GaussianVision: Vision-Language Alignment from Compressed Image Representations using 2D Gaussian Splatting

Yasmine Omri yomri@stanford.edu
Connor Ding czsding@stanford.edu
Tsachy Weissman* tsachy@stanford.edu
Thierry Tambe* ttambe@stanford.edu

*Equal advising

Department of Electrical Engineering, Stanford University

Contents

A. Additional Background

A.1. Contrastive Language-Image Pre-training

Vision–language alignment is commonly achieved through Contrastive Language–Image Pre-Training (CLIP), which trains a dual-encoder architecture on image–caption pairs. A text encoder and a vision encoder are optimized jointly so that

embeddings of aligned pairs are pulled closer, and mismatched pairs pushed apart under a contrastive loss [3]. CLIP has become the standard paradigm, demonstrating strong performance across a wide range of downstream multimodal tasks. The CLIP framework spans a family of models from lightweight 25M-parameter encoders to larger 300M+ parameter variants such as ViT-L/14, all of which require hundreds of millions of training samples and significant compute to reach high-quality alignment.In this work, we retain the CLIP training recipe, but investigate whether 2D Gaussian Splatting can serve as an alternative visual substrate. Using the OpenCLIP implementation [2], we include comparisons of RGB- and 2DGS-based encoders under identical training conditions, isolating the effect of the representation itself. Our approach further reuses frozen RGB-pretrained backbones with lightweight splat-aware input modules, enabling efficient adaptation while substantially reducing trainable parameters.

A.2. CLIP Benchmark.

The *CLIP Benchmark* is an evaluation suite for CLIP-like vision-language models, focusing on zero-shot performance across diverse tasks. In zero-shot evaluation, a pre-trained model is tested on new tasks without fine-tuning, using only natural language prompts for each target class. This provides a proxy measure of the encoder's generalization ability and representation quality dblp.org dblp.org . CLIP models [3] have demonstrated strong zero-shot classification results on numerous image recognition datasets by simply using the class names or descriptions as text inputs, indicating the efficacy of their learned representations in transferring to unseen tasks.

The CLIP Benchmark [1] encompasses a broad range of vision tasks to thoroughly assess such generalization. It includes standard zero-shot image classification datasets spanning various domains, as well as multi-label classification and image-text retrieval tasks. Notably, it incorporates all 19 tasks from the *Visual Task Adaptation Benchmark (VTAB)* [5], a suite of classification tasks designed to evaluate general visual representations across heterogeneous domains. VTAB's tasks are grouped into three categories, *Natural*, *Specialized*, and *Structured*,covering everything from everyday natural images to remote sensing and medical images, and even synthetic tasks that require counting objects or estimating distances. In addition to the VTAB tasks, the CLIP Benchmark evaluates models on even more complex zero-shot classification datasets. Table 1 summarizes the key datasets included, the type of task each represents, number of classes, and a brief description of what capability or scenario each dataset tests.

Dataset	**Task/Domain Type**	**# Classes**	**Description of Task**
ImageNet-1k	Object classification (natural)	1000	Standard ImageNet object recognition benchmark (ILSVRC-2012).
ImageNet-v2	Object classification (shifted)	1000	Re-collection of ImageNet validation set for distribution shift evaluation.
ImageNet-R	Object classification (renditions)	200	Contains renditions of ImageNet categories in artistic or abstract styles (cartoons, paintings, sculptures).
ImageNet-Sketch	Object classification (sketches)	1000	Sketch drawings of ImageNet classes, testing robustness to line-art style inputs.
ObjectNet	Object classification (viewpoints)	113	Photos of objects from unusual viewpoints/backgrounds; tests robustness to pose and context changes.
ImageNet-A	Object classification (adversarial)	200	"Naturally adversarial" real-world images curated to fool standard ImageNet models (hard OOD test).
CIFAR-10	Object classification (low-res)	10	Tiny 32×32 natural images of 10 object classes (vehicles, animals).
CIFAR-100	Object classification (low-res)	100	Tiny 32×32 images across 100 fine-grained object categories.
MNIST	Digit classification	10	Handwritten digit images (0–9) in grayscale.
Oxford Flowers-102	Fine-grained classification	102	Photographs of flowers; classify 102 flower species.

Continued on next page

Table 1 – *Continued from previous page*

Dataset	Task/Domain Type	# Classes	Description of Task
Stanford Cars	Fine-grained classification	196	High-resolution photos of cars labeled by make, model, and year.
SVHN	Digit classification (street images)	10	Street View House Numbers cropped into digits 0–9.
FER-2013	Facial emotion recognition	7	Low-resolution grayscale face images labeled with 7 expression categories.
Rendered SST-2	Text sentiment (OCR)	2	Images of text rendered from SST-2 movie reviews; classify sentiment.
Oxford-IIIT Pets	Fine-grained classification	37	Photos of 37 cat and dog breeds.
Caltech-101	Object classification (varied)	101	Images of 101 diverse object categories (plus background).
PASCAL VOC 2007 (Clf.)	Object presence (multi-label)	20	Detect presence/absence of 20 object categories in natural scenes.
SUN397	Scene classification	397	Scene recognition across 397 indoor/outdoor categories (park, office, bedroom, . . .).
FGVC Aircraft	Fine-grained classification	100	Recognition of aircraft model variants across 100 categories.
Country211	Geographic location classification	211	Predict the country where the image was taken (211 possible labels).
Describable Textures (DTD)	Texture classification	47	Classify images into 47 describable texture attributes (striped, dotted, grooved).
GTSRB	Traffic sign recognition	43	Images of 43 German traffic sign classes captured from real-world road scenes.
STL-10	Object classification	10	Larger 96×96 version of CIFAR-like images with 10 object classes.
Diabetic Retinopathy	Medical image classification	5	Classify retinal fundus images into 5 disease severity levels.
EuroSAT	Satellite image classification	10	10 land-use classes from Sentinel-2 satellite imagery (forest, farmland, river, . . .).
RESISC45	Aerial scene classification	45	Remote sensing scenes across 45 diverse aerial categories.
PatchCamelyon (PCam)	Medical image classification	2	Microscopy patches labeled as tumor vs. normal.
CLEVR Counts	Synthetic reasoning (counting)	8	Count objects in synthetic CLEVR 3D scenes (8 count categories).
CLEVR Distances	Synthetic reasoning (spatial)	6	Predict relative distance of closest object in CLEVR scenes (6 bins).
dSprites Orientation	Synthetic visual factor (orientation)	40	Classify rotation angle of a simple 2D shape.

Continued on next page

Table 1 – *Continued from previous page*

Dataset	Task/Domain Type	# Classes	Description of Task
dSprites Position	Synthetic visual factor (position)	32	Classify object position on a grid (32 discrete locations).
SmallNORB Elevation	Synthetic visual factor (3D pose)	9	Estimate camera elevation angle over 9 categories.
SmallNORB Azimuth	Synthetic visual factor (3D pose)	18	Estimate camera azimuth angle over 18 viewpoints.
DMLab	Synthetic visual reasoning (depth)	6	Classify 3D maze frames into 6 depth-related categories.
KITTI Distances	Driving vision (depth estimation)	4	Classify distance to nearest vehicle into 4 discrete range bins.

Table 1. Datasets covered in the CLIP zero-shot image classification benchmark [1] Each dataset's task type, number of classes, and a brief description of what is evaluated are given.

In summary, the CLIP Benchmark evaluates zero-shot transfer performance on a comprehensive collection of vision datasets. High accuracy across these diverse tasks (without task-specific training) indicates that a model has learned versatile and general visual features. Modern CLIP models (including large OpenCLIP variants trained on LAION-5B) indeed show strong performance on natural and specialized image classification tasks.

B. Algorithmic Optimizations: Additional Details

B.1. Structured Initialization

Figure 1 demonstrates the effectiveness of our structured initialization. Unlike random initialization, which begins from a noise-like configuration, our spatial layout-based initialization produces a meaningful coarse approximation at iteration 0. This strong prior accelerates optimization and yields consistently higher fidelity throughout training, as reflected in the PSNR curves in Fig. 1 (right). By iteration 2000, structured initialization produces sharper, more stable reconstructions under identical Gaussian budgets, thus unlocking higher asymptotic perceptual quality.

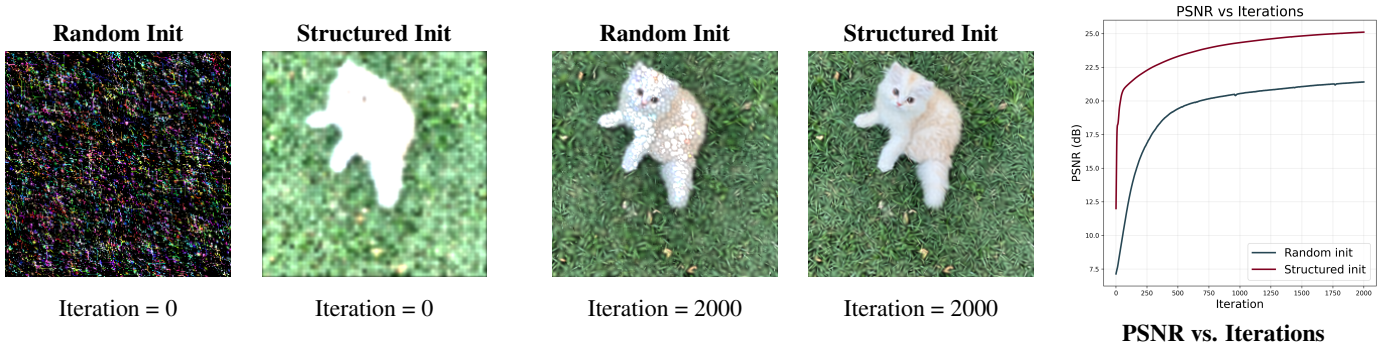


Figure 1. Structured vs. Random Initialization for 2DGS Fitting. Left: qualitative comparison at iteration 0 and iteration 2000 under random and structured initialization (using 3136 Gaussian points). Right: PSNR evolution during optimization. Structured initialization rapidly improves fidelity and consistently outperforms random initialization.

B.2. Pruning

Figure 2 summarizes our pruning study and provides additional detail beyond Sec. 4.4. To generate the contour maps (left), we sweep over luminance thresholds $\tau_{\text{th}} \in \{0, 0.01, 0.05, 0.10, 0.15, 0.20, 0.25\}$ and color regularization weights

$\lambda_{\text{reg}} \in \{0, 10^{-7}, 5 \times 10^{-7}, 10^{-6}, 5 \times 10^{-6}, 10^{-5}\}$ for each Gaussian budget $\{400, 900, 1600, 3136\}$. For every $(\lambda_{\text{reg}}, \tau_{\text{th}})$ configuration, we fit the model for 2000 iterations on 100 Mini-ImageNet samples (224×224), record the resulting sparsity level, and compute $\Delta\text{PSNR} = \text{PSNR}_{\text{post}} - \text{PSNR}_{\text{pre}}$ to quantify the fidelity impact of pruning. The contour maps therefore visualize the *averaged* relationship between pruning aggressiveness and reconstruction stability.

As shown in Fig. 2, pruning increases smoothly with stronger regularization penalties and larger luminance thresholds, while ΔPSNR remains small for moderate to large point budgets, which tend to tolerate more aggressive pruning. On the right, we present the luminance histogram, reconstructed image, and Gaussian splat visualization for a 3136-point GS fit (2000 iterations): Top corresponds to λ_{reg}=0, τ_{th}=0 (0% pruned: PSNR = 37.43). Bottom corresponds to λ_{reg}=10^{-6}, τ_{th}=0.05 (23.72% pruned: PSNR = 31.1).

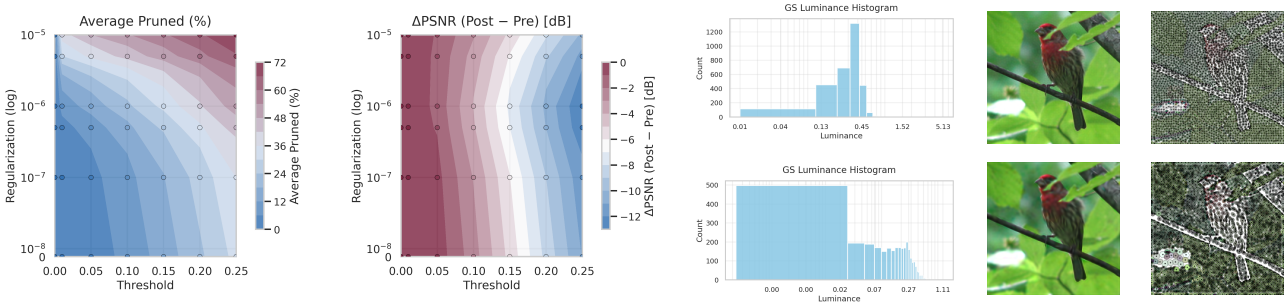


Figure 2. Luminance behavior with color regularization and reconstruction quality across settings. Left: contour plots visualizing pruning ratio and ΔPSNR across threshold and regularization. Middle: luminance histograms before/after regularization. Right: Gaussian Splat and original RGB comparisons.

C. CUDA Optimizations: Profiling Results

To support large-scale 2DGS preprocessing, we extended the public `gsplat2d` codebase (from [6] and further developed by [7]) with full batch-parallel execution, enabling thousands of images to be fitted concurrently on a single GPU. The baseline codebase codebase is structured as follows: the forward pass is decomposed into two phases: (1) a projection stage, wherein each Gaussian primitive is projected onto a 2D grid of pixels. The grid is tiled across thread blocks, where each block corresponds to a 2D tile of the output image. Within a block, individual threads are assigned to pixels, computing Gaussian weights and contributions in parallel. This design ensures that Gaussian evaluation is parallelized across both the spatial dimension (pixels) and across different Gaussians, and (2) a rasterization stage, wherein the projected Gaussians are accumulated to produce the final output image at each iteration. This stage handles blending and weighting of Gaussians for each pixel. Unlike projection, where computation is Gaussian-centric, rasterization is pixel-centric, with threads accumulating contributions into shared output buffers. The backward pass is handled by a dedicated kernel that computes gradients of the loss with respect to Gaussian parameters (positions, covariances, colors). Workload partitioning is inverted relative to the forward pass: each block is assigned to a 2D tile of the input grid, and threads within the block compute pixel-level derivatives. This tiling strategy balances workload across warps and ensures efficient coalesced memory access when accumulating gradients.

While our main paper reports overall throughput improvements after our optimizations, here we include additional profiling details on a H100 GPU using NVIDIA Nsight Systems to better illustrate kernel behavior.

We report profiling results of our revamped 2DGS CUDA pipeline under the configuration: 4000 Gaussian points and 2000 iterations on 224×224 mini-ImageNet images. Table 2 reports the total fitting time per batch, as well as Nsight Systems' breakdown of GPU active time into compute kernels vs. memory operations.

Not surprisingly (by Amdahl's law), the wall-clock fitting time grows sublinearly with batch size, helping us identify cost-effective operating points when scaling to the full **12.8M-image Datacomp dataset**. When fitting at scale, for each point budget (400, 900, 1600, 3136, and 4000), we choose a distinct batch size that maximizes throughput; the final batch sizes are reported in the Data Pre-processing section.

Batch size	Total fit time	Kernel	Memory	Speedup
1	7 s	91.9%	8.1%	1
128	13 s	97.9%	2.1%	1.86
256	22 s	92.7%	7.3%	3.14
512	39 s	93.2%	6.8%	5.57
1024	1m 16s	89.9%	10.1%	10.86
2048	2m 23s	94.6%	5.4%	20.43
4096	5m 27s	81.2%	18.8%	46.71

Table 2. **Nsight Systems profiling of our batch-parallel 2DGS CUDA kernels.** Reported numbers correspond to 4000 points and 2000 iterations per image. Kernel/Memory percentages reflect the fraction of *active* GPU time.

D. Vision-Language Alignment at Scale from 2D Gaussian Splat Representations: Additional Details

D.1. Training Data Pre-processing through 2D Gaussian Splatting

To train our models efficiently, we pre-fit the entire 12.8M-image DataComp dataset offline using 2D Gaussian Splatting. Although per-image fitting is significantly faster with 2DGS compared to classical INRs, it remains too costly to perform online. Offline fitting allows us to amortize this one-time cost across all downstream experiments and iterate rapidly on model design.

For each configuration in Table 3, the "Config." entry corresponds to the number of Gaussian points per image. We run each fit for **2000 iterations**, a conservative setting chosen, because PSNR consistently plateaus before that point. Batch sizes were selected via profiling to maximize GPU throughput for each configuration. Total GPU hours (total time to fit the entire 12.8M dataset) were obtained by dividing the full dataset size by the measured wall-clock time per batch.

We additionally report dataset statistics for each configuration. Specifically, we collect the empirical means and standard deviations of the covariance components $(\mathrm{cov}_{xx}, \mathrm{cov}_{xy}, \mathrm{cov}_{yy})$ and RGB channels from a 1M-image subset. These statistics support the normalization strategies explored later in the appendix. Figure 3 visualizes the perceptual reconstructions across configurations, highlighting that, even when perceptual fidelity degrades at lower point counts (eg. 400 Gaussian points), the semantic signal may remain largely intact for many downstream tasks.

Config.	Bsz	Total GPU hrs	Data Statistics			
			cov mean	cov std	rgb mean	rgb std
400	4096	25.6	[3.40, −0.01, 3.38]	[0.37, 1.65, 0.39]	[0.77, 0.74, 0.73]	[1.52, 1.48, 1.50]
900	2048	42.8	[2.60, −4.45e−3, 2.58]	[0.43, 1.19, 0.44]	[0.66, 0.63, 0.62]	[0.97, 0.96, 0.96]
1600	2048	35.9	[2.05, −2.53e−3, 2.07]	[0.32, 0.77, 0.30]	[0.58, 0.56, 0.55]	[0.59, 0.58, 0.59]
3136	1024	53.1	[1.63, −1.48e−3, 1.65]	[0.30, 0.58, 0.29]	[0.49, 0.48, 0.46]	[0.46, 0.45, 0.46]

Table 3. **Gaussian Splat fitting configurations.** Each configuration specifies the number of Gaussian points per image. Batch sizes were selected using CUDA profiling for maximal throughput. Total GPU hours are estimated by dividing the 12.8M-image dataset size by the measured time per batch. We report dataset-level statistics (mean and std) for covariance components $(\mathrm{cov}_{xx}, \mathrm{cov}_{xy}, \mathrm{cov}_{yy})$ and RGB channels, computed over 1M fitted images.

D.2. Model Architecture Details

Our GaussianSplatEncoder maps a set of N Gaussian splat primitives $(x, y, \mathrm{cov}_{xx}, \mathrm{cov}_{xy}, \mathrm{cov}_{yy}, R, G, B)$ to a CLIP-compatible embedding. The number of Gaussians N (e.g., 196, 400, 900, 1600, 3136) and the number of latent tokens M (e.g., 196 or 98) are fully configurable. The GaussianSplatEncoder is composed of a GSStem (Perceiver Resampler-based architecture), followed by a transformer backbone and projection. Each 8-D Gaussian point is normalized (XY scaling, signed–log covariance) and Fourier-encoded with 6 frequencies (Fourier dim $= 4 \times 6$). A linear layer projects the input to a **128-d** Perceiver space. The stem uses learnable latent queries of shape $M \times 128$ and applies a stack of **4 cross-attention layers**, each with 4 heads. The output is projected to the CLIP width of 512. The resampled M tokens are pre-pended with a learned CLS token

(a) Original (b) GS: 3136 pts (c) GS: 1600 pts (d) GS: 900 pts (e) GS: 400 pts

Figure 3. **Perceptual reconstructions across Gaussian point budgets.** Although perceptual fidelity decreases at lower point counts (higher compression ratios), the underlying semantic content often remains largely preserved, enabling effective downstream learning.

and fed to a transformer backbone with width = 512, layers L (default 12), heads H (default 8), MLP ratio = 4, RMSNorm pre-normalization. The CLS output is layer-normalized and projected to the final 512-d embedding.

Algorithm 1 GaussianSplatEncoder (forward)

Require: $X \in \mathbb{R}^{B\times N\times 8}$, number of latents M

$X \leftarrow \text{normalize\_xy\_cov\_rgb}(X)$
if Fourier enabled **then**
  $X \leftarrow \text{fourier\_encode}(X)$
end if
$X \leftarrow \text{Linear}_{8+\text{fourier\_dim}\to 128}(X)$
$\ell \leftarrow$ learnable_latents $(M \times 128)$, broadcast to batch
for $t = 1 \ldots 4$ **do**
  $\ell \leftarrow \text{CrossAttn}(\ell, X)$
end for
$T \leftarrow \text{Linear}_{128\to 512}(\ell)$
prepend CLS token: $Z = [\text{CLS}; T]$
$Z \leftarrow \text{RMSNorm}(Z)$
$Z \leftarrow \text{Transformer}_{L,H,\text{MLP}=4}(Z)$
$\mathbf{h} \leftarrow \text{LN}(Z_{\text{CLS}})$
return $\text{Linear}_{512\to 512}(\mathbf{h})$

D.3. Zero-Shot Performance Results

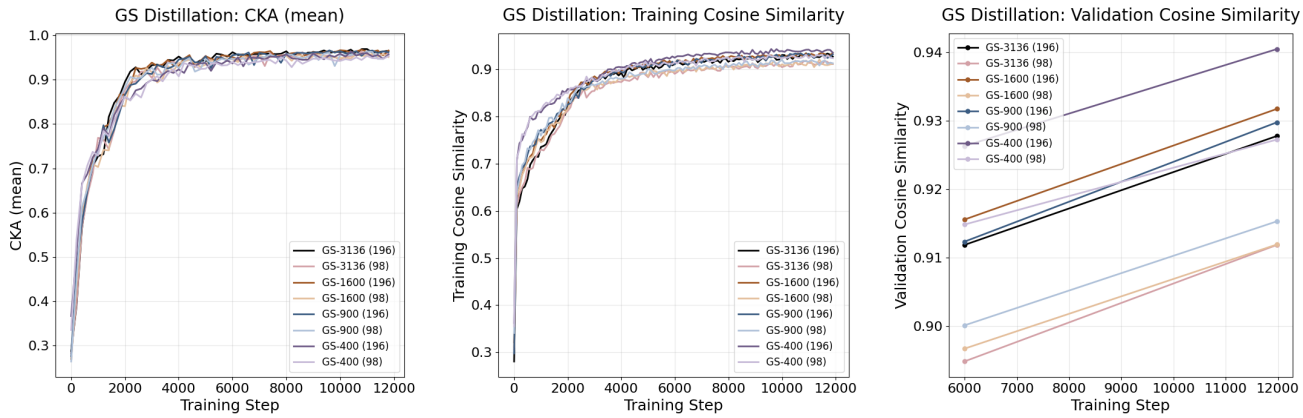
In the main paper, we presented our zero-shot accuracies in bar-plot form; for completeness, the full table of results is provided in Table 4 (breakdown of relative accuracies provided in Table 5). We also include the distillation stage metrics in Fig. 4 as well as the training loss and top-1 accuracy vs. training steps in 5. We recall that RGB ViT-B/16 (196-tokens) is used as the baseline.

Across the 38 benchmark datasets, GS encoders exhibit competitive and often superior performance to the RGB ViT baseline. Notably, 19 datasets are best solved by one of the GS variants, demonstrating that the 2DGS representation preserves strong semantic information despite its aggressive compression. Even the 400-point configuration retains a surprising amount of semantic signal.

We also observe two datasets where GS underperforms more substantially. These cases appear tied to stronger distribution shifts, where the ViT RGB model exhibits more robust generalization and higher absolute accuracy. As this work represents the first systematic exploration of 2D Gaussian Splatting for vision-language alignment, we expect that more mature GS-native architectures will further improve robustness under distribution shift, narrowing the performance gap with RGB-based models.

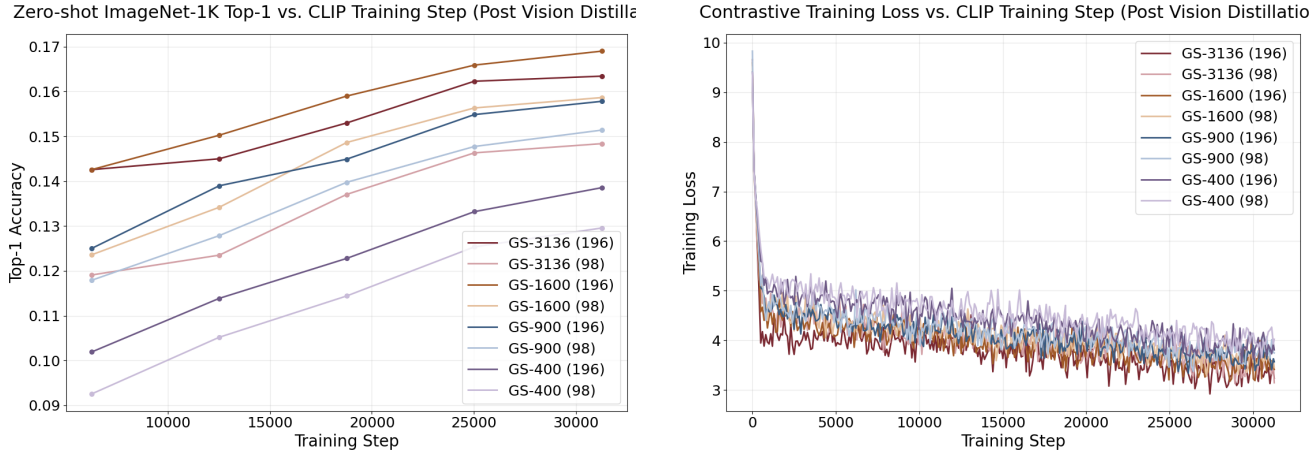
D.4. RGB Baselines: Additional Details

ViT-B/16 (Small). To keep computational cost manageable during large-scale experimentation, we use a reduced-width ViT-B/16 variant whose hidden dimension is lowered from 768 to 512. Figure 6 shows that this "ViT-B/16 (Small)" con-



(a) CKA Mean (b) Train Cosine (c) Val Cosine

Figure 4. Diagnostic distillation metrics illustrating representation alignment during CLIP training.



(a) Zero-shot Top-1 after 1 Epoch (b) Training Loss

Figure 5. Training loss and top-1 accuracy vs. training steps for our GS encoders.

figuration maintains comparable training loss and zero-shot accuracy to the standard 768-width model, confirming that the reduced-width encoder is an appropriate baseline for our GS comparisons.

Token-Count Reduction for RGB Baselines. Throughout the main paper, the RGB ViT-B/16 encoder uses the standard 196-token patch embedding. For a fair comparison to our GS encoders, which natively support fewer latent primitives, we benchmarked 3 adaptations of ViT-B-16 (Small) operating with 98 tokens. This reduction is performed *at the tokenization stage* by merging consecutive patch embeddings after positional encoding, rather than merging the full encoder output tokens.

Because retraining a true 98-token ViT-B/16 from scratch (using larger patch sizes) is computationally expensive, we explore three adaptation settings: (1) no fine-tuning (direct token merging), (2) 1-epoch fine-tuning, and (3) 5-epoch fine-tuning, as shown in Table 6 and Fig. 7. This illustrates that both RGB and GS tokenizations can both exhibit resilience to reduced token counts, with the highest 98 token relative accuracy achieved by the GS pipeline.

D.5. Model Architecture and Training Recipe Studies

Across the course of this work, we conducted extensive experimental sweeps to evaluate architectural choices, training recipes, and design decisions that influence the representational capabilities of 2D Gaussian Splats. This section summarizes a focused subset of these studies, highlighting the settings that most strongly affected performance and optimization behavior.

Dataset	RGB ViT-B/16 (Small)		GS 3136		GS 1600		GS 900		GS 400	
	196	98	196	98	196	98	196	98	196	98
cars	16.88	9.35	17.73	15.33	**17.95**	15.96	15.87	14.60	11.85	10.36
country211	**3.79**	3.01	3.45	3.01	3.42	3.34	3.76	3.71	3.10	3.03
fer2013	**17.41**	17.34	7.19	8.62	3.46	4.49	4.79	4.08	7.33	8.99
fgvc_aircraft	1.38	1.59	**1.86**	1.74	1.11	1.17	1.07	1.81	1.11	1.65
gtsrb	7.25	7.81	9.44	**9.96**	9.44	8.31	9.27	8.41	8.43	9.52
imagenet-a	**4.00**	2.79	3.51	3.04	3.69	3.16	2.00	1.56	3.21	2.99
imagenet-o	**30.20**	22.95	27.90	27.00	28.40	27.00	27.00	26.60	23.90	24.80
imagenet-r	**19.46**	12.40	17.58	16.07	18.10	17.44	12.84	11.84	16.06	15.21
imagenet1k	**18.51**	12.14	16.40	14.89	16.87	15.87	15.77	15.16	13.83	12.94
imagenet_sketch	**10.27**	6.06	8.39	7.33	8.87	8.29	2.69	2.44	7.14	6.59
imagenetv2	**15.14**	9.88	13.37	12.27	13.63	13.21	15.09	12.94	11.18	10.40
mnist	10.22	6.97	10.41	11.47	11.32	12.48	**14.60**	13.21	12.09	13.53
objectnet	**19.99**	11.96	14.64	12.92	14.41	12.75	12.86	12.36	11.11	10.22
renderedsst2	45.52	46.95	**51.07**	48.98	47.56	49.86	50.08	50.25	49.48	50.03
stl10	**75.17**	64.79	72.56	68.66	74.35	72.21	73.58	72.41	72.24	70.43
sun397	26.66	18.83	28.40	26.05	**28.46**	27.17	18.85	16.04	24.83	23.31
voc2007	**52.01**	37.42	50.49	47.27	49.49	47.98	47.55	45.78	47.27	46.12
caltech101	**63.16**	50.71	56.84	55.20	59.19	58.13	58.40	56.79	55.42	53.96
cifar10	**70.89**	57.04	68.68	64.82	68.26	66.50	67.48	65.48	68.78	67.87
cifar100	35.15	25.39	**35.62**	34.11	33.47	32.91	31.91	31.13	33.79	32.94
clevr_closest_object_distance	16.29	15.58	19.83	19.97	19.42	19.63	**20.31**	20.21	18.31	20.07
clevr_count_all	12.03	11.27	**12.85**	11.33	11.32	11.45	10.89	10.50	12.29	11.65
diabetic_retinopathy	**38.33**	33.21	3.09	2.99	4.28	5.54	4.35	3.76	4.05	8.14
dmlab	**13.42**	12.77	12.12	12.19	12.52	12.70	12.40	12.78	12.77	13.18
dsprites_label_orientation	2.81	**3.01**	1.98	2.15	2.08	2.15	1.76	2.29	2.17	2.13
dsprites_label_x_position	3.08	2.86	3.61	3.65	3.60	3.55	3.59	**3.88**	3.51	3.85
dsprites_label_y_position	1.88	2.60	**3.18**	3.14	3.15	3.12	3.17	3.10	3.11	3.09
dtd	**15.80**	12.55	15.37	15.27	15.64	15.43	14.68	14.20	12.23	11.70
eurosat	24.28	21.26	21.94	23.80	24.94	26.59	24.72	26.35	25.05	**26.61**
flowers	8.55	7.84	**9.22**	8.93	8.73	9.11	8.52	8.73	8.16	8.34
kitti_closest_vehicle_distance	**41.35**	38.82	30.80	34.18	30.52	33.05	32.77	33.05	33.76	33.61
pcam	50.10	50.30	**50.34**	50.21	50.30	50.25	50.25	50.23	50.21	50.13
pets	13.65	14.17	**15.94**	15.81	15.62	15.51	15.40	15.13	13.57	13.46
resisc45	**14.51**	10.87	11.54	11.43	12.29	10.94	11.04	11.89	10.73	10.19
smallnorb_label_azimuth	4.53	5.14	5.18	5.56	5.10	**5.67**	5.20	4.96	5.59	5.31
smallnorb_label_elevation	11.13	10.78	11.93	12.03	12.12	11.38	**12.50**	12.21	11.93	11.65
svhn	6.58	6.87	**9.91**	9.76	8.54	8.45	8.65	8.45	8.65	8.57
Absolute Average Accuracy	**22.20**	**18.52**	**20.39**	**19.76**	**20.31**	**20.07**	**19.61**	**19.14**	**19.41**	**19.37**

Table 4. Zero-shot classification accuracy across datasets. For each dataset, the best score (across all models and token counts) is shown in bold, and the second-best is underlined.

Distillation vs. No-Distillation Ablations. Figure 8 compares three initialization and supervision strategies: (1) RGB-pretrained ViT initialization, (2) vision-only distillation from a frozen RGB teacher followed by CLIP adaptation, (3) Simultaneous CLIP+vision-distillation. Our approach of **first** performing vision-distillation, **and then** CLIP-adaptation largely outperforms both other techniques, and makes the best use of both vision-vision and text-vision supervision.

Distillation Loss Variants. We evaluated multiple distillation objectives, including cosine similarity, InfoNCE, and cosine similarity augmented with a similarity-preserving constraint inspired by [4]. For the latter, we applied a layerwise similarity

Dataset	RGB ViT-B/16 (Small)		GS 3136		GS 1600		GS 900		GS 400	
	196	98	196	98	196	98	196	98	196	98
cars	1.00	0.55	1.05	0.91	**1.06**	0.95	0.94	0.86	0.70	0.61
country211	**1.00**	0.79	0.91	0.79	0.90	0.88	0.99	0.98	0.82	0.80
fer2013	**1.00**	1.00	0.41	0.50	0.20	0.26	0.28	0.23	0.42	0.52
fgvc_aircraft	1.00	1.15	**1.35**	1.26	0.80	0.85	0.78	1.31	0.80	1.20
gtsrb	1.00	1.08	1.30	**1.37**	1.30	1.15	1.28	1.16	1.16	1.31
imagenet-a	1.00	0.70	0.88	0.76	0.92	0.79	0.50	0.39	0.80	0.75
imagenet-o	1.00	0.76	0.92	0.89	**0.94**	0.89	0.89	0.88	0.79	0.82
imagenet-r	1.00	0.64	0.90	0.83	0.93	0.90	0.66	0.61	0.83	0.78
imagenet1k	1.00	0.66	0.89	0.80	0.91	0.86	0.85	0.82	0.75	0.70
imagenet_sketch	1.00	0.59	0.82	0.71	0.86	0.81	0.26	0.24	0.70	0.64
imagenetv2	**1.00**	0.65	0.88	0.81	0.90	0.87	1.00	0.85	0.74	0.69
mnist	1.00	0.68	1.02	1.12	1.11	1.22	**1.43**	1.29	1.18	1.32
objectnet	**1.00**	0.60	0.73	0.65	0.72	0.64	0.64	0.62	0.56	0.51
renderedsst2	1.00	1.03	**1.12**	1.08	1.04	1.10	1.10	1.10	1.09	1.10
stl10	1.00	0.86	0.97	0.91	0.99	0.96	0.98	0.96	0.96	**0.94**
sun397	1.00	0.71	1.07	0.98	**1.07**	1.02	0.71	0.60	0.93	0.87
voc2007	**1.00**	0.72	0.97	0.91	0.95	0.92	0.91	0.88	0.91	0.89
caltech101	**1.00**	0.80	0.90	0.87	0.94	0.92	0.92	0.90	0.88	0.85
cifar10	1.00	0.80	0.97	0.91	0.96	0.94	0.95	0.92	**0.97**	0.96
cifar100	1.00	0.72	1.01	0.97	0.95	0.94	0.91	0.89	0.96	0.94
clevr_closest_object_distance	1.00	0.96	1.22	1.23	1.19	1.21	**1.25**	1.24	1.12	1.23
clevr_count_all	1.00	0.94	**1.07**	0.94	0.94	0.95	0.91	0.87	1.02	0.97
diabetic_retinopathy	**1.00**	0.87	0.08	0.08	0.11	0.14	0.11	0.10	0.11	0.21
dmlab	**1.00**	0.95	0.90	0.91	0.93	0.95	0.92	0.95	0.95	0.98
dsprites_label_orientation	**1.00**	1.07	0.70	0.77	0.74	0.77	0.63	0.81	0.77	0.76
dsprites_label_x_position	1.00	0.93	1.17	1.19	1.17	1.15	1.17	**1.26**	1.14	1.25
dsprites_label_y_position	1.00	1.38	1.69	1.67	**1.68**	1.66	1.69	1.65	1.65	1.64
dtd	**1.00**	0.79	0.97	0.97	0.99	0.98	0.93	0.90	0.77	0.74
eurosat	1.00	0.88	0.90	0.98	1.03	1.10	1.02	1.09	1.03	**1.10**
flowers	1.00	0.92	**1.08**	1.04	1.02	1.07	1.00	1.02	0.95	0.98
kitti_closest_vehicle_distance	**1.00**	0.94	0.74	0.83	0.74	0.80	0.79	0.80	0.82	0.81
pcam	**1.00**	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
pets	1.00	1.04	**1.17**	1.16	1.14	1.14	1.13	1.11	0.99	0.99
resisc45	1.00	0.75	0.80	0.79	0.85	0.75	0.76	**0.82**	0.74	0.70
smallnorb_label_azimuth	1.00	1.13	1.14	1.23	1.13	**1.25**	1.15	1.09	1.23	1.17
smallnorb_label_elevation	1.00	0.97	1.07	1.08	1.09	1.02	**1.12**	1.10	1.07	1.05
svhn	1.00	1.04	**1.51**	1.48	1.30	1.28	1.31	1.28	1.31	1.30
Average Relative Accuracy	**1.00**	**0.87**	**0.98**	**0.96**	**0.96**	**0.95**	**0.92**	**0.91**	**0.91**	**0.92**

Table 5. Relative accuracy table with bold marking the best per row and underline marking the second-best per row.

loss with weight $\gamma = 2000$ to encourage structural alignment between the GS and RGB feature spaces. Figure 9 shows that although similarity-preserving and InfoNCE losses improve stability, simple cosine-matching achieves the highest CLIP-level zero-shot accuracy and offers the most reliable convergence behavior in practice.

Normalization Studies. We additionally assessed the effect of normalization choices on CLIP training dynamics (Fig. 10). We found log-scaling of covariance components as well as normalization of the x,y components by the resolution to work better than z-score normalization using the collected dataset statistics.

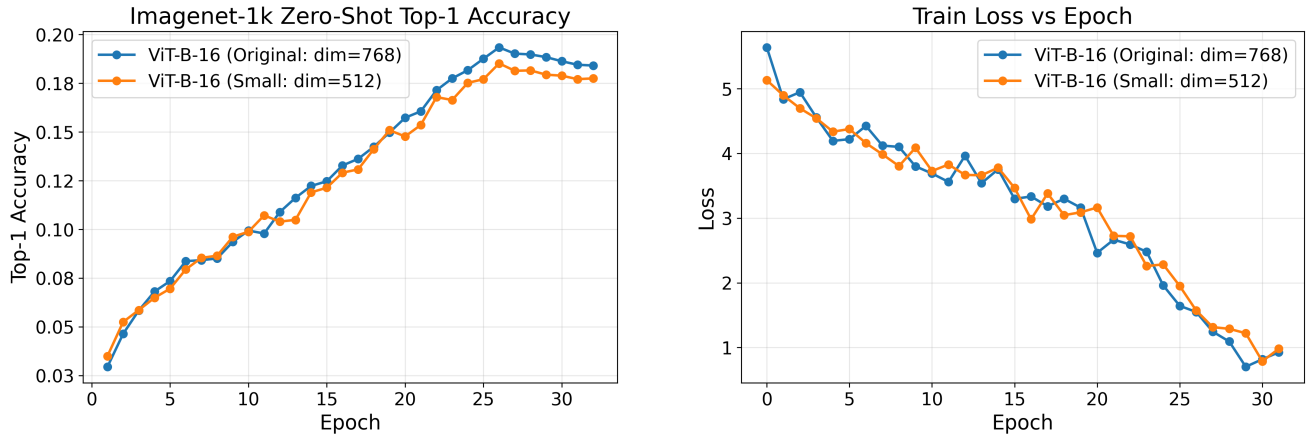


Figure 6. **Effect of ViT Width Reduction (768 → 512).** Zero-shot accuracy (left) and train loss (right) for the standard ViT-B/16 (width 768) and our reduced-cost ViT-B/16 (Small) variant (width 512). Both models exhibit nearly identical behavior, validating the reduced-width encoder as an appropriate baseline.

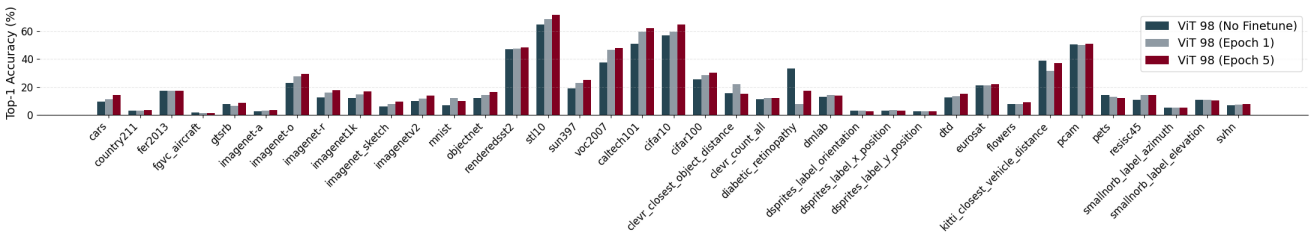


Figure 7. **Impact of Token-Count Reduction on RGB ViT Performance.** Comparison of ViT-B/16 models using 98 tokens under no fine-tuning, 1-epoch fine-tuning, and 5-epoch fine-tuning.

Hyperparameter Ablations. Figure 11 presents ablations over logit-scale initialization, text-encoder freezing policies, and learning-rate magnitude. Logit-scale transfer from the RGB baseline improves early performance, whereas fully freezing the text tower degrades convergence speed in later epochs.

GSStem Architectures. We explored several designs for projecting N Gaussian points into a compact set of M latent tokens. As shown in Fig. 12, a Perceiver-style cross-attention stem produces the best zero-shot performance and the smoothest training loss. Alternative stems, including (i) grid-based Gaussian pooling that mimics ViT patchification and (ii) Hilbert-ordered chunking with localized convolution over point sequences, proved less effective. The Perceiver design consistently extracted richer, more expressive latent tokens.

Point Transformer Study. Inspired by the 3D origin of Gaussian-splatting, we also evaluated point-based transformer encoders for 2DGS features. Figure 13 shows zero-shot accuracy after one epoch of training (left) and the corresponding training losses (right). We found point transformer based models to be less stable to train and suboptimal in terms of performance compared to perceiver architectures, albeit point transformers' ability to emit variable token counts aligns well with 2DGS adaptivity, and merits further study.

D.6. Case Studies on Representational Power and Bottlenecks of 2DGS.

To better understand the fundamental capabilities and limitations of 2D Gaussian Splat (2DGS) representations, we conducted a series of controlled case studies, summarized in Figs. 14–16. These analyses isolate the effects of lossy compression, lack of pretrained inductive bias, and architectural choices on downstream representational performance.

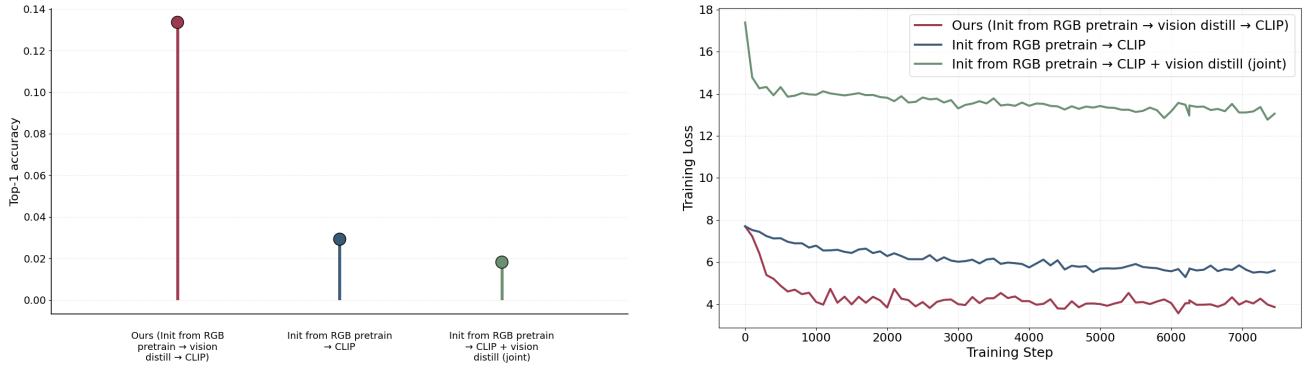
1. **Rendering Back Into Pixels.** We render our GS-1600 dataset back into RGB pixel space and retrain a ViT-B/16 (Small) encoder on the reconstructed images (Fig. 14). As expected for a lossy compressor, accuracy declines relative to training

Dataset	ViT-98 no-ft	ViT-98 ep1	ViT-98 ep5
cars	9.35	11.28	14.07
country211	3.01	3.10	3.32
fer2013	17.34	17.09	17.23
fgvc_aircraft	1.59	1.17	1.32
gtsrb	7.81	6.50	8.67
imagenet-a	2.79	2.99	3.61
imagenet-o	22.95	27.80	29.40
imagenet-r	12.40	15.94	17.84
imagenet1k	12.14	14.47	16.64
imagenet_sketch	6.06	7.86	9.49
imagenetv2	9.88	11.69	13.64
mnist	6.97	11.99	10.13
objectnet	11.96	14.04	16.35
renderedsst2	46.95	47.45	48.11
stl10	64.79	68.66	71.43
sun397	18.83	22.91	24.84
voc2007	37.42	46.59	47.98
caltech101	50.71	59.61	62.14
cifar10	57.04	59.52	64.83
cifar100	25.39	28.59	30.08
clevr_closest_object_distance	15.58	21.90	15.20
clevr_count_all	11.27	12.29	11.92
diabetic_retinopathy	33.21	7.59	17.40
dmlab	12.77	14.06	13.73
dsprites_label_orientation	3.01	2.96	2.56
dsprites_label_x_position	2.86	3.43	3.05
dsprites_label_y_position	2.60	2.43	2.55
dtd	12.55	13.24	14.89
eurosat	21.26	21.24	21.87
flowers	7.84	7.66	9.19
kitti_closest_vehicle_distance	38.82	31.65	37.27
pcam	50.30	50.17	50.76
pets	14.17	12.92	11.94
resisc45	10.87	14.10	14.19
smallnorb_label_azimuth	5.14	5.00	5.00
smallnorb_label_elevation	10.78	10.91	10.40
svhn	6.87	7.36	7.58
Average Absolute Accuracy	**18.52**	**19.41**	**20.56**
Avg Relative Accuracy to Base (ViT 196)	**0.87**	**0.91**	**0.95**

Table 6. Top-1 accuracy (%) of ViT-B/16 with 98 tokens under three settings: no fine-tuning, 1-epoch fine-tuning, and 5-epoch fine-tuning.

on original RGB, but the *relative drop* closely mirrors the performance drop observed when training CLIP using GS representations directly. This alignment suggests that our Perceiver-based GS encoder is extracting the majority of the semantic content available in the rendered images, and that the overall performance ceiling is fundamentally constrained by the quality of the compressed RGB surrogate. Because our GS encoder is distilled from an RGB-pretrained teacher, its achievable zero-shot performance is similarly bounded by the representational limits of the underlying pixel-domain ViT.

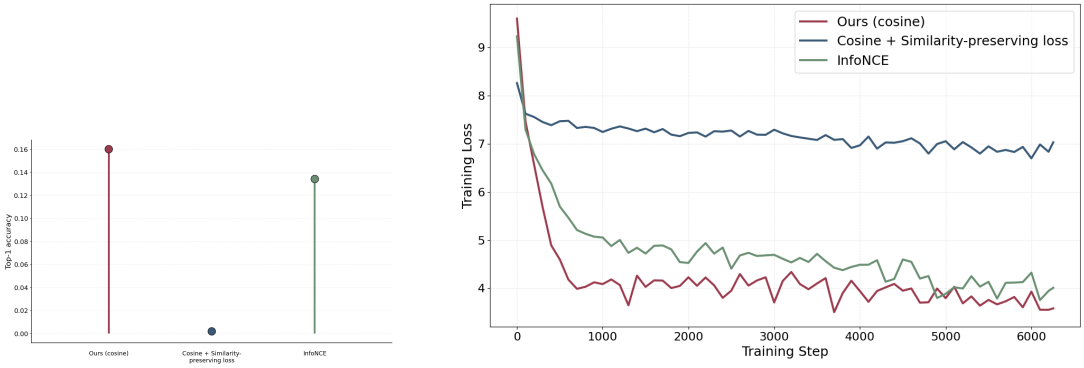
2. **Training GS Encoders Fully From Scratch.** We train our GS-1600 (196-token) encoder *entirely from scratch*, without RGB initialization, pretraining, or distillation (Fig. 15). This model exhibits substantially lower zero-shot accuracy and slower convergence, highlighting the absence of strong native inductive biases for 2DGS representations when trained



(a) Zero-shot Top-1 after 1 Epoch

(b) Training Loss

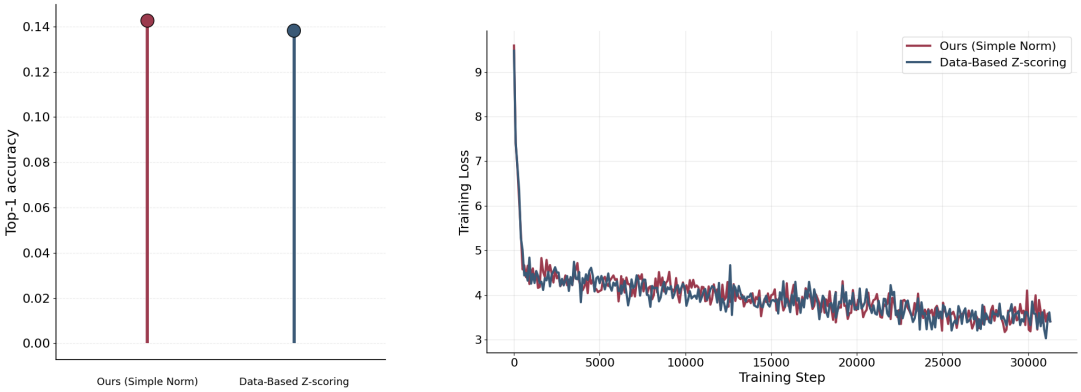
Figure 8. Comparison of distillation strategies and their effect on CLIP alignment and optimization dynamics.



(a) Zero-shot Top-1 after 1 Epoch

(b) Training Loss

Figure 9. Comparison of distillation losses and their effect on CLIP alignment and optimization dynamics.



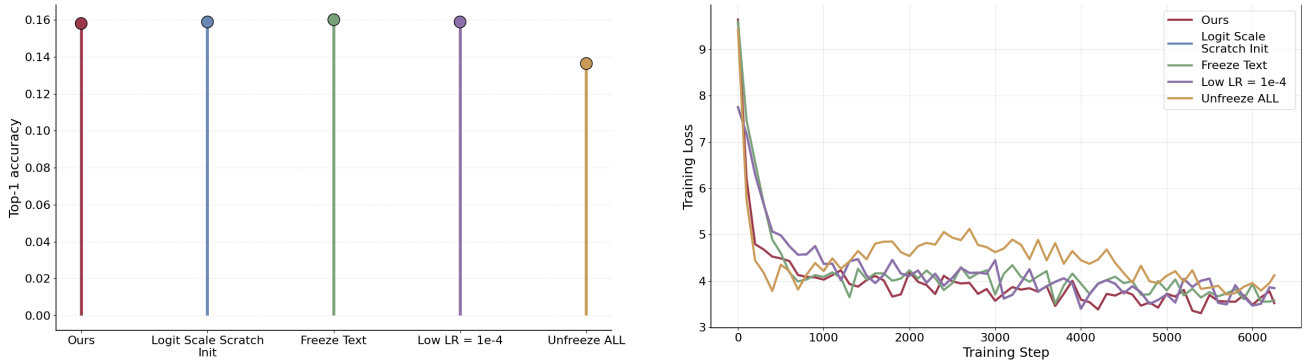
(a) Zero-shot Top-1

(b) Training Loss

Figure 10. Normalization study: effects of dataset-level normalization vs. simple rescaling.

without external guidance. In contrast, distillation from RGB features provides a powerful initialization signal that lifts the GS encoder out of the suboptimal optimization basin associated with scratch training, emphasizing the importance of cross-domain supervision for early-stage 2DGS models.

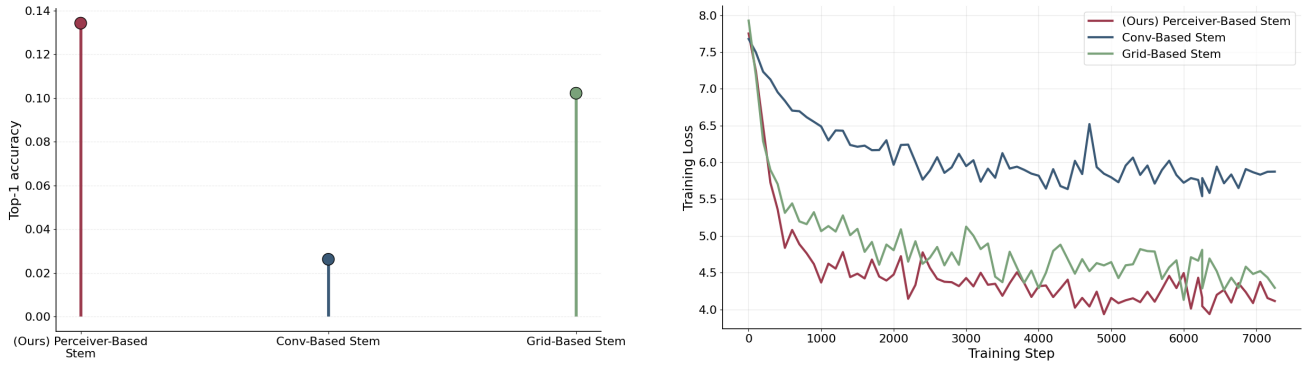
3. **2DGS as a Tokenizer for Vanilla Transformers.** To test the representational power of 2DGS in extreme compression regimes without specialized architectures, we feed a low number of Gaussian splats directly into a vanilla Transformer



(a) Zero-shot Top-1 after 1 Epoch

(b) Training Loss

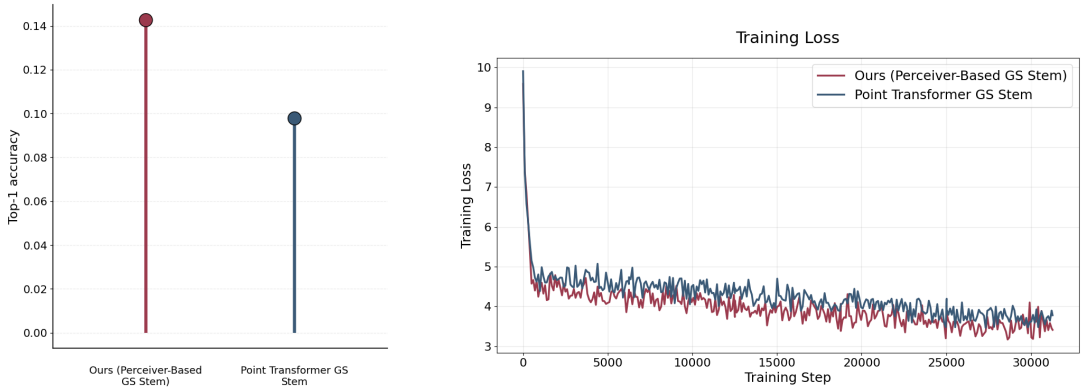
Figure 11. Hyperparameter sensitivity analysis across logit-scale initialization, learning rate, and text-freezing behaviors.



(a) Zero-shot Top-1 after 1 Epoch

(b) Training Loss

Figure 12. GS-Stem architecture study: different approaches to going from N points to smaller M latents.



(a) Zero-shot Top-1 after 1 Epoch

(b) Training Loss

Figure 13. Point Transformer ablation: early-stage CLIP alignment performance when varying initialization and distillation strategy.

after lightweight per-channel normalization, rescaling, and Fourier feature augmentation (Fig. 16). We pre-train the GS encoder from scratch in this case. We evaluate extreme compression settings (196 points) and low ones (400 points). Accuracy lags behind architectures that explicitly map from N splats to M latents (e.g., Perceiver stems). The results indicate that starting from a richer Gaussian budget yields better fidelity and that an architectural bottleneck is required to appropriately condense spatially distributed Gaussians into semantically meaningful tokens. This points toward the need

for future architectures that can *adaptively* produce the number of output tokens based on the density and structure of the input splats.

D.7. Alignment using Pruned GS Inputs

To close the loop on pruning, we run end-to-end CLIP alignment with pruned GS-1600 using a KV padding mask to support variable points. Table 7 shows minimal degradation. Notably, pruning GS-1600 down to ∼800 points yields better downstream performance than fitting from our GS-900 point budget directly, consistent with our PSNR observation in the main paper. An interesting future direction is to support adaptive numbers of output tokens, particularly for aggressively pruned GS inputs.

	GS-1600			GS-900
Tokens	**No prune**	**Prune 25%**	**Prune 50%**	**No prune**
196 tok	16.90	16.39	16.40	15.78
98 tok	15.87	15.34	15.67	15.14

Table 7. ImageNet-1K Val Accuracy for CLIP using pruned GS points ($Th = 0.05$, $\lambda{=}e{-}5(5e{-}6)$ for prune $\sim 50\%(\sim 25\%)$.

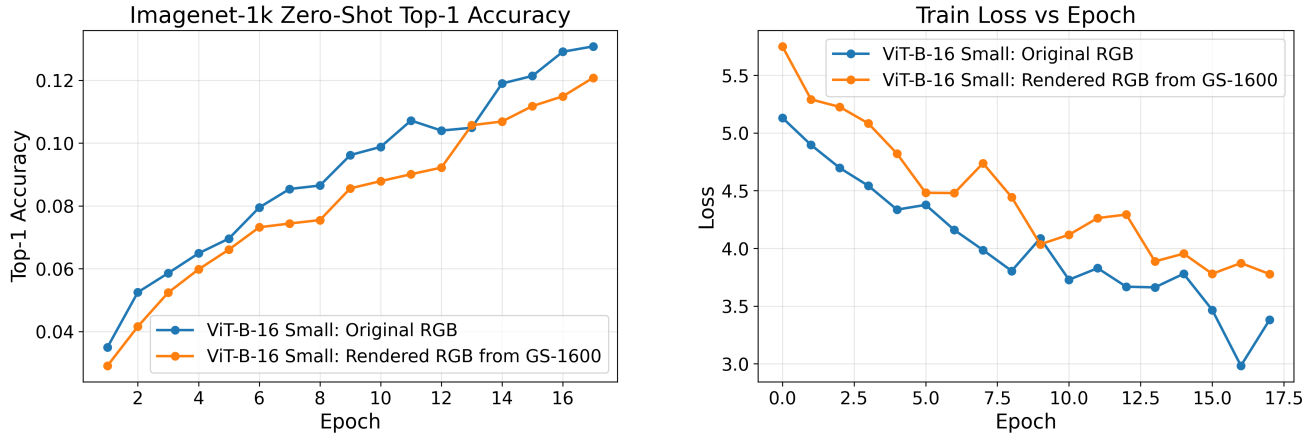


Figure 14. **Rendered ViT:** Zero-shot accuracy (left) and train loss (right).

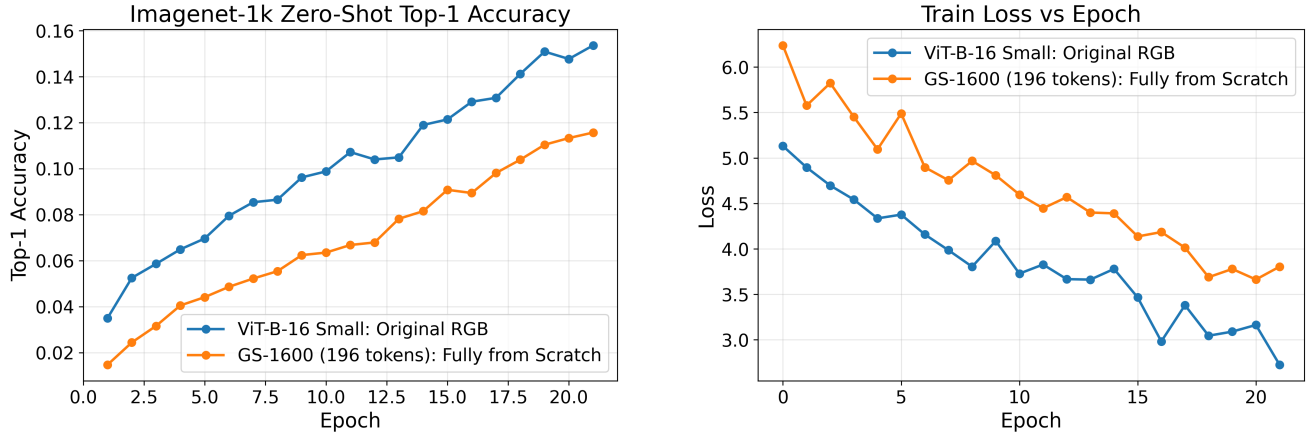


Figure 15. **Training From Scratch:** Zero-shot accuracy (left) and train loss (right).

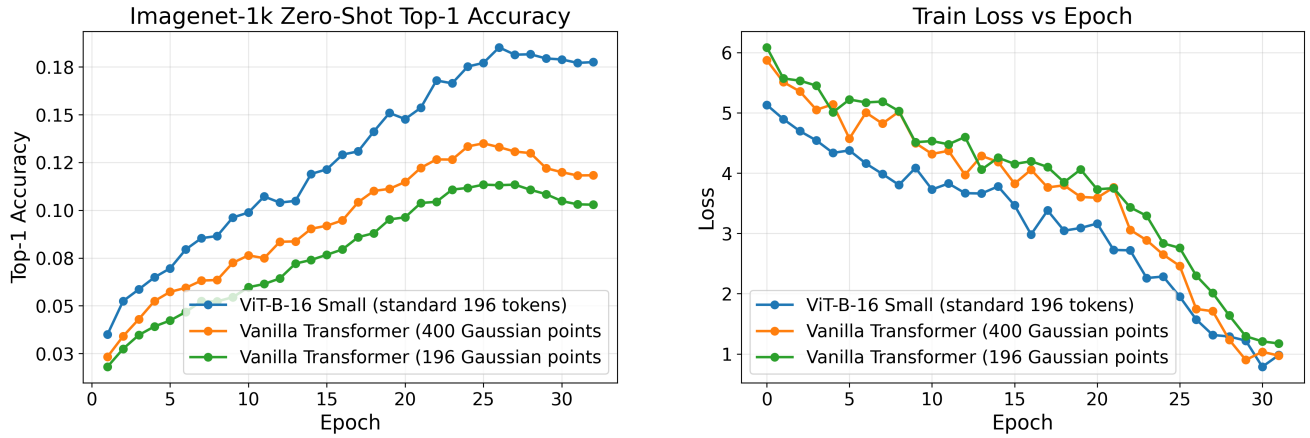


Figure 16. **Token Ablation Study:** Zero-shot accuracy (left) and train loss (right).

GS Fitting Loss for 1600 points (iters)	**Cos. Sim.**	**IN-1K**
MSE (1k)	93.17	**16.90**
LPIPS (300)	**96.24**	8.37
Hybrid (*400 MSE + 200 (0.9 LPIPS + 0.1 MSE)*)	94.57	16.76

Table 8. Effect of GS fitting loss on distillation and CLIP stages.

Codec	**KB/img**	**Codec**	**KB/img**
JPEG (Q=75)	9.69	GS-1600 (FP16)	25.0
JPEG (Q=95)	22.74	GS-1600 Prune 50% (FP16)	12.5
WebP (Q=95)	18.98	GaussianImage GS codec	11.4

Table 9. Average Kodak size at 224x224. Uncompressed=147KB.

D.8. On PSNR and Semantic Expressivity

We started with PSNR (i.e., pixel-wise MSE), as it is the standard objective in GS pipelines, but we believe that exploring more semantic losses is an important orthogonal research direction. To probe this question, we experimented with perceptual losses based on LPIPS (commonly used as a learned similarity metric in neural rendering and implicit representation works). We also evaluate a hybrid loss that combines MSE efficiency with LPIPS perceptual guidance. LPIPS is substantially more expensive than MSE ($\sim 5\times$ slower), so we reduce iterations to match runtime. Interestingly, LPIPS-derived GS points improve cosine similarity at the distillation stage but converge poorly at the CLIP alignment stage, leading to significantly degraded zero-shot accuracy, per Table 8.

D.9. Discussion on Standard Codecs

The compression ratios we report are relative to raw pixels (exclude quantization/entropy coding), and thus should be viewed as conservative lower bounds. We report #params/image to characterize encoder-side representational complexity, distinct from upstream transport formats. Prior work such as *GaussianImage* shows that, with quantization and entropy coding, 2DGS can match or outperform JPEG for transport/storage at low BPP, but this incurs decoding overhead. To ground this discussion, we include a bitrate comparison on the resized Kodak dataset in Table 9. Since DataComp images are already JPEG/WebP/PNG, our reported data-loading speedups already include compressed I/O and decoding overheads.

References

[1] Mehdi Cherti and Romain Beaumont. Clip benchmark, 2025. 2, 4

[2] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip. https://doi.org/10.5281/zenodo.5143773, 2021. Version 0.1, Zenodo. DOI: 10.5281/zenodo.5143773. 2

[3] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021. 2

[4] Yonglong Tian, Dilip Krishnan, and Phillip Isola. Contrastive representation distillation. *arXiv preprint arXiv:1907.09682*, 2019. 9

[5] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andr'e Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, Lucas Beyer, Olivier Bachem, Michael Tschannen, Marcin Michalski, Olivier Bousquet, Sylvain Gelly, and Neil Houlsby. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019. 2

[6] Xinjie Zhang, Xingtong Ge, Tongda Xu, Dailan He, Yan Wang, Hongwei Qin, Guo Lu, Jing Geng, and Jun Zhang. Gaussianimage: 1000 fps image representation and compression by 2d gaussian splatting. In *Proceedings of the European Conference on Computer Vision (ECCV)*, 2024. 5

[7] Lingting Zhu, Guying Lin, Jinnan Chen, Xinjie Zhang, Zhenchao Jin, Zhao Wang, and Lequan Yu. Large images are gaussians: High-quality large image representation with levels of 2d gaussian splatting. *in Annual AAAI Conference on Artificial Intelligence (AAAI)*, 2025. 5