of 25 codes the sequential model trails that grid by 2.4/2.2 points (Table 8), and adaptive allocation closes the difference. The main contrasts also hold with ImageNet training, with a DINO-B teacher, and with the larger STROP-M generator, although the curriculum matters less for STROP-M (Appendix H).

4.4 Comparison with public adaptive tokenizers

Table 2 applies the transformer readout used for STROP programs to frozen tokens from FlexTok (Bachmann et al., 2025), One-D-Piece (Miwa et al., 2025), and ALIT (Duggal et al., 2025a). At about 250 bits per image, STROP reaches 39.7 mIoU on COCO-Stuff-27 and 62.6 on VOC, 7.0 points above FlexTok at 256 bits. With a DINO-B teacher it reaches 42.3 and 66.9 at about 234 bits. Longer sequences help the public tokenizers little. FlexTok peaks at 36.3 on COCO-Stuff-27 with 1024 bits and at 61.9 on VOC with 2048 bits, and One-D-Piece and ALIT stay below 22 mIoU on

COCO-Stuff-27 and 32 on VOC at every budget (Fig. 10). The tokenizers differ in training data, teacher, and objective, so the comparison measures how much segmentation-relevant content each system keeps per bit and cannot attribute the difference to a single component.

4.5 Alignment, transfer and structure

Alignment. Under the same 640-bit ceiling, alignment with the teacher falls as the teacher grows (cosine 0.86 for DINO-S, 0.81 for DINO-B and 0.75 for DINO-L with STROP-S), yet DINO-B and DINO-L segment better than DINO-S (Appendix I). **Downstream.** Across all ten variants, the COCO-trained DINO-B/STROP-S model segments best, with 47.3 mean latent mIoU against 61.3 for its frozen teacher. A small transformer readout on the program trails the latent readout by 2–7 mIoU on VOC, COCO-Stuff-27 and ADE20K, is ahead on Cityscapes, and reaches a slightly lower RMSE on NYUv2 depth, so most of the dense-prediction information survives in the discrete program (Appendix K). **Structure.** Erasing a single token changes a spatially localized part of the interpreted field (Fig. 1; token pairs in Appendix J), and clustering these erasure effects gives image regions similar to those obtained from DINO and latent features (Fig. 4). Program length correlates with object count on CLEVR (Pearson $r \approx 0.74$) and more weakly with scene complexity on COCO.

5 Limitations

(i) The most visible limitation is the readability of the raw program. With a small transformer readout, the program comes close to the interpreted field on depth and on most segmentation benchmarks. On multi-label classification, where the original attention-pooled linear readout is used, it trails far behind (Appendix K). Separating readout capacity from information lost at quantization will need readouts designed for programs that make fuller use of token order and prefix structure. (ii) Latent alignment is an incomplete predictor of transfer. The DINO-S encoder gives the best reconstruction scores in Table 11, but DINO-B and DINO-L transfer better (Table 10 in Appendix I). Useful structure in the teacher features can therefore survive quantization even when the raw reconstruction loss is higher. This calls for more discriminative latent-quality measures. (iii) We compare public adaptive tokenizers on COCO-Stuff-27 and VOC only, with a single frozen-token readout (Table 2). The systems differ in training data, teacher, and objective, so this comparison does not isolate adaptive length; Table 1 tests that against separately trained fixed-length models. (iv) The evidence is also unevenly replicated. The rate-matched comparison and the ablations use three tokenizer seeds, and their main contrasts hold for other training data, teachers, and generator sizes (Appendix H). The model-size grid uses one checkpoint per configuration, and the rate-matched fixed baseline is calibrated on COCO. (v) LLM scene decoding (Appendix M) recovers coarse layout but not reliable attribute binding. The learned codebook is therefore not yet a fully transparent symbolic interface for downstream language models. Closing this gap, either by changing the codebook structure or by aligning code identities with a captioning signal, is a direction we leave to future work.

6 Conclusion

We introduced STROP, a discrete visual tokenizer that predicts an image-dependent program length using local rate–distortion supervision against frozen DINOv3 features. At a matched rate, its adaptive programs keep more segmentation-relevant information than separately trained fixed-length programs, and with a shared readout they outperform public adaptive tokenizers. The gain depends mostly on training against features without pixel gradients and on the staged length curriculum, and it holds with ImageNet training and with a DINO-B teacher. Program length follows scene complexity, and erasing a single token changes a localized part of the interpreted field.

AI use statement

Claude (Anthropic), Codex, and ChatGPT (OpenAI) were used to improve clarity, grammar, and style of writing; to help identify and locate relevant related work; and to draft selected sections of the manuscript, which were subsequently reviewed, edited, and verified by the authors, who take full responsibility for the accuracy, originality, and integrity of the final content.

Reproducibility statement

To facilitate reproducibility, we released our implementation (see p. 3). All relevant hyperparameters and settings are provided in the main paper, the appendices, or in the referenced prior works.

References

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

Roman Bachmann, Jesse Allardice, David Mizrahi, Enrico Fini, Oğuzhan Fatih Kar, Elmira Amirloo, Alaaeldin El-Nouby, Amir Zamir, and Afshin Dehghan. FlexTok: Resampling images into 1D token sequences of flexible length. In *International Conference on Machine Learning (ICML)*, 2025.

Doojin Baek, Gyubin Lee, Junyeob Baek, Hosung Lee, and Sungjin Ahn. Learning to theorize the world from observation. *arXiv preprint arXiv:2605.03413*, 2026.

Adrien Bardes, Jean Ponce, and Yann LeCun. VICReg: Variance-invariance-covariance regularization for self-supervised learning. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=xm6YD62D1Ub.

Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your ViT but faster. In *International Conference on Learning Representations (ICLR)*, 2023.

Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Hao Chen, Yujin Han, Fangyi Chen, Xiang Li, Yidong Wang, Jindong Wang, Ze Wang, Zicheng Liu, Difan Zou, and Bhiksha Raj. Masked autoencoders are effective tokenizers for diffusion models. In *International Conference on Machine Learning (ICML)*, 2025a.

Hao Chen, Ze Wang, Xiang Li, Ximeng Sun, Fangyi Chen, Jiang Liu, Jindong Wang, Bhiksha Raj, Zicheng Liu, and Emad Barsoum. SoftVQ-VAE: Efficient 1-dimensional continuous tokenizer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025b.

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, 2009. doi: 10.1109/CVPR.2009.5206848.

Shivam Duggal, Phillip Isola, Antonio Torralba, and William T. Freeman. Adaptive length image tokenization via recurrent allocation. In *International Conference on Learning Representations (ICLR)*, 2025a.

Shivam Duggal et al. Single-pass adaptive image tokenization for minimum program search. *arXiv preprint arXiv:2507.07995*, 2025b.

Patrick Esser, Robin Rombach, and Björn Ommer. Taming transformers for high-resolution image synthesis. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

Christopher Fifty, Ronald G. Junkins, Dennis Duan, Aniketh Iger, Jerry W. Liu, Ehsan Amid, Sebastian Thrun, and Christopher Ré. Restructuring vector quantization with the rotation trick. In *International Conference on Learning Representations (ICLR)*, 2025.

Harari et al. Seeing more with less, 2025. PLACEHOLDER — verify against Reviewer d7RA's citation.

Andrew Jaegle, Felix Gimeno, Andrew Brock, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning (ICML)*, 2021.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier Hénaff, Matthew M. Botvinick, Andrew Zisserman, Oriol Vinyals, and João Carreira. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations (ICLR)*, 2022.

Jian Jia, Jingtong Gao, Ben Xue, Junhao Wang, Qingpeng Cai, Quan Chen, Xiangyu Zhao, Peng Jiang, and Kun Gai. From principles to applications: A comprehensive survey of discrete tokenizers in generation, comprehension, recommendation, and information retrieval, 2025. URL https://arxiv.org/abs/2502.12448.

Justin Johnson, Bharath Hariharan, Laurens van der Maaten, Li Fei-Fei, C. Lawrence Zitnick, and Ross B. Girshick. CLEVR: A diagnostic dataset for compositional language and elementary visual reasoning. *CoRR*, abs/1612.06890, 2016. URL http://arxiv.org/abs/1612.06890.

Andrei N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

Xingjian Leng, Jaskirat Singh, Yunzhong Hou, Zhenchang Xing, Saining Xie, and Liang Zheng. REPA-E: Unlocking VAE for end-to-end tuning with latent diffusion transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Ming Li and Paul Vitányi. *An Introduction to Kolmogorov Complexity and Its Applications*. Springer, 3 edition, 2008.

Tianhong Li, Dina Katabi, and Kaiming He. Return of unconditional generation: A self-supervised representation generation method. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024a.

Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Jiuxiang Gu, Bhiksha Raj, and Zhe Lin. ImageFolder: Autoregressive image generation with folded tokens. 2024b.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

Francesco Locatello, Dirk Weissenborn, Thomas Unterthiner, Aravindh Mahendran, Georg Heigold, Jakob Uszkoreit, Alexey Dosovitskiy, and Thomas Kipf. Object-centric learning with slot attention. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. Finite scalar quantization: VQ-VAE made simple. In *International Conference on Learning Representations (ICLR)*, 2024.

Keita Miwa, Kento Sasaki, Hidehisa Arai, Tsubasa Takahashi, and Yu Yamaguchi. One-D-Piece: Image tokenizer meets quality-controllable compression. *arXiv preprint arXiv:2501.10064*, 2025.

Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. BEiT v2: Masked image modeling with vector-quantized visual tokenizers. *arXiv preprint arXiv:2208.06366*, 2022.

Yongming Rao, Wenliang Zhao, Benlin Liu, Jiwen Lu, Jie Zhou, and Cho-Jui Hsieh. Dynamicvit: Efficient vision transformers with dynamic token sparsification. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Ali Razavi, Aäron van den Oord, and Oriol Vinyals. Generating diverse high-fidelity images with VQ-VAE-2. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Scott Reed and Nando de Freitas. Neural programmer-interpreters. In *International Conference on Learning Representations (ICLR)*, 2016.

Maximilian Seitzer, Max Horn, Andrii Zadaianchuk, Dominik Zietlow, Tianjun Xiao, Carl-Johann Simon-Gabriel, Tong He, Zheng Zhang, Bernhard Schölkopf, Thomas Brox, and Francesco Locatello. Bridging the gap to real-world object-centric learning. In *International Conference on Learning Representations (ICLR)*, 2023.

Junhong Shen, Kushal Tirumala, Michihiro Yasunaga, Ishan Misra, Luke Zettlemoyer, Lili Yu, and Chunting Zhou. CAT: Content-adaptive image tokenization. *arXiv preprint arXiv:2501.03120*, 2025.

Oriane Siméoni, Huy V. Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, Francisco Massa, Daniel Haziza, Luca Wehrstedt, Jianyuan Wang, Timothée Darcet, Théo Moutakanni, Leonel Sentana, Claire Roberts, Andrea Vedaldi, Jamie Tolan, John Brandt, Camille Couprie, Julien Mairal, Hervé Jégou, Patrick Labatut, and Piotr Bojanowski. DINOv3. *arXiv preprint arXiv:2508.10104*, 2025.

Gautam Singh, Fei Deng, and Sungjin Ahn. Illiterate DALL-E learns to compose. In *International Conference on Learning Representations (ICLR)*, 2022.

Keyu Tian, Yi Jiang, Zehuan Yuan, Bingyue Peng, and Liwei Wang. Visual autoregressive modeling: Scalable image generation via next-scale prediction. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Aäron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. Neural discrete representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Xin Wen, Bingchen Zhao, Ismail Elezi, Jiankang Deng, and Xiaojuan Qi. "Principal components" enable a new language of images. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025.

Ziyi Wu, Nikita Dvornik, Klaus Greff, Thomas Kipf, and Leonid Sigal. Slotformer: Unsupervised visual dynamics simulation with object-centric models. In *International Conference on Learning Representations (ICLR)*, 2023a.

Ziyi Wu, Jingyu Hu, Wuyue Lu, Igor Gilitschenski, and Animesh Garg. SlotDiffusion: Object-centric generative modeling with diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023b.

Wilson Yan, Matei Zaharia, Volodymyr Mnih, Pieter Abbeel, Aleksandra Faust, and Hao Liu. ElasticTok: Adaptive tokenization for image and video. In *International Conference on Learning Representations (ICLR)*, 2025.

Jiawei Yang, Tianhong Li, Lijie Fan, Yonglong Tian, and Yue Wang. Latent denoising makes good visual tokenizers. *arXiv preprint arXiv:2507.15856*, 2025.

Jingfeng Yao and Xinggang Wang. Reconstruction vs. generation: Taming optimization dilemma in latent diffusion models. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

Haotian Ye, Qiyuan He, Jiaqi Han, Puheng Li, Jiaojiao Fan, Zekun Hao, Fitsum Reda, Yogesh Balaji, Huayu Chen, Sheng Liu, Angela Yao, James Zou, Stefano Ermon, Haoxiang Wang, and Ming-Yu Liu. InfoTok: Adaptive discrete video tokenizer via information-theoretic compression. *arXiv preprint arXiv:2512.16975*, 2025.

Lijun Yu, Jose Lezama, Nitesh Bharadwaj Gundavarapu, Luca Versari, Kihyuk Sohn, David Minnen, Yong Cheng, Agrim Gupta, Xiuye Gu, Alexander G Hauptmann, Boqing Gong, Ming-Hsuan Yang, Irfan Essa, David A Ross, and Lu Jiang. Language model beats diffusion - tokenizer is key to visual generation. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=gzqrANCF4g.

Qihang Yu, Mark Weber, Xueqing Deng, Xiaohui Shen, Daniel Cremers, and Liang-Chieh Chen. An image is worth 32 tokens for reconstruction and generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024b.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *International Conference on Learning Representations (ICLR)*, 2025.

Yue Zhao, Hanwen Jiang, Zhenlin Xu, Chutong Yang, Ehsan Adeli, and Philipp Krähenbühl. Spherical leech quantization for visual tokenization and generation, 2025. URL https://arxiv.org/abs/2512.14697.

A ADAPTIVE PROGRAM-LENGTH CURRICULUM DETAILS

This appendix gives the full parameterization of the adaptive-length curriculum used in Section 2.2.

The generator emits a fixed sequence of K quantized tokens for every image, but the interpreter attends only to a prefix of length $L_i \in \{1, \dots, K\}$ for sample x_i. Tokens with positions $k > L_i$ are masked during training; at inference they are not passed to the interpreter. Curriculum design therefore reduces to the prediction of L_i during training, which we schedule across the following four phases.

Phase 1: Random truncation. The model is trained only on randomly truncated programs, which forces informative content to be front-loaded into earlier tokens. Lengths are sampled as

$$u_i \sim \mathrm{Beta}(\alpha_t, 1), \qquad L_i = \mathrm{clip}\big(\mathrm{round}\big(L_{\min}^{\mathrm{trunc}} + u_i(K - L_{\min}^{\mathrm{trunc}})\big),\, L_{\min}^{\mathrm{trunc}},\, K\big), \tag{1}$$

with the schedule $\alpha_s = \alpha_0 + (1 - \alpha_0)\min(s/T_{\mathrm{bias}}, 1)$, where s denotes the global optimization step. Since $\alpha_0 > 1$, early training is biased toward longer prefixes (easier reconstruction, faster initial convergence) and anneals linearly to a uniform distribution at $s = T_{\mathrm{bias}}$.

Phase 2: Oracle target estimation. We continue with random truncation but begin estimating for each sample how long the program *should* be. The base oracle uses the feature-space reconstruction error at the active length, $e_i(L_i) = \mathcal{L}_{\mathrm{lat}}(Z_{\mathrm{out},i}(L_i), Z_{e,i})$, normalized by a batch-level EMA $\bar{e}_t = \rho\,\bar{e}_{t-1} + (1-\rho)\,\frac{1}{B}\sum_i e_i(L_i)$:

$$L_i^{\mathrm{base}} = \mathrm{clip}\Big(\tfrac{e_i(L_i)}{\bar{e}_t + \epsilon}\,\beta K,\; L_{\min}^{\mathrm{oracle}},\, L_{\max}^{\mathrm{oracle}}\Big). \tag{2}$$

Samples reconstructing worse than the EMA running average receive longer targets; easier samples receive shorter ones. The hyperparameter β sets the average target length when $e_i \approx \bar{e}_t$. Targets are not yet used to drive truncation in this phase—they are only accumulated.

As L_i^{base} conflates sample difficulty with length adequacy (a hard sample has large error at any length), we complement it with a signal of the local slope of the rate–feature-distortion curve at L_i, which captures how much the *additional* tokens reduce error. For a probe radius δ, we calculate the reconstruction error at the clipped neighbors $L_i^{\pm} = \mathrm{clip}(L_i \pm \delta,\, L_{\min}^{\mathrm{trunc}},\, K)$ and form one-sided relative slopes

$$r_i^{\mathrm{short}} = \max\Big(0,\, \tfrac{e_i^- - e_i}{e_i + \epsilon}\Big), \qquad r_i^{\mathrm{long}} = \max\Big(0,\, \tfrac{e_i - e_i^+}{e_i + \epsilon}\Big), \tag{3}$$

which measure how much shortening hurts and how much lengthening still helps. We average over non-clamped probes, maintain EMAs $\bar{r}^{\mathrm{short}}, \bar{r}^{\mathrm{long}}$, and squash with a temperature-τ saturation $u_\bullet = \tanh(\bar{r}^{\bullet}/\tau) \in [0, 1]$. The pair $(u_{\mathrm{short}}, u_{\mathrm{long}})$ induces a soft three-way decision through the partition of unity

$$w_{\mathrm{compress}} = (1 - u_{\mathrm{short}})(1 - u_{\mathrm{long}}), \quad w_{\mathrm{keep}} = u_{\mathrm{short}}(1 - u_{\mathrm{long}}), \quad w_{\mathrm{extend}} = u_{\mathrm{long}}, \tag{4}$$

corresponding to three regimes: *compress* when the curve is flat near L_i, *keep* near the elbow (only shortening hurts), and *extend* whenever lengthening still pays. Extension dominates by design—unused marginal utility should be exploited regardless of the lower-side signal. The corrected target is

$$\tilde{L}_i = \mathrm{clip}\big(m\,L_i^{\mathrm{base}},\, L_{\min}^{\mathrm{oracle}},\, L_{\max}^{\mathrm{oracle}}\big), \qquad m = \sum_k m_k\, w_k, \tag{5}$$

with defaults $m_{\mathrm{compress}} < 1$, $m_{\mathrm{keep}} = 1$, and $m_{\mathrm{extend}} > 1$.

Phase 3: Supervised length-head training. We activate a length head $\hat{L}_i = K h_\phi(\mathrm{pool}(Z_{e,i}))$, a sigmoid MLP on pooled DINOv3 encoder features, supervised against the oracle target:

$$\mathcal{L}_{\mathrm{len}} = \tfrac{1}{B}\sum_i \Big(\tfrac{\hat{L}_i - \mathrm{sg}[\tilde{L}_i]}{K}\Big)^2. \tag{6}$$

The objective $\mathcal{L}_{\mathrm{len}}$ is optimized only with respect to the head parameters ϕ from this point onward. During Phase 3, the head is being trained but not yet deployed: truncation is still random (Eq. 1), so h_ϕ learns from a stable target distribution before controlling the active prefix.

Table 4: Curriculum phase schedule (in training steps $\times 10^3$) and shared hyperparameters. STROP-S variants begin the curriculum earlier to match their faster convergence; all other settings are identical across variants and datasets.

	STROP-S	STROP-M
Phase 1: random truncation only	0–100k	0–200k
Phase 2: + oracle targets	100k–150k	200k–250k
Phase 3: + length-head training	150k–200k	250k–300k
Phase 4: handoff ramp	200k–250k	300k–350k
Predicted lengths only	250k–500k	350k–500k
Shared curriculum hyperparameters		
Oracle: β=0.75, ρ=0.999, $L_{\min}^{\text{oracle}}$=2		
Probes: δ=2, τ=0.3, slope EMA=0.99		
Modulators: m_{compress}=0.4, m_{keep}=1.0, m_{extend}=1.3		
Truncation: $L_{\min}^{\text{trunc}}$=8, λ_{len}=1.0		

Table 5: Training objective, quantizer and optimization settings shared across all model variants.

Component	**Configuration**
Program	K=64 tokens, 2D patch + learned prefix positional encoding
Quantizer	V=1024, d_c=16, ℓ_2-normalized, EMA (γ=0.95), β=1.0
Decoder	FastGAN, L1 + 0.1 LPIPS-VGG, trained with stop-gradient on interpreter output
$\mathcal{L}_{\text{lat}}$	cosine + MSE vs. frozen DINOv3 patches, λ_{lat}=1.0
$\mathcal{L}_{\text{commit}}$	commitment loss, λ_{q}=1.0
$\mathcal{L}_{\text{div}}$	diversity regularizer, λ_{div}=0.3, 40k-step warmup
Optimizer	AdamW, batch size 32, 5×10^5 steps
Schedule	peak LR 10^{-3}; 30k-step warmup from $0.1\times$, hold to 100k, cosine decay to 10^{-4} at 500k
Length head	learning rate 3×10^{-4}

Phase 4: Handoff to predicted lengths. Let $s^{(4)}_{\text{ramp,start}}$ and $s^{(4)}_{\text{ramp,end}}$ be the first and last steps of the handoff ramp. We linearly ramp $a_s = \text{clip}((s - s^{(4)}_{\text{start}})/(s^{(4)}_{\text{end}} - s^{(4)}_{\text{start}}), 0, 1)$ and truncate the program by sampling based on the predicted length $\hat{L}_i$ as

$$L_i = \begin{cases} \text{clip}(\text{round}(\hat{L}_i), 1, K) & \text{w.p. } a_s, \\ L_i^{\text{rand}} & \text{w.p. } 1 - a_s, \end{cases} \tag{7}$$

Thus, Phase 4 gradually replaces random truncations with predicted ones, thereby avoiding the covariate shift of an abrupt switch. After $s \geq s^{(4)}_{\text{end}}$, the ramp has completed: all remaining training steps use predicted truncation lengths, while the length head continues to be updated by $\mathcal{L}_{\text{len}}$. The head continues to be trained with $\mathcal{L}_{\text{len}}$ throughout this phase.

At inference, no sampling or length loss is used: $\hat{L}_i$ is rounded and clipped to $[1, K]$, and the program is hard-truncated at that length before decoding.

Table 4 lists the phase schedule used for all main variants. The only difference between STROP-S and STROP-M is the onset of Phase 2: smaller generators converge faster during random truncation, so the curriculum begins at 100k steps rather than 200k. Each subsequent phase occupies a 50k-step window. All oracle, probe, and modulator hyperparameters are shared.

B CURRICULUM TIMING STABILITY

We ablated the *timing* of the phases rather than their removal: both schedules run the same four phases followed by training entirely on predicted lengths, under the same 500k-step budget. The schedule used in the main experiments completes the handoff by step 50k; the prolonged schedules complete it between steps 250k and 450k depending on generator size. Across ten otherwise matched



Figure 5: **Longest predicted programs.** Crops receiving the highest predicted lengths L_{pred}, typically cluttered or multi-object scenes.

configurations, the prolonged schedule reduced the final full-capacity composite validation loss in $8/10$ cases. This motivated the final schedule and shows that a large change to when the handoff happens does not systematically destabilize training.

C Qualitative program lengths

To give a direct sense of what the length head does on individual images, we show representative COCO crops selected along two axes: the predicted length itself, and the gap between the predicted length and what the reconstruction curve actually needs. For the latter we define L^*_{95} as the shortest prefix length at which reconstruction quality reaches 95% of the quality at full length $K{=}64$, computed per image from its own rate–distortion curve. Each panel caption reports the predicted length L_{pred}, L^*_{95}, reconstruction RMSE at L_{pred}, the gap to full length $\Delta_{64} = \text{RMSE}(L_{\text{pred}}) - \text{RMSE}(K{=}64)$, and the number of annotated classes present in the crop.

Figure 5 and Figure 6 show the programs STROP allocates the most and the fewest tokens to. Figures 7 and 8 isolate cases where L_{pred} and L^*_{95} disagree: *over-allocated* crops receive more tokens than the curve indicates are needed, and *under-allocated* crops receive fewer. These disagreement cases are the more informative ones, since they show where the length head's estimate and the direct rate–distortion measurement diverge, rather than where they agree.



Figure 6: **Shortest predicted programs.** Crops receiving the lowest predicted lengths L_{pred}, typically low-clutter, single-subject scenes.

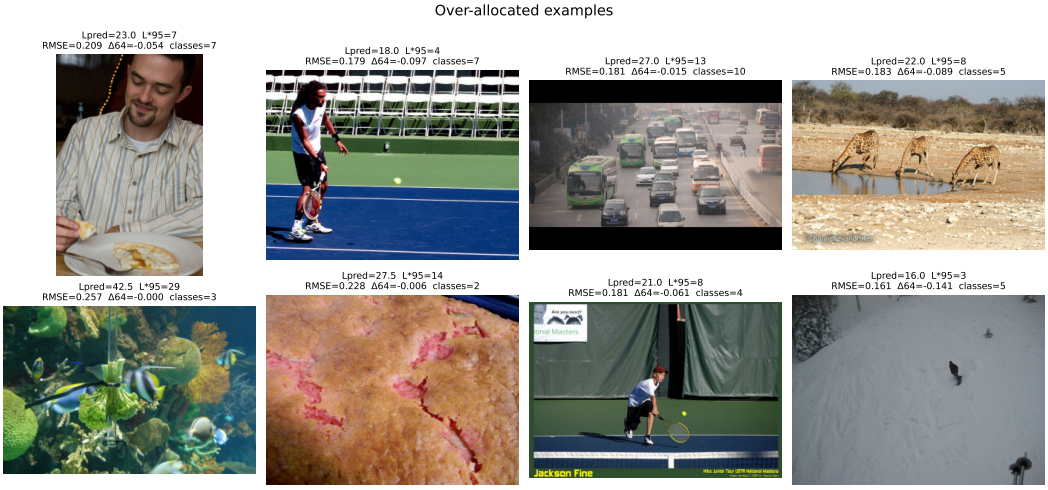


Figure 7: **Over-allocated.** Crops where L_{pred} exceeds L^*_{95} by the largest margin: the head spends more tokens than the rate–distortion curve indicates are needed.

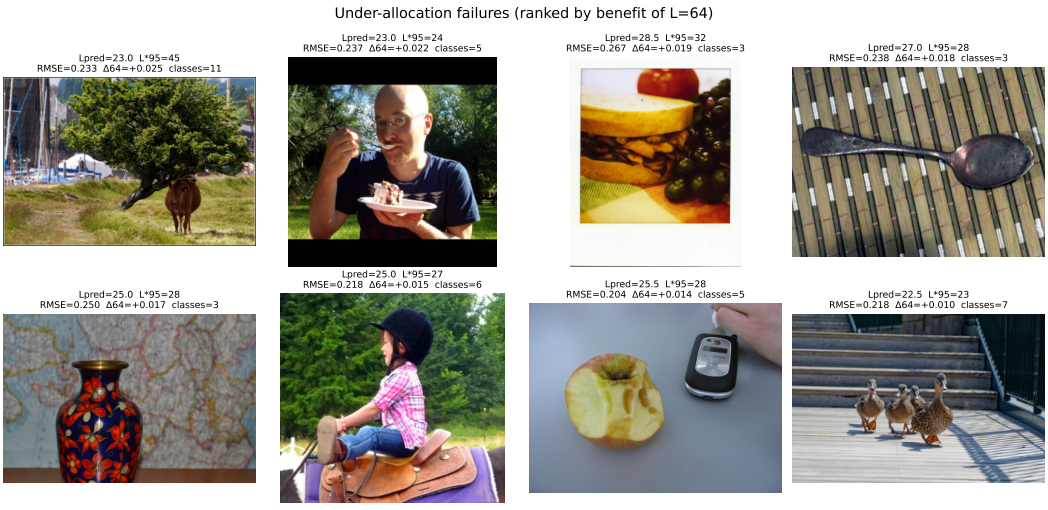


Figure 8: **Under-allocated.** Crops where L^*_{95} exceeds L_{pred} by the largest margin: the head spends fewer tokens than the curve indicates are needed.

D ARCHITECTURAL DETAILS

Table 6 lists the STROP variants used in the main experiments, including backbone, generator–interpreter size, parameter count, and single-H100 training time.

Table 6: Model variants trained on both COCO and ImageNet. The interpreter mirrors the generator's architecture. Training time is wall-clock on a single NVIDIA H100 (ImageNet).

Encoder	Model	d_{model}	Layers	FFN	Trainable (M)	Total (M)	Time (h)
DINO-S	STROP-S	256	8	1024	13.5	35.1	27
DINO-S	STROP-M	512	10	2048	64.3	85.9	45
DINO-B	STROP-S	256	8	1024	14.2	99.8	32
DINO-B	STROP-M	512	10	2048	65.1	150.8	50
DINO-L	STROP-M	512	10	2048	65.6	368.8	66

E EVALUATION PROTOCOL DETAILS

Frozen representations. All representation models are frozen before evaluation. We train a separate readout for each representation and dataset. Gradients from the readouts do not update DINOv3, the programmer, the codebook, or the interpreter. Unless stated otherwise, each result uses one probe initialization. The rate-matched comparison and the ablations (Tables 1 and 3) average three probe seeds for each of three tokenizer seeds; the external-tokenizer comparison uses three probe seeds where marked (Fig. 10).

Token-erasure interventions. Let Z_{out} denote the interpreted patch field obtained from the complete active program. To measure the effect of token position k, we mask that position and run the interpreter again, obtaining Z_{out}^{-k}. Masking preserves the original sequence length and token indices. The patchwise erasure effect is

$$\Delta_k(s) = 1 - \cos\left(Z_{\text{out}}(s), Z_{\text{out}}^{-k}(s)\right), \tag{8}$$

where s indexes a spatial position. We assign each patch to the active token whose removal produces the largest change,

$$a(s) = \arg\max_{k \in \mathcal{A}} \Delta_k(s), \tag{9}$$

where $\mathcal{A}$ is the set of active token positions. Masking inactive positions provides a control for the single-token intervention.

For a pair of active positions (i, j), let Z_{out}^{-ij} denote the interpreted field obtained after masking both tokens. We define

$$\Delta_{ij}(s) = 1 - \cos\left(Z_{\text{out}}(s), Z_{\text{out}}^{-ij}(s)\right) \tag{10}$$

and measure their positive supra-additive interaction as

$$\text{Syn}_{ij}^{+}(s) = \max\left(\Delta_{ij}(s) - \Delta_i(s) - \Delta_j(s), 0\right). \tag{11}$$

The scalar pair score is the mean over the 10% of patches with the largest positive interaction values. At the 16×16 feature resolution, this corresponds to 26 patches. These interventions are used in Figs. 1 and 11. The sampling and qualitative selection procedure for Fig. 11 is described alongside that figure.

Vocabulary reuse. Let V be the vocabulary size, n_c the number of occurrences of code c, and

$$N_{\text{tok}} = \sum_c n_c \tag{12}$$

the total number of active program tokens. We report the fraction of vocabulary entries used at least once,

$$\text{V}\% = 100\,\frac{|\{c : n_c > 0\}|}{V}. \tag{13}$$

To measure how evenly the vocabulary is used, we define $\pi_c = n_c/N_{\text{tok}}$ and compute the empirical code entropy

$$\mathcal{H}_{\text{code}} = -\sum_{c:n_c>0} \pi_c \log \pi_c. \tag{14}$$

The effective vocabulary size is $\exp(\mathcal{H}_{\text{code}})$. We report it relative to the complete vocabulary as

$$\text{Eff.}\% = 100\,\frac{\exp(\mathcal{H}_{\text{code}})}{V}. \tag{15}$$

A high V% indicates broad vocabulary coverage. The effective utilization remains low when most occurrences are concentrated on a small subset of codes.

Unsupervised region evaluation. The results in Fig. 4 use bundles of 512 images. Images 0–255 form the calibration subset, while images 256–511 are held out for evaluation. The calibration images come from the validation bundle; the official training split is unused.

For the DINOv3 and interpreted-field readouts, we cluster patch vectors from Z_e and Z_{out}, respectively. Both are ℓ_2-normalized before clustering. For the token readout, we rank active positions by their image-averaged erasure effect and retain at most 16 positions. At each spatial position s, their erasure values form the attribution vector

$$\boldsymbol{\Delta}(s) = (\Delta_{k_1}(s), \ldots, \Delta_{k_{16}}(s))\,. \tag{16}$$

Vectors from programs with fewer than 16 active positions are padded with zeros. Padded code identifiers are set to -1. The attribution vectors are not normalized before clustering.

We fit k-means using at most 40k labeled calibration patches, 40 iterations, one initialization, and seed 0. We use 27 clusters for COCO-Stuff-27 and 11 clusters for CLEVR.

For code-to-class transfer, each patch is assigned to the active token with the largest erasure effect. We then associate the code at that token position with its most frequent ground-truth class on the calibration subset. Void pixels are excluded when estimating this mapping. Codes absent from the calibration subset predict the void class. The resulting mapping remains fixed during evaluation on the held-out images.

Token grounding. For an image, we rank the active program tokens by their image-averaged erasure effect and keep the k most influential. Each token's map Δ_k is clipped at zero, divided by its 99th percentile and clipped to $[0, 1]$; patches with a normalized value of at least 0.45 form the token's predicted mask. Ground-truth object masks are projected to the 16×16 patch grid, and a patch belongs to an object when the object covers at least 20% of it. Predicted and ground-truth masks are matched one-to-one by the Hungarian algorithm on their IoU. We report the mean IoU over ground-truth objects (unmatched objects count as zero), the fraction of objects with IoU ≥ 0.5 (Recall@0.5), and the mean IoU over matched pairs. Scores are averaged over images; the comparison between models is relative, since no random-token baseline is reported.

Semantic-segmentation readouts. The semantic-segmentation results in Fig. 12 use the complete official training and validation splits. Spatial readouts operate on Z_e or Z_{out}. The program readout receives the active prefix of Z_q.

During training, we sample a scale $r \sim \mathcal{U}(0.5, 2)$ and resize the image so that its shorter side is $\lfloor 256r \rfloor$, preserving its aspect ratio. Images use bilinear interpolation with antialiasing, while segmentation masks use nearest-neighbor interpolation. We then extract a random 256×256 crop. When a multi-class crop is possible, sampling is repeated up to ten times if one non-void class occupies at least 75% of the labeled pixels. Horizontal flipping is applied with probability 0.5.

The photometric augmentation follows the DINOv3 evaluation pipeline. Independently with probability 0.5, we apply a brightness offset sampled from $[-32, 32]$, a contrast factor from $[0.5, 1.5]$, a saturation factor from $[0.5, 1.5]$, and a hue shift from $[-0.5, 0.5]$. Contrast is placed randomly before or after the saturation and hue transformations. Images are normalized using the ImageNet mean $(0.485, 0.456, 0.406)$ and standard deviation $(0.229, 0.224, 0.225)$. If padding is required, normalized images are padded with zeros and target masks with the ignore label 255. The frozen representations are computed online from the augmented images.

The readout for Z_e and Z_{out} consists of `Dropout2d` with probability 0.1, synchronized batch normalization, and a 1×1 convolution. For Z_q, the active quantized vectors are projected to width

Table 7: Reference evaluation of the probe implementation on frozen DINOv3 teachers at 512×512 (sliding window, stride 341), following the DINOv3 linear segmentation protocol (Siméoni et al., 2025). On ADE20K, our implementation matches the result of the official evaluation code for DINO-L (54.9 mIoU, the published value). These numbers validate the evaluation pipeline only: they concern teachers, not STROP, and are not comparable with the matched 256×256 results. Segmentation mIoU (%, ↑).

Dataset	Evaluation	DINO-S	DINO-B	DINO-L
ADE20K	DINOv3 evaluation code	—	—	54.9
ADE20K	our implementation of the same protocol	—	—	55.1
VOC	our implementation, DINOv3-style linear	84.3	88.2	88.2
Citysc.	our implementation, DINOv3-style linear	67.3	71.6	73.5

256 and combined with learned positional embeddings. The program readout contains two pre-normalization Transformer encoder layers, 256 learned output queries arranged as a 16×16 field, and two Transformer decoder layers. Each layer uses four attention heads, an FFN width of 1024, and dropout 0.1. A final linear projection produces the class logits.

We train each readout for 40k updates, including 1.5k warmup updates, using AdamW with a learning rate of 10^{-3}. The global batch size is 16, formed by two crops on each of eight processes. We use probe seed 0 and data seed 100 unless stated otherwise. Weight decay is 10^{-3} for the spatial readouts and 10^{-4} for the program readout. Training minimizes cross-entropy over non-void pixels. We evaluate the complete validation set every 5k updates to monitor training and report the final checkpoint after 40k updates.

At validation time, we resize the shorter image side to 256 and apply no stochastic augmentation. We evaluate 256×256 windows with stride 171. The logits from each window are resized bilinearly and averaged in overlapping regions. The merged logits are then resized to the original label resolution. All non-void pixels are evaluated, without flip-based test-time augmentation.

As a reference evaluation of the probe implementation, we also evaluate the frozen teachers at 512×512 with stride 341, following the DINOv3 linear protocol (Table 7). These numbers concern teachers only and are not comparable with the matched 256×256 results.

Multi-label classification. The multi-label experiment retains its original task-specific protocol. We train on the first 1000 images of the official training split and evaluate on the complete validation split. No online augmentation is applied. Representations from Z_e and Z_{out} are mean-pooled over their spatial dimensions. The active prefix of Z_q is aggregated using attention pooling.

We train for 20 epochs using AdamW with a learning rate of 10^{-3}, weight decay 10^{-4}, batch size 256, and seed 0. The objective is binary cross-entropy. We report mean average precision from the final epoch.

Monocular depth estimation. For NYUv2, the dense readout predicts logits over 256 linearly spaced depth bins between $0.001\,\mathrm{m}$ and $10\,\mathrm{m}$. Given logit ℓ_b and bin centre d_b, we compute

$$w_b = \frac{\max(\ell_b, 0) + 0.1}{\sum_{b'} \left(\max(\ell_{b'}, 0) + 0.1\right)}. \tag{17}$$

The continuous depth prediction is

$$\hat{d} = \mathrm{clip}\left(\sum_b w_b d_b,\ 0.001,\ 10\right). \tag{18}$$

We train the depth readouts for 38.4k updates, including 12.8k warmup updates, using AdamW with a learning rate of 3×10^{-4}, weight decay 10^{-4}, a global batch size of 16, and seed 321. We report the final checkpoint. As a reference, the frozen teachers are additionally evaluated with the official DINOv3 NYUv2 depth protocol (Table 17, bottom block); all STROP comparisons use the matched protocol above.

Matched external-tokenizer comparison. For the matched comparison with external tokenizers, all program representations use the same readout architecture and semantic-segmentation pipeline.

Each readout is trained for 40k updates, including 1.5k warmup updates, with a global batch size of 16, a learning rate of 10^{-3}, and weight decay 10^{-4}. Budgets shown with filled markers in Fig. 10 average three probe seeds; the remaining budgets use one. Depending on the input token dimension, the readouts contain 3.76–3.84M parameters and require 1.14–1.18 GFLOPs.

Rate accounting. We report the mean active program length,

$$\bar{L} = \frac{1}{N_{\text{img}}} \sum_i L_i. \tag{19}$$

For $H \times H$ inputs, we also estimate the idealized entropy rate from the empirical code distribution,

$$\text{bpp} = \frac{\sum_i \sum_{t=1}^{L_i} -\log_2 \pi_{c_{i,t}}}{N_{\text{img}} H^2}, \tag{20}$$

where $c_{i,t}$ is the discrete code ID at position t in the program for image i. This estimate includes the emitted token sequence and excludes the model, codebook, and sequence-length metadata.

F RATE-QUALITY CURVES

F.1 INTERNAL RATE-QUALITY: ADAPTIVE VS. FIXED LENGTH

Nominal bits. Codes are drawn from an $V{=}1024$ codebook, so $b_{\text{code}} = \log_2 1024 = 10$ bits/token. The adaptive model also carries a length field, $\lceil \log_2 64 \rceil = 6$ bits, so a crop of length L costs $R(L) = 10L + 6$ bits; the plotted mean rate uses $\mathbb{E}[L]$ (24.38 on both COCO-Stuff-27 and VOC, $\bar{R} \approx 250$ bits). Fixed-K baselines carry no length field: $R_K = 10K$ (160/250/320/640 bits for $K{=}16/25/32/64$). These are bits per crop, not per source image – larger images yield multiple crops.

Takeaway. Figure 9 shows the effect directly. At matched rate, the learned length head matches or beats both a uniform-allocation control and the fixed-K frontier, consistent with Table 1. The adaptive curve saturates past its elbow, while the fixed-K frontier keeps climbing: forcing extra tokens through a checkpoint trained for adaptive allocation stops helping, but budgeting for more tokens from the start keeps buying quality.

F.2 EXTERNAL SEMANTIC RATE-TRANSFER COMPARISON

Figure 10 plots the resulting frontier for all tokenizers. This is a semantic rate-transfer comparison, not a reconstruction rate-distortion curve, and the rates are code-index accounting, not actual compressed file size. The single matched-rate point closest to ≈ 250 bits is reported in Table 2; these curves show the full frontier each tokenizer traces, and the caveat given there – that this is a system-level comparison across different architectures, training data, and objectives, not a controlled ablation – applies to the whole curve, not just the matched point.

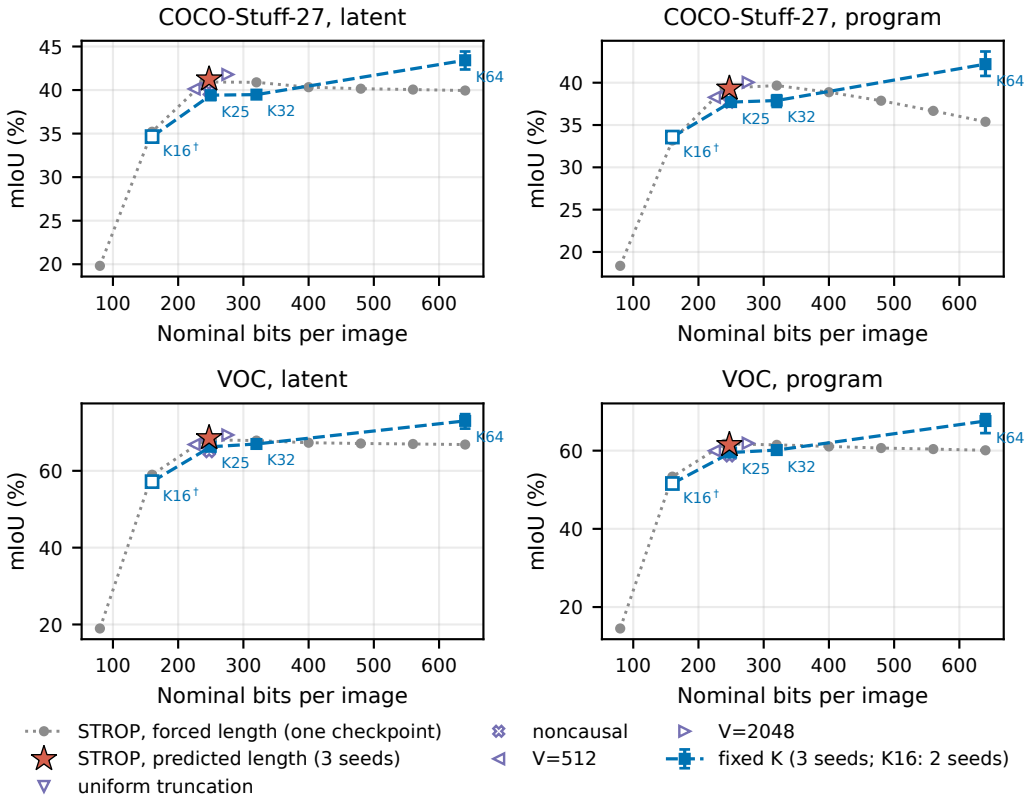


Figure 9: **Internal rate–quality.** Segmentation mIoU against nominal bits per 256×256 image for separately trained fixed-length models ($K=16$, 25, 32, 64; mean and range over 3 tokenizer seeds; †for $K=16$ over the 2 seeds that did not collapse), the adaptive STROP at its predicted length (star; mean and range over 3 tokenizer seeds), the same adaptive checkpoint truncated to forced lengths 8–64 (one checkpoint, final probes), and single ablations at their measured rate (open markers). Probes as in Table 1.

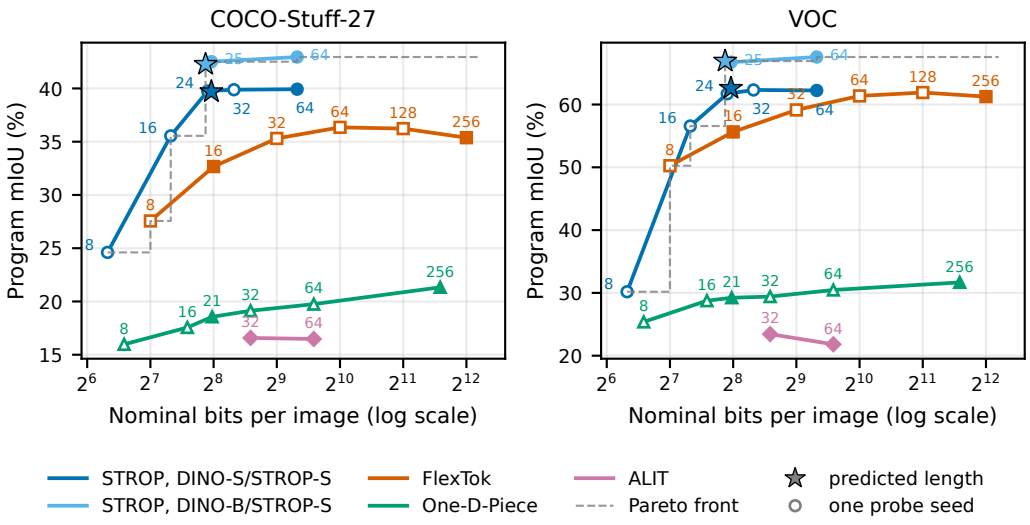


Figure 10: **Semantic rate–transfer frontier.** Program mIoU against the nominal number of code-index bits per 256×256 image for STROP and public adaptive tokenizers, under the same frozen-token probe (Table 2). Each line connects one tokenizer's token budgets (labelled by token count); stars mark STROP's predicted length. Filled markers average three probe seeds, hollow markers use one. The dashed line is the Pareto front over all points.

G ABLATION DETAILS

Table 8 reports the absolute scores behind the changes in Table 3, with the spread over tokenizer seeds and higher-rate spatial references.

Table 8: Absolute segmentation mIoU (%, ↑) of every separately trained system in the ablation study (DINO-S / STROP-S, COCO training), complementing the changes shown in Table 3. Values are means over 3 tokenizer seeds, each averaged over 3 probe seeds, with the protocol of Table 1; Mean averages the four benchmarks and its subscript is the standard deviation over tokenizer seeds. Bits are nominal index bits per image on COCO val; — marks rates that were not measured (the continuous grid has no discrete rate). The high-rate spatial references encode the DINO patch grid with a fixed 8×8 or 16×16 token grid and are shown for context only. †One of the three seeds collapses: mean latent mIoU 3.0 / 32.5 / 32.8 for MSE only and 34.5 / 2.5 / 34.5 for $K=16$ across the three seeds. Stuff-27 denotes COCO-Stuff-27.

		Latent					Program				
Variant	Bits	VOC	Stuff-27	ADE20K	Citysc.	Mean	VOC	Stuff-27	ADE20K	Citysc.	Mean
Teacher (frozen DINO)											
DINO-S teacher	—	80.3	46.2	42.1	56.7	56.3	—	—	—	—	—
STROP (ours)											
STROP (adaptive)	≈250	68.6	41.2	28.0	36.8	43.7$_{\pm0.4}$	61.5	39.3	25.3	37.8	41.0$_{\pm0.4}$
Fixed length											
$K=16$†	160	39.3	23.3	13.3	19.5	23.8$_{\pm18.5}$	35.6	23.1	13.1	21.6	23.4$_{\pm17.5}$
$K=25$	250	66.3	39.4	25.0	33.9	41.1$_{\pm1.5}$	59.5	37.7	23.4	35.7	39.1$_{\pm1.4}$
$K=32$	320	67.0	39.5	25.8	34.9	41.8$_{\pm0.4}$	60.1	37.9	23.7	35.9	39.4$_{\pm0.4}$
$K=64$	640	73.0	43.4	32.3	42.0	47.7$_{\pm1.8}$	67.6	42.2	29.5	42.9	45.6$_{\pm2.0}$
Objective											
+ pixel gradients	—	58.3	36.0	19.3	29.4	35.7$_{\pm0.4}$	49.8	32.1	17.2	30.6	32.4$_{\pm0.3}$
cosine loss only	—	65.6	39.6	25.4	34.2	41.2$_{\pm2.3}$	59.0	37.6	23.4	35.5	38.9$_{\pm2.1}$
MSE loss only†	—	38.0	23.1	12.0	18.1	22.8$_{\pm17.1}$	34.1	22.1	11.8	20.2	22.1$_{\pm16.5}$
Ordering											
noncausal generator	≈248	65.0	39.8	25.5	33.4	40.9$_{\pm2.9}$	58.6	37.7	23.4	34.3	38.5$_{\pm2.8}$
spatial 5×5 grid	250	68.3	41.2	27.9	36.8	43.6$_{\pm0.5}$	61.5	39.0	25.8	38.7	41.3$_{\pm0.3}$
Curriculum											
joint from start	—	63.9	38.2	23.9	31.7	39.4$_{\pm1.6}$	57.7	35.7	22.2	33.3	37.2$_{\pm1.1}$
uniform truncation	≈244	67.3	40.2	26.5	35.1	42.2$_{\pm0.8}$	60.9	38.3	24.5	36.0	39.9$_{\pm0.5}$
abrupt handoff	—	68.7	41.2	27.9	36.3	43.5$_{\pm0.1}$	61.4	39.3	25.6	37.7	41.0$_{\pm0.3}$
no oracle warm-up	—	68.6	41.4	28.1	36.8	43.7$_{\pm0.2}$	61.2	39.4	26.0	37.3	41.0$_{\pm0.1}$
Vocabulary											
$V=512$	≈225	66.8	40.1	26.5	34.6	42.0$_{\pm0.3}$	60.0	38.3	24.0	36.0	39.6$_{\pm0.3}$
$V=2048$	≈276	69.3	41.8	29.1	38.6	44.7$_{\pm0.2}$	61.9	40.0	26.5	39.3	41.9$_{\pm0.5}$
High-rate spatial references											
grid 8×8, VQ	640	75.0	44.3	34.2	45.9	49.8$_{\pm0.6}$	68.1	43.0	31.5	46.2	47.2$_{\pm1.1}$
grid 8×8, VQ, conv.	640	71.0	42.2	30.8	42.8	46.7$_{\pm0.1}$	68.1	43.3	30.6	46.5	47.1$_{\pm0.0}$
grid 16×16, VQ	2560	78.0	45.5	38.4	51.6	53.4$_{\pm0.1}$	72.2	44.9	35.0	52.9	51.2$_{\pm0.8}$
grid 8×8, continuous	—	78.3	45.8	39.2	52.6	54.0$_{\pm0.1}$	75.6	47.6	37.6	54.5	53.8$_{\pm0.2}$

H REPLICATION ACROSS CONFIGURATIONS

Table 9 repeats the main contrasts with a different training set (ImageNet), a larger teacher (DINO-B) and a larger generator (STROP-M). Each configuration is trained with three tokenizer seeds.

Table 9: Replication of the main contrasts across training data, teacher size and generator size. Each column pair is one configuration (training data, encoder / model); cells give the change in COCO-Stuff-27 segmentation mIoU (points) for the latent (L) and program (P) probes, shaded as in Table 3 (red: worse, blue: better). Means over 3 tokenizer seeds; a superscript gives the number of seeds that agree in sign with the mean when fewer than all three do; — marks contrasts that were not run. The first configuration uses the protocol of Table 1; the others use one probe seed. Bits give the nominal rate of the adaptive and the rate-matched fixed tokenizer on COCO val (the ImageNet-trained length was matched on ImageNet val, so on COCO its adaptive rate is higher). Order: teacher cosine of the interpreted latents before → after shuffling the program at test time (mean over tokenizer seeds; 5,000 COCO val images). "Adaptive − fixed" compares with the separately trained fixed-length tokenizer matched to the adaptive rate; "sequential − grid" compares a sequential and a 5×5 spatial tokenizer, both with 25 tokens.

	COCO DINO-S/STROP-S		ImageNet DINO-S/STROP-S		COCO DINO-B/STROP-S		COCO DINO-S/STROP-M	
Contrast	L	P	L	P	L	P	L	P
adaptive − fixed	+1.8	+1.6	+2.4	+2.1	+4.8	+3.8	—	—
+ pixel gradients	−5.3	−7.2	−4.6	−6.0	—	—	—	—
sequential − grid, K=25	−1.8	−1.3	—	—	−3.2	−2.5	—	—
noncausal generator	$-1.4^{2/3}$	$-1.6^{2/3}$	—	—	—	—	−0.5	−2.3
joint from start	−3.1	−3.6	—	—	—	—	$-0.8^{2/3}$	$-0.9^{2/3}$
Bits (adapt. / fixed)	≈250 / 250		269–280 / 250		234–236 / 230		259–266 / —	
Order (Cos)	0.86 → 0.53		0.85 → 0.49		0.81 → 0.53		0.85 → 0.47	

I ALIGNMENT

Table 12 expands Table 11 with per-dataset latent alignment and codebook statistics.

Table 11 compares the interpreted feature field of each variant with frozen DINOv3 features. Under the same 64-token, 1024-entry bottleneck, DINO-S is easiest to compress: DINO-S/STROP-S gives the best COCO alignment (.855 cosine, $R^2 = .734$), while DINO-S/STROP-M is best on ImageNet (.851, $R^2 = .727$). DINO-B remains around .80 cosine and $R^2 \approx .64$, and DINO-L drops to $\sim .72/.53$, consistent with a fixed-rate bottleneck facing higher-dimensional teacher features. Larger generators mainly increase vocabulary coverage: STROP-M uses 91–98% of the codebook, but this does not monotonically improve alignment, e.g., COCO DINO-S/STROP-S outperforms DINO-S/STROP-M despite lower codebook usage.

Alignment and transfer. Table 10 varies the teacher with the STROP-S generator fixed. Teacher cosine is 0.858 for DINO-S, 0.810 for DINO-B, and 0.747 for DINO-L, yet DINO-B segments better than DINO-S (44.8 against 41.5 latent mIoU on COCO-Stuff-27) and DINO-L matches DINO-B. The DINO-L teacher itself is only slightly better than DINO-B on these benchmarks. Across the full model grid, the larger STROP-M generator does not consistently help under the shared training schedule (Appendix K).

Compression. With 256×256 inputs and a patch size of 16, every encoder emits a fixed grid of 256 patch tokens (with dimension D) per image, giving per-image feature tensors of shape 256×384 (DINO-S), 256×768 (DINO-B), and 256×1024 (DINO-L). STROP compresses each of these backbone-dependent tensors into an adaptive sequence of at most 64 discrete tokens drawn from a codebook of size V=1024 (d_c=16), corresponding to a hard ceiling of $64 \cdot \log_2 V = 640$ bits per image regardless of encoder choice. Against the continuous float32 source representations of $256 \cdot D \cdot 32$ bits, this yields worst-case (full 64-token) compression ratios of ∼4,915× (DINO-S), ∼9,830× (DINO-B), and ∼13,107× (DINO-L), with the use of length head typically leading to shorter sequences and higher ratios. The total source entropy scales linearly with D, but the bit budget is held constant, so larger backbones face a strictly harder rate–distortion problem: DINO-L must compress 2.67× more raw feature dimensions than DINO-S into the same 640-bit envelope.

Table 10: Teacher size at fixed generator capacity. The five main variants (Table 6) pair DINO-L only with STROP-M, which confounds teacher and generator size; here the teacher varies with STROP-S fixed (COCO training). Bits is the nominal rate and Cos the cosine similarity between the interpreted latents and the frozen teacher's patch features, both on 5,000 COCO val images. Segmentation mIoU (%, ↑) is reported for the frozen teacher (an upper bound) and for the STROP latent and program readouts, with probes as in Table 1; one checkpoint and one probe seed per row. A second DINO-L tokenizer seed gives 44.8 (latent) and 42.6 (program) on COCO-Stuff-27 at 232.8 bits. Stuff-27 denotes COCO-Stuff-27. The best non-teacher value in each column is **bolded**.

			Stuff-27 mIoU ↑			VOC mIoU ↑		
Encoder	Bits	Cos↑	Teacher	Latent	Program	Teacher	Latent	Program
DINO-S	≈250	**0.858**	46.1	41.5	39.7	80.2	68.8	62.5
DINO-B	≈234	0.810	51.6	**44.8**	42.3	84.3	**72.9**	66.6
DINO-L	≈234	0.747	52.7	**44.8**	**42.8**	84.7	**72.9**	**67.7**

Table 11: Latent reconstruction quality and codebook utilization, averaged over five evaluation datasets (VOC, COCO-Stuff-27, ADE20K, Cityscapes, NYUv2). Cos and R^2 are measured against frozen DINO teacher patches; RMSE is computed per patch-feature scalar. V% is the fraction of the 1024-entry codebook used at least once by active program tokens; Eff.% is the entropy-effective utilization (Appendix E).

	Trained on COCO					Trained on ImageNet				
Variant	Cos↑	R^2 ↑	RMSE↓	V%↑	Eff.%↑	Cos↑	R^2 ↑	RMSE↓	V%↑	Eff.%↑
DINO-S/STROP-S	0.855	0.734	0.196	56.1	44.7	0.849	0.724	0.199	56.2	44.5
DINO-S/STROP-M	0.835	0.697	0.208	98.0	78.6	0.851	0.727	0.198	98.1	72.0
DINO-B/STROP-S	0.801	0.643	0.246	53.2	41.2	0.763	0.583	0.266	77.2	60.0
DINO-B/STROP-M	0.801	0.642	0.246	94.8	78.7	0.801	0.643	0.246	91.2	75.3
DINO-L/STROP-M	0.725	0.526	0.198	95.9	81.7	0.723	0.527	0.198	92.2	72.9

Table 12: Latent alignment per evaluation dataset. Cosine similarity and R^2 between the interpreter's predicted patch latents and frozen DINO teacher patch features on the validation split of each evaluation dataset. Codebook statistics are reported in Table 11. Stuff-27 denotes COCO-Stuff-27.

	Cos↑					R^2 ↑				
Variant	VOC	Stuff-27	ADE20K	Citysc.	NYUv2	VOC	Stuff-27	ADE20K	Citysc.	NYUv2
Trained on COCO										
DINO-S/STROP-S	0.859	0.858	0.854	0.849	0.853	0.742	0.742	0.731	0.722	0.731
DINO-S/STROP-M	0.841	0.839	0.835	0.834	0.825	0.709	0.707	0.696	0.693	0.679
DINO-B/STROP-S	0.804	0.810	0.792	0.799	0.798	0.650	0.661	0.629	0.639	0.637
DINO-B/STROP-M	0.808	0.810	0.794	0.797	0.794	0.656	0.661	0.631	0.633	0.630
DINO-L/STROP-M	0.745	0.748	0.714	0.713	0.703	0.557	0.565	0.510	0.505	0.491
Trained on ImageNet										
DINO-S/STROP-S	0.863	0.854	0.854	0.828	0.848	0.750	0.734	0.731	0.685	0.721
DINO-S/STROP-M	0.865	0.854	0.857	0.834	0.844	0.753	0.734	0.736	0.694	0.716
DINO-B/STROP-S	0.782	0.768	0.761	0.744	0.759	0.614	0.594	0.581	0.551	0.576
DINO-B/STROP-M	0.819	0.805	0.803	0.780	0.796	0.674	0.654	0.647	0.605	0.636
DINO-L/STROP-M	0.759	0.739	0.723	0.688	0.708	0.582	0.554	0.528	0.467	0.504

J PAIRWISE TOKEN SYNERGIES ALIGN WITH SEMANTIC REGIONS

Pairwise interactions provide another view of the structure learned by the discrete image programs. As shown in Fig. 11, erasing two tokens jointly can produce localized changes that align with coherent semantic regions.

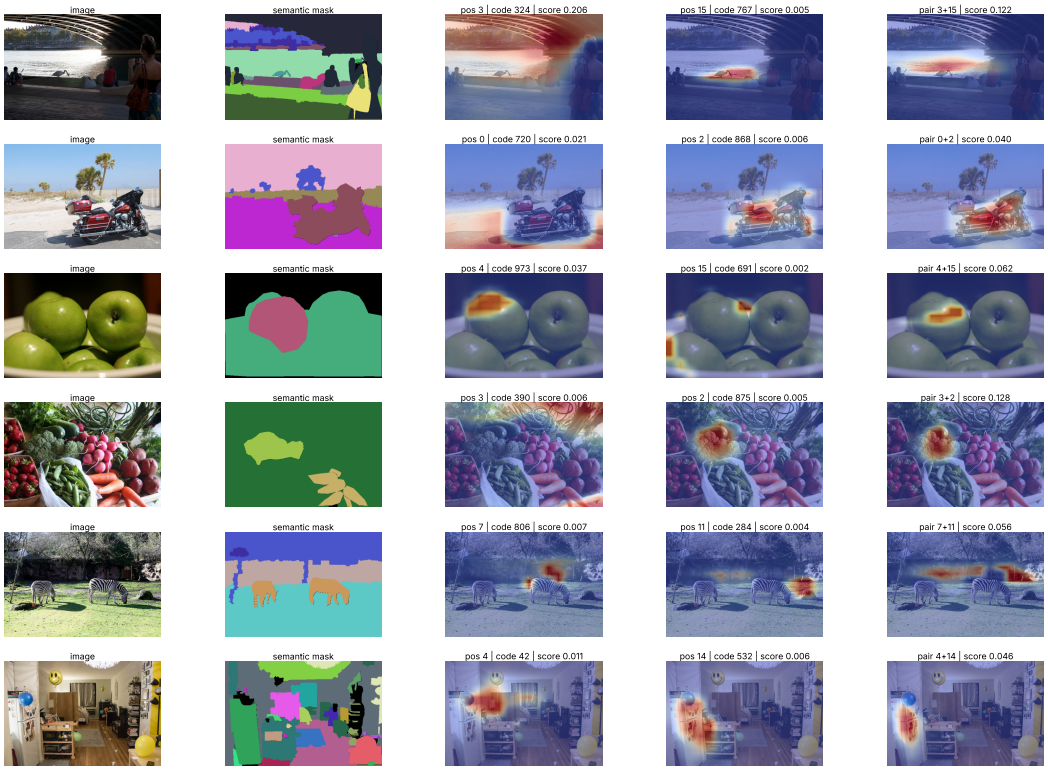


Figure 11: **Pairwise program-token synergies.** We erase pairs of discrete program tokens and reinterpret the remaining sequence, then visualize the resulting change in the DINO-aligned patch field. The selected COCO-Stuff-27 examples show that high-synergy token pairs often produce localized responses aligned with semantic mask regions. Heatmaps are normalized per panel for visualization; pair scores are raw attribution magnitudes.

K DOWNSTREAM PROBE DETAILS

We report downstream probes of the teacher, interpreted-field, and program representations for all model variants: Tables 13–15 give multi-label classification under its original protocol, Table 16 gives semantic segmentation, and Table 17 gives NYUv2 depth (protocols in Appendix E). The COCO-trained DINO-B/STROP-S model segments best, with 47.3 mean mIoU for the latent readout against 61.3 for its frozen teacher. For NYUv2 depth, the program readouts of DINO-L/STROP-M and DINO-B/STROP-S give the lowest errors among STROP variants, still above those of the frozen teachers (Table 17).

Program and interpreted field. The program readout is a small transformer over the active codes. On segmentation it trails the latent readout by 7 points on VOC and by 2–3 points on COCO-Stuff-27 and ADE20K, and it is ahead on Cityscapes (Table 1). On depth it reaches a slightly lower RMSE than the latent readout for all COCO-trained variants. The program falls far behind only in the multi-label experiment, which keeps its original attention-pooled linear readout. Most of the dense-prediction information is therefore still present in the discrete program.

Program length tracks scene complexity. STROP allocates longer programs to richer scenes. On CLEVR the correlation between active program length and ground-truth object count is strong

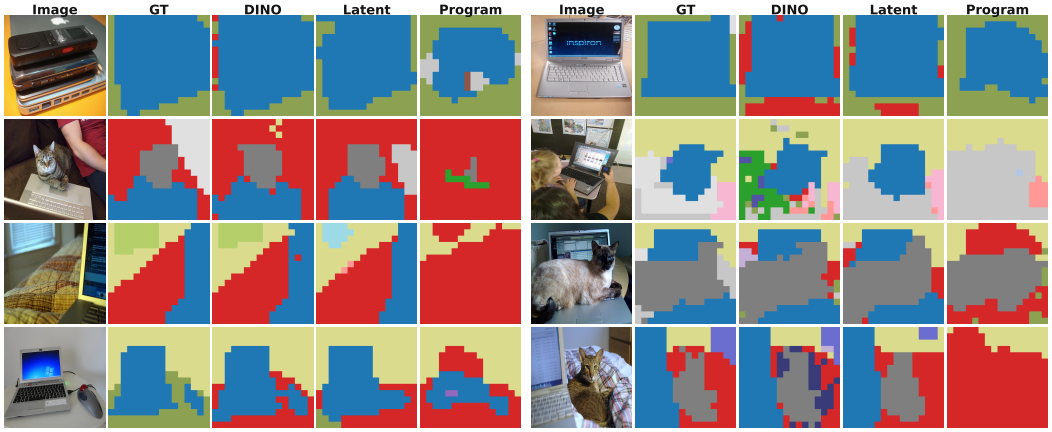


Figure 12: Supervised downstream segmentation probes on COCO-Stuff-27. DINO and Latent show predictions from linear patch probes trained on frozen DINO teacher features and STROP latent patch features, respectively. Program shows predictions from a probe trained on the quantized program.

Table 13: Linear probe mAP (%, ↑) on multi-label classification. STROP trained on COCO; Stuff-27 = COCO-Stuff-27; Mean = mean of Stuff-27 and VOC mAP. Teacher uses mean-pooled DINO patch features; latent is mean-pooled over the interpreter's spatial latent field; program is attention-pooled over the active program token vectors z_q. Best non-teacher value per column is **bolded**. Probes follow the original multi-label protocol (1,000 training images; Appendix E).

			Multi-label mAP ↑		
Encoder	Model	Repr.	Stuff-27	VOC	Mean
Teacher (frozen DINO)					
DINO-S	—	teacher	56.2	59.4	57.8
DINO-B	—	teacher	61.2	74.7	67.9
DINO-L	—	teacher	59.7	68.4	64.0
STROP (ours)					
DINO-S	STROP-S	latent	56.6	61.5	59.1
DINO-S	STROP-S	program	34.9	17.8	26.3
DINO-S	STROP-M	latent	56.5	59.4	58.0
DINO-S	STROP-M	program	30.5	13.3	21.9
DINO-B	STROP-S	latent	**61.7**	74.1	67.9
DINO-B	STROP-S	program	36.1	18.5	27.3
DINO-B	STROP-M	latent	61.6	**75.2**	**68.4**
DINO-B	STROP-M	program	31.3	14.7	23.0
DINO-L	STROP-M	latent	59.9	68.3	64.1
DINO-L	STROP-M	program	34.5	16.1	25.3

(Pearson $r \approx 0.74$; Figure 13a). On COCO, where scene complexity is a noisier quantity, the correlation against the number of unique semantic classes is positive but weaker (Figure 13b). The same trend holds against the spatial variance of patch-level DINO features (Figure 14). CLEVR ablations are in Appendix L.

Table 14: Linear probe mAP (%, ↑) on multi-label classification. STROP trained on ImageNet; Stuff-27 = COCO-Stuff-27; Mean = mean of Stuff-27 and VOC mAP. Teacher uses mean-pooled DINO patch features; latent is mean-pooled over the interpreter's spatial latent field; program is attention-pooled over the active program token vectors z_q. Selected variants; for all results please refer to Appendix Table 15. Best non-teacher value per column is **bolded**. Probes follow the original multi-label protocol (1,000 training images; Appendix E).

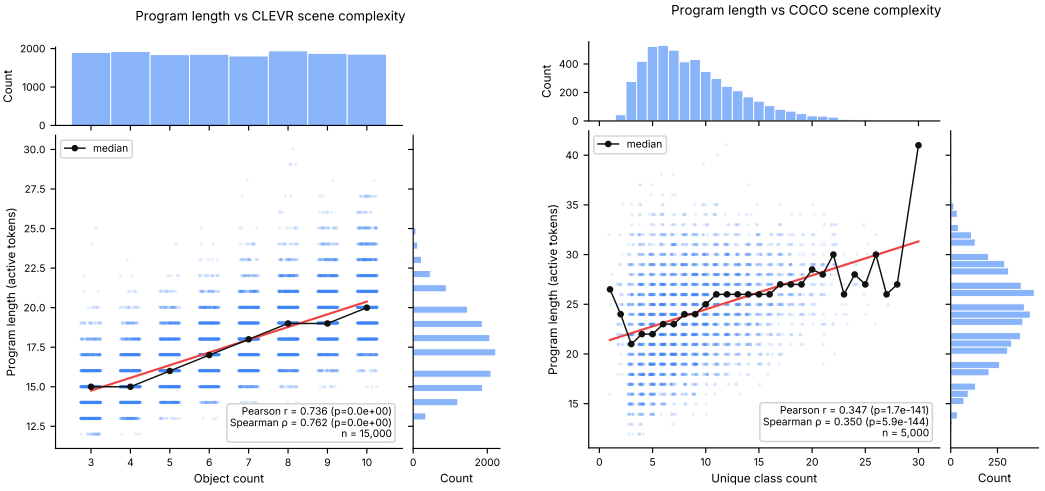
Encoder	Model	Repr.	Stuff-27	VOC	Mean
Teacher (frozen DINO)					
DINO-S	—	teacher	56.2	59.4	57.8
DINO-B	—	teacher	61.2	74.7	67.9
STROP (ours)					
DINO-S	STROP-S	latent	56.6	61.5	59.1
DINO-S	STROP-S	program	34.9	17.8	26.3
DINO-B	STROP-S	latent	**61.7**	74.1	67.9
DINO-B	STROP-S	program	36.1	18.5	27.3
DINO-B	STROP-M	latent	61.6	**75.2**	**68.4**
DINO-B	STROP-M	program	31.3	14.7	23.0

Table 15: Linear probe mAP (%, ↑) on multi-label classification. STROP trained on ImageNet; Stuff-27 = COCO-Stuff-27; Mean = mean of Stuff-27 and VOC mAP. Teacher uses mean-pooled DINO patch features; latent is mean-pooled over the interpreter's spatial latent field; program is attention-pooled over the active program token vectors z_q. Best non-teacher value per column is **bolded**. Probes follow the original multi-label protocol (1,000 training images; Appendix E).

			Multi-label mAP ↑		
Encoder	Model	Repr.	Stuff-27	VOC	Mean
Teacher (frozen DINO)					
DINO-S	—	teacher	56.2	59.4	57.8
DINO-B	—	teacher	61.2	74.7	67.9
DINO-L	—	teacher	59.7	68.4	64.0
STROP (ours)					
DINO-S	STROP-S	latent	55.7	57.6	56.6
DINO-S	STROP-S	program	32.3	15.0	23.7
DINO-S	STROP-M	latent	56.0	59.7	57.9
DINO-S	STROP-M	program	34.1	14.3	24.2
DINO-B	STROP-S	latent	60.2	69.6	64.9
DINO-B	STROP-S	program	32.6	14.0	23.3
DINO-B	STROP-M	latent	**60.5**	**73.4**	**67.0**
DINO-B	STROP-M	program	30.1	12.6	21.3
DINO-L	STROP-M	latent	59.1	68.2	63.6
DINO-L	STROP-M	program	32.2	13.9	23.1

Table 16: Segmentation mIoU (%, ↑) of frozen representations for all model variants, trained on COCO or ImageNet. All rows use the matched 256×256 protocol of Appendix E (full training split, 40k updates, final checkpoint, probe seed 0). Teacher rows probe frozen DINO patch features as an upper bound and do not depend on the training set of STROP. Latent probes the interpreter's spatial latent field; program probes the active program token vectors z_q. Stuff-27 denotes COCO-Stuff-27. The best non-teacher value in each column is **bolded**.

			Trained on COCO				Trained on ImageNet			
Encoder	Model	Repr.	VOC	Stuff-27	ADE20K	Citysc.	VOC	Stuff-27	ADE20K	Citysc.
Teacher (frozen DINO)										
DINO-S	—	teacher	80.2	46.1	42.1	56.6	80.2	46.1	42.1	56.6
DINO-B	—	teacher	84.3	51.6	47.1	62.1	84.3	51.6	47.1	62.1
DINO-L	—	teacher	84.7	52.7	50.5	63.5	84.7	52.7	50.5	63.5
STROP (ours)										
DINO-S	STROP-S	latent	68.8	41.5	29.0	37.0	68.7	40.1	27.9	35.8
DINO-S	STROP-S	program	62.5	39.7	25.3	38.8	64.1	39.4	25.1	**37.9**
DINO-S	STROP-M	latent	61.2	36.3	22.2	29.7	65.7	38.3	24.9	31.5
DINO-S	STROP-M	program	54.1	32.0	20.4	30.4	58.9	35.4	21.2	32.9
DINO-B	STROP-S	latent	**72.9**	**44.8**	**30.7**	40.8	66.6	38.9	24.4	29.7
DINO-B	STROP-S	program	66.6	42.3	28.3	**42.4**	62.0	37.2	22.7	33.9
DINO-B	STROP-M	latent	67.3	41.6	26.8	35.2	69.6	41.8	27.4	33.8
DINO-B	STROP-M	program	61.4	38.0	23.6	37.1	62.0	38.0	24.0	35.6
DINO-L	STROP-M	latent	65.6	41.4	27.4	34.2	**71.3**	**41.9**	**29.0**	35.8
DINO-L	STROP-M	program	60.3	37.0	24.6	37.6	64.0	38.5	25.9	37.6



(a) CLEVR; codebook size 32.

(b) COCO; codebook size 1024.

Figure 13: Program length and scene complexity on CLEVR and COCO. (a) On CLEVR, STROP produces longer programs as the number of objects increases. (b) On COCO, length also grows with the number of annotated classes, but the relation is weaker and noisier (Pearson $r = 0.35$); medians at the highest class counts rest on few images. Class counts are taken from the full-image annotations, while the model sees a 256×256 view.

Table 17: Monocular depth estimation on NYUv2 for all model variants, trained on COCO or ImageNet. All STROP rows and the matched teacher rows use the 256×256 protocol of Appendix E (final checkpoint); RMSE in meters, AbsRel and δ_1 in %, computed over valid depth pixels. Teacher rows probe frozen DINO patch features; the bottom block evaluates the teachers with the official DINOv3 depth protocol as a reference and is not comparable with the matched rows. Latent uses the interpreter's spatial latent field; program uses the active program token vectors z_q. The best non-teacher value in each column is **bolded**.

			Trained on COCO			Trained on ImageNet		
Encoder	Model	Repr.	RMSE↓	AbsRel↓	δ_1 ↑	RMSE↓	AbsRel↓	δ_1 ↑
Teacher (frozen DINO), matched protocol								
DINO-S	—	teacher	0.481	14.7	80.7	0.481	14.7	80.7
DINO-B	—	teacher	0.424	12.8	85.1	0.424	12.8	85.1
DINO-L	—	teacher	0.382	11.2	89.0	0.382	11.2	89.0
STROP (ours)								
DINO-S	STROP-S	latent	0.562	17.4	75.1	0.570	17.7	73.6
DINO-S	STROP-S	program	0.559	17.2	76.0	0.572	16.8	75.1
DINO-S	STROP-M	latent	0.592	19.0	72.5	0.565	17.5	74.6
DINO-S	STROP-M	program	0.583	17.9	74.3	0.578	17.1	74.8
DINO-B	STROP-S	latent	0.527	16.9	76.3	0.571	17.3	74.6
DINO-B	STROP-S	program	0.525	15.3	79.1	0.567	16.8	75.7
DINO-B	STROP-M	latent	0.583	18.5	73.8	**0.546**	**16.1**	**77.2**
DINO-B	STROP-M	program	0.573	17.4	76.0	0.568	16.3	76.1
DINO-L	STROP-M	latent	0.532	16.0	77.9	0.554	17.0	76.4
DINO-L	STROP-M	program	**0.522**	**15.2**	**79.5**	0.560	16.3	77.0
Teacher, official DINOv3 protocol (reference)								
DINO-S	—	teacher	0.405	12.0	87.3			
DINO-B	—	teacher	0.372	11.3	89.6			
DINO-L	—	teacher	0.351	10.2	91.9			

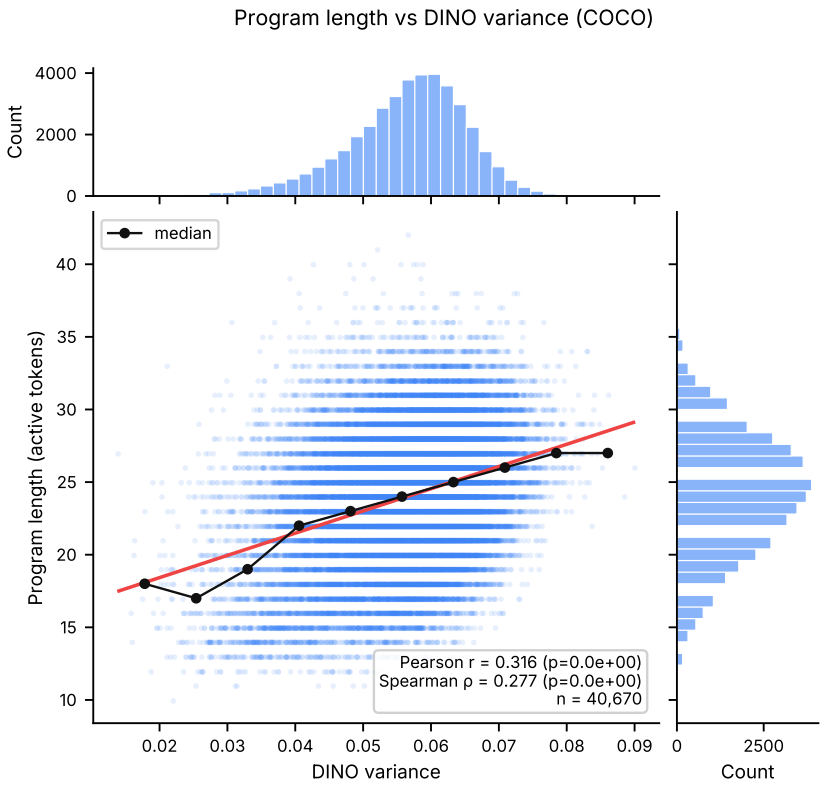


Figure 14: Program length and scene complexity analysis on COCO using variance of DINO embeddings as a proxy for scene complexity.

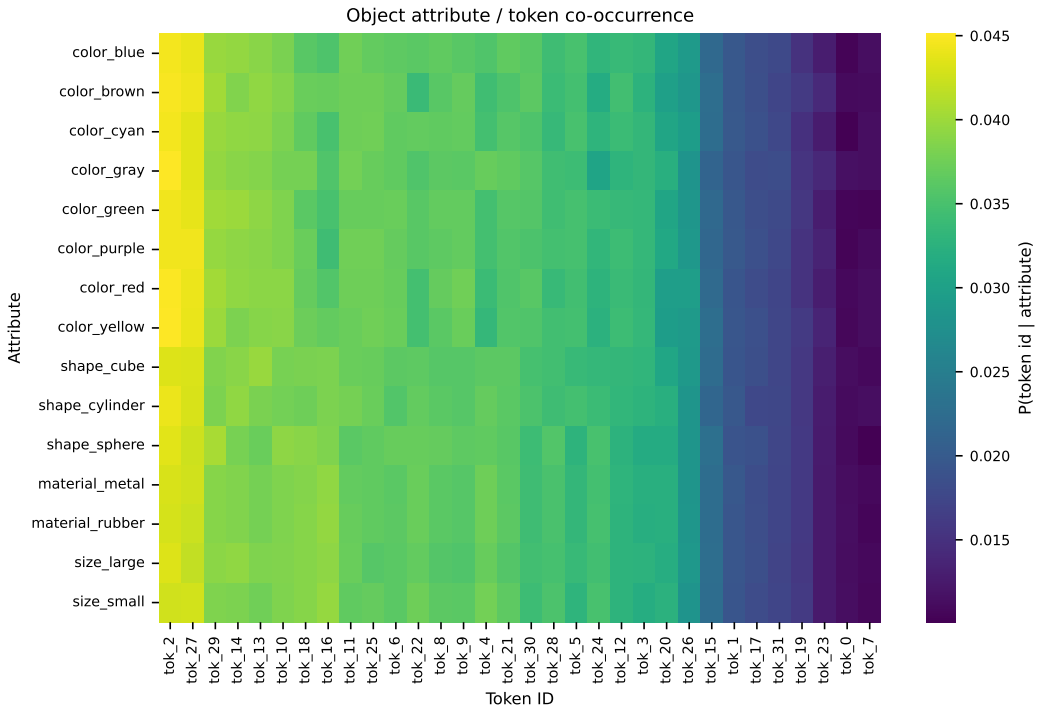


Figure 15: Contingency heatmap for CLEVR attributes and token ids; codebook size 32.

L CLEVR

We train additional variants on CLEVR (Johnson et al., 2016) at 128×128 resolution to study codebook sizing and program structure on a controlled scene domain. All CLEVR models use a DINO-S encoder with STROP-S generator/interpreter ($d_{\text{model}}=256$, 8 layers), and $K=32$ program tokens. The only variable is codebook size $V \in \{32, 64, 128\}$; all other hyperparameters match the main configuration (Table 5). Training runs for 5×10^5 steps with batch size 128 on a single H100 ($\sim$27h each).

Table 18 reports results. All three codebook sizes achieve $R^2 > 0.93$, confirming that the architecture can faithfully represent CLEVR scenes through a short discrete program. The $V=128$ variant reaches the highest R^2 (.948) and SSIM (.787). Notably, $V=64$ shows a dip in alignment ($R^2=.931$) that is explained by its truncated codebook utilization equal to 95%, meaning nearly every code is active within the adaptive prefix and the vocabulary has little remaining capacity. At $V=32$ the codebook is larger relative to scene complexity, giving more headroom (91%), while at $V=128$ the vocabulary is comfortably under-utilized (86%).

Curriculum vs. fixed-length programs. To isolate the cost of adaptive-length training, we compare the curriculum variants ($V \in \{64, 128\}$) against matched fixed-length baselines that always use all $K=32$ tokens with no halting mechanism.

Table 19 shows that fixed-length programs outperform curriculum variants on R^2, with gains of .018–.030. Fixed-length training is also $\sim10\%$ faster in wall time (24.7–24.9h vs. 27.3h) because it avoids the halting head forward pass, oracle target computation, and phase-dependent loss modulation. This gap is expected: the curriculum solves a strictly harder problem—learning *when to stop* in addition to *what to encode*—while the fixed-length model always has access to all K tokens.

The curriculum's value is not in raw metric improvement but in the capability it unlocks: variable-length programs that allocate fewer tokens to simple scenes and more to complex ones (see Fig. 13). On CLEVR, where scene complexity varies from 3 to 10 objects, learned program lengths range from 8 to 32 tokens, enabling a $\sim2.5\times$ compression ratio on the simplest scenes with no manual tuning.

Symbolic content and grounding. Table 20 tests which scene properties can be read linearly from the discrete tokens of the CLEVR models, and Figure 16 measures how well individual tokens localize objects. Linear readouts of the CLEVR tokens predict object count and attributes better when they are given token positions, and shuffling the token order removes that advantage (Table 20). On these symbolic targets, fixed-length programs are as informative as adaptive ones. The most influential tokens overlap individual objects, and the overlap grows as more tokens are included (Fig. 16).

Table 18: CLEVR codebook-size variants (DINO-S / STROP-S, $K=32$, 128×128).

V	Cos↑	R^2↑	SSIM↑	LPIPS↓	V%
32	.971	.943	.775	.173	73.5
64	.965	.931	.756	.187	**94.7**
128	**.973**	**.948**	**.787**	**.162**	77.8

Table 19: Curriculum vs. fixed-length training on CLEVR (DINOv3-S / STROP-S, K=32, 128×128, 500k steps). Fixed-length models use all 32 tokens; curriculum models learn an adaptive prefix via the four-phase schedule (Sec. 2.2).

V	Mode	R^2↑	V%	Time (h)
64	Fixed	**.961**	88.6	**24.7**
64	Curriculum	.931	94.7	27.3
128	Fixed	**.966**	94.2	**24.9**
128	Curriculum	.948	77.8	27.3

Table 20: What the discrete tokens encode on CLEVR. Linear predictors of object count (MAE, ↓) and of 39 attribute-presence labels (mean accuracy in %, ↑) are trained on the tokens of frozen CLEVR models (DINO-S / STROP-S, 128×128, programs of up to 32 codes; 90k / 5k / 5k train / validation / test images; epoch selected on validation, test reported). Bag uses code counts only; Bigram adds counts of adjacent code pairs; Pos. adds code-by-position indicators; Ord. combines both; Shuf. is Ord. with the active tokens randomly permuted. The best readout in each row and metric group is **bolded**: position-aware readouts are best for every model, and shuffling the order removes their advantage. Fixed-length programs are as informative as adaptive ones on these symbolic targets.

	Object count MAE↓					Attribute accuracy (%)↑				
Model	Bag	Bigram	Pos.	Ord.	Shuf.	Bag	Bigram	Pos.	Ord.	Shuf.
Adaptive, V=32	0.943	0.900	0.705	**0.700**	0.956	76.4	76.1	**78.0**	77.6	75.5
Adaptive, V=64	1.063	1.025	**0.742**	0.764	1.083	76.2	74.4	**76.5**	75.2	74.5
Adaptive, V=128, noncausal	0.932	0.928	**0.686**	0.721	0.974	76.4	75.2	**78.6**	77.5	74.3
Fixed K=32, V=64	0.876	0.851	**0.613**	0.629	0.926	76.6	76.1	**80.9**	80.1	74.7
Fixed K=32, V=128	0.907	0.916	**0.574**	0.634	1.004	76.9	75.7	**81.7**	80.3	74.0

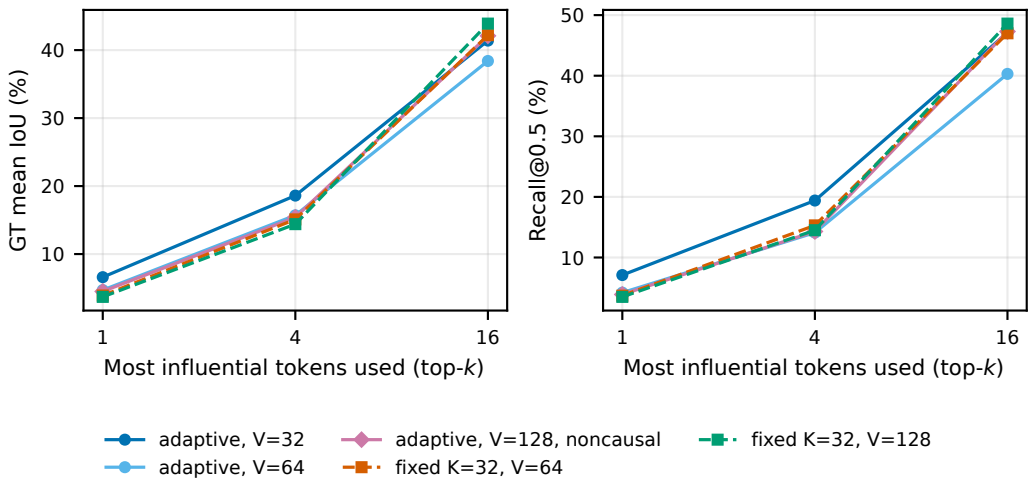


Figure 16: **Token grounding on CLEVR.** Overlap between the influence maps of the k most influential program tokens and the ground-truth object masks on 5,000 CLEVR test images, for adaptive and fixed-length programs (protocol in Appendix E). Left: mean IoU of each object with its matched token; right: fraction of objects matched with IoU ≥ 0.5. Comparisons are relative: no random-token baseline is reported.

M LLM SCENE DECODING FROM LEARNED VISUAL TOKENS

We evaluate whether frontier large language models can decode continuous CLEVR scene descriptions directly from discrete codebook tokens produced by a learned visual tokenizer. Given a sequence of integer codes from a DINO-S encoder (with codebook sizes $V \in \{32, 64, 128\}$), the models must predict the full scene structure as a JSON string, including object count, their attributes and continuous (x, y) positions. We benchmark three frontier LLMs (Qwen3.5-397B-A17B, DeepSeek-V3.1-vLLM-2, and GPT-OSS-120B, all from Hugging Face) against a *Naive Baseline*, which uses no language model and predicts a fixed "average scene" derived entirely from the statistical distributions of the few-shot pool. This baseline establishes a lower bound measuring how much of a scene can be guessed from prior dataset statistics alone, without decoding the actual visual tokens. All models are evaluated on 15 held-out test images using a 50-shot random in-context learning setup, with one retry permitted on parse failure.

Key findings. Two observations stand out. First, the stronger frontier models (Qwen3.5 and DeepSeek) decode *spatial* structure substantially better than the naive baseline across all codebook sizes, with DeepSeek achieving the best count error (0.93 at $V{=}64$) and the best position errors (3.15 MSE / 1.84 MAE at $V{=}128$). Second, and more strikingly, *no model surpasses the naive baseline's attribute accuracy of* 56.8%: while the models recover "where" and "how many" (Count Err), binding tokens to discrete textual properties such as color or material remains beyond their reach without dataset priors. The smaller GPT-OSS-120B exhibits a different failure mode entirely, with count errors (~ 2.4–2.8) at or above the naive baseline despite achieving competitive position errors at $V{=}32$, suggesting it leans more heavily on memorized layout statistics than on token-level decoding.

Attribute–token co-occurrence. Figure 15 shows the contingency table between CLEVR object attributes (color, shape, material, size) and assigned token IDs. The heatmap reveals no strong systematic correspondence—consistent with the observation that attribute identity is not yet reliably encoded in individual codebook entries. Understanding whether attribute information is distributed across token combinations rather than localized in single tokens is an important direction for future work.

We document the exact prompt used to elicit scene predictions from each frontier LLM. All models were queried at temperature 0.0 with 50 random in-context examples preceding the test query. To minimize token consumption across the 50-shot context, attribute values are emitted in an abbreviated form, with a per-prompt legend instructing the model to use the short codes.

Attribute abbreviations. Object attributes are remapped to single- or two-character codes, summarized in Table 22. The same scheme is applied uniformly to the few-shot examples, the schema description in the prompt, and the expected model output, so that the model never needs to translate between long and short forms.

Prompt template. The prompt below is templated with the codebook size V (substituted for `{codebook_size}`), the maximum code index $V - 1$ (substituted for `{codebook_max}`), the inline abbreviation legend (`{abbr_legend}`), and an optional reasoning suffix (`{think_suffix}`) used only for DeepSeek-V3.1 to elicit step-by-step reasoning prior to JSON emission.

Table 21: Scene decoding performance of frontier LLMs compared to a statistical baseline. Parse Fails denote outputs with invalid JSON structures (max one retry). Metrics are averaged across 15 test examples. Best LLM result per metric in **bold**.

Model	V	Parse Fails	Count Err ↓	Attr Acc ↑	Pos MSE ↓	Pos MAE ↓
Naive Baseline	N/A	0 / 15	2.27	0.568	3.84	2.22
GPT-OSS-120B	32	1 / 15	2.36	0.469	2.45	1.71
GPT-OSS-120B	64	3 / 15	2.83	0.536	3.26	1.87
GPT-OSS-120B	128	1 / 15	2.36	0.524	4.61	2.13
Qwen3.5-397B-A17B	32	0 / 15	1.43	0.510	3.22	1.90
Qwen3.5-397B-A17B	64	0 / 15	1.00	0.458	3.37	2.00
Qwen3.5-397B-A17B	128	0 / 15	1.53	0.505	3.17	1.91
DeepSeek-V3.1-vLLM-2	32	1 / 15	1.36	0.513	3.68	2.11
DeepSeek-V3.1-vLLM-2	64	0 / 15	**0.93**	**0.543**	3.64	1.97
DeepSeek-V3.1-vLLM-2	128	0 / 15	1.13	0.487	**3.15**	**1.84**

Table 22: Attribute abbreviation scheme used in both few-shot examples and model outputs.

Color	**Shape**	**Size**	**Material**
gray → `gr`	cube → `cu`	small → `S`	rubber → `R`
red → `r`	sphere → `sp`	large → `L`	metal → `M`
blue → `bl`	cylinder → `cy`		
green → `gn`			
brown → `br`			
purple → `pr`			
cyan → `cy`			
yellow → `y`			

Decoding prompt

```
Predict a CLEVR scene (3-10 objects) from integer codes produced
by a visual tokenizer (codebook size={codebook_size}, codes in
[0,{codebook_max}]).
Output compact JSON using short keys and abbreviated values:
{abbr_legend}
Schema: {"o":[{"s":"<shape>","c":"<color>","z":"<size>",
"m":"<mat>","p":[x,y]},...]}
Position: x,y in [-3,3], 2 decimal places.
Example: {"o":[{"s":"cu","c":"r","z":"L","m":"M","p":[1.23,-0.45]}]}
JSON only. No prose.{think_suffix}
```

Reasoning suffix. For DeepSeek-V3.1 and GPT-OSS-120B, the placeholder `{think_suffix}` is replaced with the following string; for Qwen3.5 it is left empty.

DeepSeek reasoning suffix

```
Before your final answer, reason step-by-step inside <think> and
</think> tags. Then output the JSON.
```

Decoding settings. All models were queried with temperature 0.0 to encourage deterministic outputs. A single retry was permitted on JSON parse failure, after which the example was counted as a parse failure in Table 21.