ONEFIELD: A SHARED EMBEDDING SPACE FOR NERFS AND 3D GAUSSIAN SPLATTING

Anonymous authors
Paper under double-blind review

ABSTRACT

Radiance fields are a data modality where an object is stored as a function of space. Recent encoders read that function directly to perform classification, retrieval, segmentation, and language-based tasks. A radiance field is commonly stored in one of two formats: Neural Radiance Fields (NeRFs) or 3D Gaussian Splatting (3DGS). These formats exhibit a different trade-off between memory footprint and rendering speed: NeRFs are memory-efficient but slow to render, whereas 3DGS enable fast rendering at a substantially higher memory cost. Existing encoders can process only one format, i.e., a model trained on NeRF weights does not read Gaussian primitives, and vice versa. Yet, the two formats can be trained on the same rendered views of an object. Thus, they differ in the representation used to model the object, not in the object they encode. ***OneField*** is an encoder that maps both formats into a common embedding space, aligned across formats by a contrastive objective, and with a shared decoder enforcing the object's field reconstruction. In this space, the format of the input no longer matters: in classification, part segmentation, and language tasks, a head trained on one format transfers to the other one, with little to no performance degradation. ***OneField*** remains competitive with format-specific encoders in all tasks. In addition, it enables retrieval from a unified gallery containing both formats, a task that no format-specific encoder can do, and retrieves the correct object at 88–90% recall@1, 14–18 points above a CLIP baseline processing rendered views.

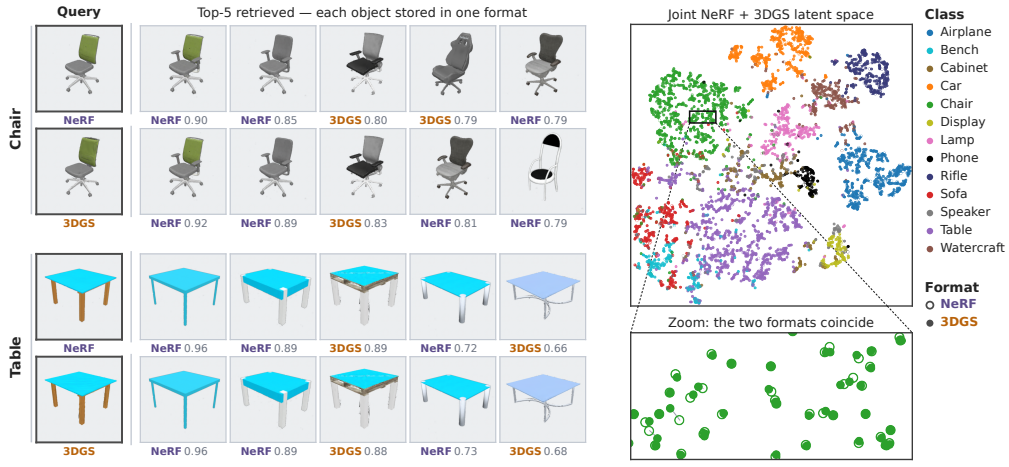


Figure 1: ***OneField*** **organizes its embedding space by object and class, not by format.** *Left:* top-5 retrievals for two query objects, each queried from both its NeRF and its 3DGS. The gallery is mixed, with each object represented as either a NeRF or a 3DGS. Changing the query format leaves the retrieval results substantially unchanged. The retrieved neighbors span both formats, and their ranking depends on the similarity to the query object in terms of geometry and appearance rather than on the representation format. *Right:* t-SNE of ***OneField*** embeddings for the ShapeNet test split. Classes form distinct clusters, where the two representation formats are interleaved. The zoom shows how the two representations of the same object map to virtually the same point.

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.

1 INTRODUCTION

Radiance fields have become a popular data modality: they are collected into large datasets and directly processed by deep learning models (Amaduzzi et al., 2024; 2025; Ma et al., 2025; Li et al., 2025b). A radiance field describes an object as a function assigning to each 3D point in space a density and a color. Rendering an image means integrating this function along the rays of a camera, so one field replaces the images used to fit it.

A radiance field is usually stored in two different formats. A Neural Radiance Field (NeRF) (Mildenhall et al., 2021) stores the function in the weights of a neural network. This field is continuous, can be queried at any point in space, and fits in a few megabytes of weights. Its compact representation, however, comes at the cost of slow rendering: evaluating the network along the rays of a camera is computationally expensive. 3D Gaussian Splatting (3DGS) (Kerbl et al., 2023), instead, stores the field as a set of anisotropic Gaussian primitives, each with a position, a covariance, an opacity, and spherical harmonic coefficients. The explicit spatial representation enables fast rendering, but requires storing a potentially large number of primitives, resulting in a substantially larger memory footprint. Thus, the two formats establish a different trade-off between memory efficiency and rendering speed. In addition, encoding these formats presents distinct challenges. In NeRFs, the geometric structure of the object is entangled within the weights of the network, therefore lacking spatial grounding; in 3DGS, instead, every primitive is grounded in a specific point in space. What 3DGS lacks is continuity and structure: the primitives form an unordered, discrete set with no encoded relation between neighbors (Fang et al., 2025).

These formats can be processed in several ways. One option is to render images and feed them to vision models. This method involves the choice of the rendering viewpoints and computationally expensive rendering pipelines. On the other hand, a recent line of work proposes to process the field directly, training encoders that ingest NeRF weights (Ramirez et al., 2024) or sets of Gaussians (Ma et al., 2025; Li et al., 2025b), to produce latents that support downstream tasks. Each of these encoders is designed with one single format in mind. NeRF encoders consume weight matrices, while 3DGS encoders process point sets. This discrepancy has two consequences. The first is duplication. No pretrained encoder transfers between the two formats. Therefore, datasets gathered for NeRFs cannot supervise 3DGS models, and vice versa. The second is that the representation determines the entire processing pipeline: an object represented as a NeRF and the same object represented as 3DGS must be encoded by different architectures, despite both describing the same underlying field.

As a NeRF and a 3DGS are two parameterizations of the same field, we believe that an encoder should process each format, regardless of its parameterization.

We build ***OneField*** on this principle. Each format is processed by its own encoder branch, and both branches contribute to a single embedding shaped by two objectives: a contrastive loss aligns the two formats, and a shared radiance decoder reconstructs the original field from the embedding. ***OneField*** is trained on our dataset of paired NeRFs and 3DGS, optimized from the same rendered views of 300k objects from ShapeNet (Chang et al., 2015) and Objaverse (Deitke et al., 2023). In the resulting space, the input format is no longer a constraint (Figure 1, Right): task heads trained on NeRFs or 3DGS alone perform classification, part segmentation, captioning, and QA on both formats. Moreover, ***OneField*** unlocks cross-format retrieval from mixed galleries containing objects represented either as NeRF or 3DGS (Figure 1, Left).

We summarize our **contributions** as follows:

- ***OneField***, a single architecture that maps NeRFs and 3DGS into one shared embedding space.
- Evidence that this space is format-agnostic: a classifier, a segmentation model, and a language assistant trained on one format transfer to the other, with little to no performance degradation.
- Cross-format retrieval on mixed galleries, where ***OneField*** reaches 88–90% recall@1, 14–18 points above a rendering-based contrastive baseline, without its rendering cost and viewpoint selection.
- The first paired dataset of NeRFs and 3DGS, with both representations fitted to the same rendered views of 300k objects from ShapeNet and Objaverse.

2 Related Work

Encoders for Neural Radiance Fields. A NeRF (Mildenhall et al., 2021) represents an object as a radiance field parameterized by the weights of a neural network. nf2vec (Ramirez et al., 2024) is the first encoder to directly process the weights of this MLP: it stacks the weights and biases into a matrix, processes each row with a shared MLP, and max-pools the rows into a global embedding that supports downstream tasks. Later work connects this embedding to language. LLaNA (Amaduzzi et al., 2024) projects the output of nf2vec into the embedding space of a large language model, which answers questions and generates descriptions directly from NeRF weights. Spatial LLaNA (Amaduzzi et al., 2026) encodes the same tokens into a sequence arranged on a triplane, to provide localized features for reasoning about object details and spatial relations. NeRFs can be stored in different network architectures, such as MLPs, tri-planes, and hash tables. Ballerini et al. (2026) address this with a graph meta-network that encodes NeRFs of thirteen architectures. However, all thirteen are collections of weights. A NeRF and a set of Gaussians share no such structure, so a unified encoder for both remains an open question.

Encoders for 3D Gaussian Splatting. 3DGS (Kerbl et al., 2023) stores the radiance field as an unordered set of primitives, which connects this format to point-cloud processing (Qi et al., 2017a;b; Zhao et al., 2021; Yu et al., 2022). A primitive, however, carries spatial extent and appearance in addition to its position, which motivates encoders that read the full set of attributes. ShapeSplat (Ma et al., 2025) pretrains a masked autoencoder over the primitives to perform object recognition. UniGS (Li et al., 2025b) instead aligns the 3DGS embedding with the CLIP embeddings of the object's renders and captions. SceneSplat (Li et al., 2025c) and Chorus (Li et al., 2026) extend language supervision to scenes, predicting a feature per Gaussian rather than one descriptor per object. These encoders are the mirror image of the NeRF encoders above: a 3DGS encoder cannot process the weights of a NeRF, and a NeRF encoder cannot process a set of Gaussians.

Combining NeRFs and 3D Gaussian Splats. Reconstruction methods already exploit the two formats together. Fang et al. (2025) optimize a NeRF alongside a set of Gaussians for the same scene, and VDGS (Malarz et al., 2025) drives Gaussian appearance with a neural field. In both cases, each format supplies what the other lacks. These methods support our premise that the two formats describe the same field in complementary ways. These methods produce a better radiance field, optimized per scene. ***OneField*** moves in the opposite direction: instead of combining the two formats into one field, it learns a joint embedding from paired NeRF–3DGS data during training, and at inference either format produces that embedding on its own.

Shared embedding spaces. Mapping heterogeneous inputs into one space with a per-input encoder and a contrastive objective is an established recipe. CLIP (Radford et al., 2021) aligns images and text to turn the text encoder into a zero-shot classifier for images; ULIP (Xue et al., 2023; 2024) and OpenShape (Liu et al., 2023) add a 3D encoder to a frozen image-text space; ImageBind (Girdhar et al., 2023) trains every encoder against images alone and finds that encoders never trained together are aligned at test time. These works showed that a shared space transfers across inputs. The inputs they align, however, carry different information: an image and its caption correspond without being equivalent, so a residual separation between their encoders is expected. This modality gap is measurable and persistent, leaving a systematic offset between contrastively aligned encoders (Liang et al., 2022). This gap degrades retrieval as soon as a single gallery mixes items of both kinds (Li et al., 2025a). The same effect appears when both formats are rendered and embedded with CLIP (Table 2). A NeRF and a set of Gaussians fitted to the same views, instead, are two fits to the same evidence, so a head trained on one branch should apply to the other without correction.

3 Method

In ***OneField***, each input format is processed by an encoder that produces a set of tokens. An Attentive Pooler aggregates each set into a single embedding $\boldsymbol{e} \in \mathbb{R}^D$. A decoder, shared by both branches, reconstructs the density and color field of the object from each of the embeddings. After training, the decoder is discarded, leaving two encoders that map both formats into a common space.

This embedding space has to satisfy two requirements. It has to be *shared*: the two encodings of an object must land on the same point, so that a model trained on one format can generalize to the other. It also has to be *expressive*: the embedding must retain the full nature of the input object, to

unlock fine-grained downstream tasks. An alignment objective on the two embeddings makes the space shared, and the reconstruction through the shared decoder makes it expressive.

3.1 Radiance Field

A radiance field represents an object o as a function of a 3D coordinate $\mathbf{x}$ of a point in space:

$$f_o : \mathbb{R}^3 \to \mathbb{R}^3 \times \mathbb{R}_{\geq 0}, \qquad \mathbf{x} \mapsto \big(\mathbf{c}(\mathbf{x}),\ \sigma(\mathbf{x})\big), \tag{1}$$

where $\sigma(\mathbf{x})$ is the density at $\mathbf{x}$ and $\mathbf{c}(\mathbf{x})$ the color emitted there. Volumetric rendering integrates f_o along the rays of a camera, so a set of posed images of o determines f_o at every point of the space. The NeRFs we encode share the architecture of nf2vec (Ramirez et al., 2024): an MLP with three hidden layers of 64 units. Every 3DGS of our dataset keeps the zeroth-order spherical harmonic of each primitive, so Equation (1) describes both formats exactly.

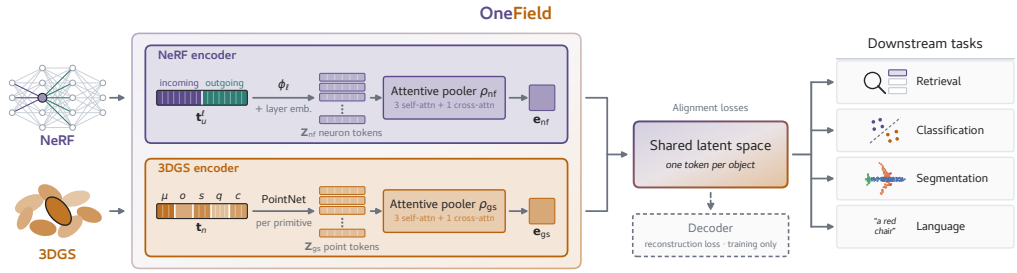


Figure 2: ***OneField*:** Format-specific encoders map the NeRF and the 3DGS of an object to a single token in a shared latent space, where alignment losses pull the two formats together. The decoder is trained with a reconstruction loss and discarded during inference. The embeddings can then be used for downstream tasks in a format-agnostic scenario.

3.2 NeRF Encoder: one token per neuron

As Figure 2 (top left) shows, the NeRF encoder of ***OneField*** takes as input the weights of a NeRF MLP. As in Ramirez et al. (2024), we concatenate the $L = 4$ bias-free weight matrices $\mathbf{W}_l \in \mathbb{R}^{n_l \times n_{l-1}}$ of the NeRF MLP, where n_i is the number of neurons in the i-th layer. Ramirez et al. (2024) reshape this vector into rows of a fixed width and embed each row independently. However, in this way, one row will mix weights that belong to different neurons. To avoid this artifact, we tokenize the weights by neuron. The token of neuron u in layer l will be associated with the row of its input weights and the column of its output weights:

$$\mathbf{t}_u^{(l)} \ = \ \big[\, \mathbf{W}_l[u,:] \ ; \ \mathbf{W}_{l+1}[:,u] \,\big] \ \in \ \mathbb{R}^{\,n_{l-1}+n_{l+1}}, \tag{2}$$

with the second block empty for the output layer. This tokenization has two advantages. First, as each token corresponds to a neuron, the linear layers of the NeRF encoder will compute over a genuine structure of the MLP, rather than over slices of weights, which are artifacts of tokenization. Second, the number of output tokens $T = \sum_l n_l = 208$ is fixed by the architecture of the NeRF rather than by a chosen row width.

Once stacked, each layer of the NeRF MLP will form a sequence of tokens. Each sequence is processed by a 3-layer MLP $\phi_\ell : \mathbb{R}^{n_{l-1}+n_{l+1}} \to \mathbb{R}^{512} \to \mathbb{R}^D$. In addition, differently from Ramirez et al. (2024), we sum a learnable embedding, $\mathbf{p}_l \in \mathbb{R}^D$, which encodes the layer a token comes from:

$$\mathbf{z}_u^{(l)} \ = \ \phi_\ell\big(\mathbf{t}_u^{(l)}\big) + \mathbf{p}_l. \tag{3}$$

Layer normalization over all units produces the full token set of the NeRF branch, $\mathbf{Z}_{\text{nf}} \in \mathbb{R}^{T \times D}$.

Finally, the sequence of tokens is aggregated into a single embedding $\boldsymbol{e}_{\text{nf}}$ by an Attentive Pooler (Assran et al., 2025): three self-attention blocks followed by a cross-attention block, where a single learnable query acts as $\boldsymbol{Q}$ and the neuron tokens as $\boldsymbol{K}$ and $\boldsymbol{V}$.

$$\mathbf{e}_{\text{nf}} \ = \ \rho_{\text{nf}}\big(\mathbf{Z}_{\text{nf}}\big) \ \in \ \mathbb{R}^D, \qquad D = 1024 \tag{4}$$

The information entangled within the weights of the NeRF is embedded into a single vector of dimension D.

3.3 3DGS ENCODER: ONE TOKEN PER GAUSSIAN PRIMITIVE

A 3D Gaussian Splat is an explicit, unordered set of primitives, and each primitive is a self-contained description of one region of the object. We therefore treat the primitive as the unit of tokenization.

A Gaussian g_n is stored as a position $\boldsymbol{\mu}_n \in \mathbb{R}^3$, an opacity logit $o_n \in \mathbb{R}$, a log-scale $\mathbf{s}_n \in \mathbb{R}^3$, an unnormalized rotation quaternion $\mathbf{q}_n \in \mathbb{R}^4$, and 48 spherical-harmonic coefficients $\mathbf{F}_n \in \mathbb{R}^{3\times 16}$ for color. We pre-process these primitives as follows. First, we apply the activations used at rendering time, so that the token carries the opacity and the scale the renderer would actually use. Second, we keep only the zeroth-order harmonic, $\mathbf{c}_n = C_0\,\mathbf{F}_n^{\mathrm{dc}} + \frac{1}{2}$ with $C_0 = 1/(2\sqrt{\pi})$, and drop the remaining 45 coefficients: they carry view-dependent appearance, which the field of Equation (1) does not model. Third, we canonicalize the quaternion to a positive real part, since $\mathbf{q}$ and $-\mathbf{q}$ denote the same rotation. Thus, a primitive becomes the token:

$$\mathbf{g}_n = \Big[\, \boldsymbol{\mu}_n \;;\; \mathrm{sigmoid}(o_n) \;;\; \exp(\mathbf{s}_n) \;;\; \mathbf{q}_n/\|\mathbf{q}_n\| \;;\; \mathbf{c}_n \,\Big] \in \mathbb{R}^{14}. \tag{5}$$

As shown in Figure 2 (bottom left), the tokens are projected to $\mathbb{R}^D$ by a PointNet-style encoder (Qi et al., 2017a), a five-layer MLP with batch normalization and ReLU. As ϕ_ℓ on the NeRF side, it is applied to one primitive at a time with weights shared across primitives, and produces a set of tokens $\boldsymbol{Z}_{\mathrm{gs}} \in \mathbb{R}^{N\times D}$, where N is the number of primitives. Appendix A.2 details how we handle the variable number of primitives in 3DGS.

A second Attentive Pooler ρ_{gs}, identical in architecture to ρ_{nf} but with its own weights, collapses the set into one vector, $\mathbf{e}_{\mathrm{gs}} = \rho_{\mathrm{gs}}(\boldsymbol{Z}_{\mathrm{gs}}) \in \mathbb{R}^D$.

3.4 DECODER

As shown in Figure 2 (right), both encoders feed a single implicit decoder, adopted from nf2vec (Ramirez et al., 2024). The decoder receives one embedding at a time, either $\boldsymbol{e}_{\mathrm{nf}}$ from the NeRF encoder or $\boldsymbol{e}_{\mathrm{gs}}$ from the 3DGS encoder, concatenated with a frequency encoding of a query point $\boldsymbol{x} \in \mathbb{R}^3$, and returns the color and volume density of the field at that point, $\mathrm{dec}(\boldsymbol{e}, \boldsymbol{x}) = (\boldsymbol{c}, \sigma)$. The decoder is an MLP composed of linear layers and ReLUs. Because the same decoder must reconstruct the field from either embedding, with no indication of which encoder produced it, both encoders are pushed to encode the field in the same way, regardless of the input format.

3.5 TRAINING AND LOSSES

OneField is trained end-to-end with three complementary losses that constrain the shared space. The embedding losses encourage the two formats of each object to align, while the rendering loss ensures that each embedding retains enough information to reconstruct the object through the decoder.

The embedding losses require every training object to be available in both formats, so that its NeRF and its 3DGS form a positive pair. Matched pairs are pulled together by the SigLIP loss $\mathcal{L}_{\mathrm{siglip}}$ (Zhai et al., 2023). The similarity of the two ℓ_2-normalized embeddings, rescaled by a learned temperature and bias, is pushed up when they come from the same object and down otherwise.

Alignment alone does not prevent the two formats from settling into separate regions of the space, nor does it prevent the embeddings from occupying a low-dimensional subspace within it. We therefore add a cross-format VICReg term (Bardes et al., 2022), $\mathcal{L}_{\mathrm{vic}}$, that combines an invariance term that pulls matched pairs together, a variance term that prevents dimensions from collapsing, and a covariance term that decorrelates dimensions.

The rendering supervision follows nf2vec (Ramirez et al., 2024). For each object, we sample pixels from the views used for fitting, render them through the decoder conditioned on each of the object's embeddings, and compare the resulting colors against those rendered by the object's NeRF using a smooth L_1 loss. We apply the loss to both embeddings against the same target and average the two terms.

The training objective is a weighted sum of the loss terms described above. Encoders, attentive poolers, decoder, and SigLIP parameters are optimized end-to-end. Additional implementation details and the full loss formulations are provided in Appendix A.3 and Appendix A.4, respectively.

4 EXPERIMENTS

4.1 DATASET

We train and validate ***OneField*** on our proposed dataset, which contains paired NeRFs and 3DGS of objects from ShapeNet (Chang et al., 2015) and Objaverse (Deitke et al., 2023). The NeRFs belong to the dataset released by Amaduzzi et al. (2025), whereas we fit the 3DGS from the same views, so that the two formats of an object carry the same evidence and differ only in the parameterization. Appendix A.1 describes in detail the fitting of the 3DGS. The dataset contains $294{,}084$ objects for training, $4{,}675$ for validation and $3{,}958$ for testing.

4.2 BASELINES AND TASKS

To the best of our knowledge, no encoder supports both NeRFs and 3DGS, so we compare ***OneField*** against format-specific architectures: nf2vec (Ramirez et al., 2024) for NeRFs, ShapeSplat-GMAE (Ma et al., 2025) and UniGS (Li et al., 2025b) for 3DGS. UniGS additionally aligns 3DGS embeddings to CLIP, and thus uses 3DGS, images, and textual descriptions, whereas ***OneField***, ShapeSplat-GMAE, and nf2vec use only the radiance field parameters as input. We selected these encoders for their focus on object-centric representations and tasks. For a fair comparison, we train all baselines from scratch on our dataset using their released code, architecture, and training recipe. To provide the additional supervision required by UniGS, we use images and textual descriptions from ShapeNeRF-Text (Amaduzzi et al., 2024) and ObjaNeRF-Text (Amaduzzi et al., 2025). To compare the different models, we address classification, retrieval, and part segmentation, as well as language tasks such as captioning and QA. Additional details on the MLP heads used for the downstream tasks are provided in Appendix A.5.

4.3 CLASSIFICATION

Table 1 reports classification accuracy on the 13-class ShapeNet test split, containing $3{,}958$ objects. To perform the task, we freeze each encoder and train an MLP head on embeddings from one format. ***OneField*** is tested on both formats, thereby evaluating whether object categories remain legible across formats in the shared space. Format-specific encoders can only be evaluated on the format they were trained for. Thus, the empty cells are not omitted runs: nf2vec takes NeRFs, while ShapeSplat-GMAE and UniGS take only 3DGS, so neither admits off-diagonal evaluation. ***OneField*** is the only encoder for which all four train/test pairs are defined.

Table 1: **A classifier trained on one format keeps its accuracy on the other.** Accuracy in % over the objects of ShapeNet test split; bold marks the best result per column.

Method	Head trained on	Accuracy	
		NeRF	3DGS
nf2vec	NeRF	91.64	—
ShapeSplat-GMAE	3DGS	—	91.03
UniGS	3DGS	—	**95.38**
OneField	NeRF	**93.56**	93.48
OneField	3DGS	**93.56**	93.66

A head trained on ***OneField***'s NeRF embeddings loses only 0.08 accuracy points when tested on 3DGS, while one trained on 3DGS loses 0.10 points when tested on NeRFs, showing that the two formats are effectively interchangeable for downstream classification. On their native formats, ***OneField*** outperforms both single-format encoders: 93.56 vs. 91.64 for nf2vec on NeRFs, and 93.66 vs. 91.03 for ShapeSplat-GMAE on 3DGS. UniGS, which, unlike ***OneField*** and ShapeSplat-GMAE, draws additional supervision from textual captions and rendered images, achieves the highest accuracy on 3DGS (95.38). Overall, the experiments demonstrate that ***OneField*** provides a shared, format-agnostic embedding space that enables effective classification across both representation formats.

4.4 RETRIEVAL

The retrieval experiments reported in this subsection quantitatively assess whether the embedding space is format-biased or instead captures global geometry and appearance across representations.

Table 2: ***OneField* retrieves an object across formats, and mixing the two formats in one gallery costs it 3 points instead of 15.** Given an object encoded from one format as query (rows), we retrieve the same object stored in the other format, from a gallery holding only the other format (*left*) or both formats (*right*). Recall@K in %, on the 3,958 objects of the ShapeNet test split.

	Other-format gallery						Mixed gallery					
	NeRF query			3DGS query			NeRF query			3DGS query		
Method	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10	R@1	R@5	R@10
CLIP	86.41	93.99	95.76	88.83	96.21	97.55	71.55	85.88	90.47	74.08	88.50	92.55
OneField	**92.29**	**99.19**	**99.72**	**91.61**	**98.76**	**99.44**	**89.69**	**98.08**	**99.19**	**87.87**	**97.60**	**98.76**

Given a NeRF or 3DGS query, we retrieve its counterpart in the other format by cosine similarity. A query is a hit at rank K if its counterpart is among the K nearest items. We report recall@K over the 3,958 objects of the ShapeNet test split in Table 2. Specifically, for each query object, represented as a NeRF or 3DGS, we consider galleries that include the same object represented in the other format (i.e., 3DGS or NeRF), along with all other objects represented either in the other format or in both (Table 2, Other-format and Mixed galleries, respectively). The mixed gallery is the hardest setting: each query is compared with 7,915 items and the correct match must outrank the 3,957 same-format objects, exposing whether the embedding is organized by object identity or by format.

Since format-specific encoders cannot perform cross-format retrieval, we use CLIP as a baseline. Each object is rendered from 36 views, encoded with pre-trained ViT-L/14 (Radford et al., 2021), and mean-pooled across views.

In the Other-format gallery, ***OneField*** reaches 92.29 and 91.61 R@1 for NeRF and 3DGS queries, respectively, outperforming the baseline by 5.88 and 2.78 points; R@10 exceeds 99.40 in both directions. Mixing the formats leaves these numbers almost unchanged: in the mixed gallery, R@1 drops by only 2.60 and 3.74 points, whereas the baseline drops by 14.86 and 14.75 points. The CLIP baseline is partly organized by format-specific rendering signatures, causing same-format distractors to outrank the true match. ***OneField*** learns to suppress these signatures, which is particularly evident in the mixed gallery.

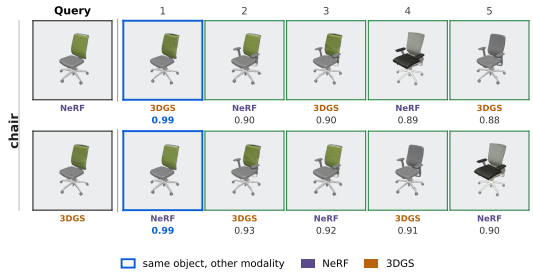


Figure 3: **Retrieval against a mixed gallery.** The chair is queried from its NeRF and 3DGS embeddings against a gallery holding both formats, tiles ordered by cosine similarity. The top match is always the same object in the other format, clearly separated from the rest.

Figure 3 shows an exemplar ranking: the nearest neighbor is the query's counterpart, followed by objects similar in shape and appearance, with NeRFs and 3DGS alternating throughout the ranking. Additional examples are reported in Appendix A.8.

4.5 Segmentation

Unlike classification and retrieval, part segmentation tests whether the embedding preserves local part identity and spatial structure. Given an object embedding and a point coordinate, a two-hidden-layer, category-specific head predicts the part assigned to that point. As in Section 4.3, each head is trained on one format and evaluated on both, where possible.

Table 3 reports per-category mIoU on the five ShapeNetPart categories. ***OneField*** is most accurate in every column except "Car" on 3DGS, where ShapeSplat-GMAE reaches 66.01 versus 62.62. UniGS has the lowest overall score (67.70).

Cross-format transfer is also strong: the NeRF- and 3DGS-trained heads differ by at most 0.92 mIoU on any category and 0.60 on the category mean, across both evaluation formats. This is a more challenging test than classification because the global embedding must retain information on individual objects' parts. In ***OneField***, the agreement is point-

Table 3: **A part segmentation head trained on one format transfers to the other within 0.92 mIoU.** Given the global embedding of an object and the coordinates of a point, a per-category head predicts the part the point belongs to. We report mIoU in % on the ShapeNetPart test split; bold marks the best result per column.

Method	Head trained on	Airplane		Car		Chair		Lamp		Table		Mean	
		NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS
nf2vec	NeRF	71.38	—	60.88	—	82.62	—	69.04	—	75.25	—	71.83	—
ShapeSplat-GMAE	3DGS	—	74.02	—	**66.01**	—	81.51	—	71.02	—	76.13	—	73.74
UniGS	3DGS	—	75.87	—	23.95	—	84.98	—	74.42	—	79.29	—	67.70
OneField	NeRF	**76.95**	77.11	**62.00**	62.07	**86.82**	87.11	74.45	74.50	**79.16**	**79.30**	**75.88**	76.02
OneField	3DGS	76.39	**77.31**	61.85	62.62	86.64	**87.41**	**74.72**	**74.90**	78.74	79.04	75.67	**76.26**

wise for both heads, i.e., the head for NeRFs assigns identical parts to 97.7% of points using 3DGS embeddings, and similar performance is achieved by the 3DGS head (97.4%). Read together with Section 4.3 and Section 4.4, these results show that ***OneField*** supports both global object recognition and local part prediction: unlike category, which is constant over an object, part identity varies across points and therefore requires spatial structure. ***OneField*** preserves both across formats, allowing a head trained on one to be applied to the other. Figure 4 shows the predictions side by side. Additional examples are in Appendix A.7.

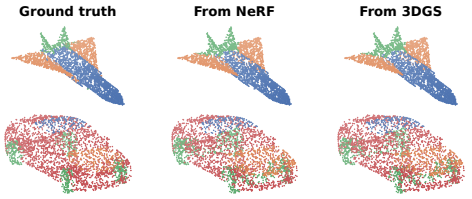


Figure 4: **The same head segments an object almost identically from its NeRF or 3DGS.**

4.6 Captioning and QA

Our final evaluation assesses whether a single embedding can condition a language model for open-ended description and question answering. We follow the pipeline of Amaduzzi et al. (2024), replacing their encoder nf2vec with ***OneField***. An MLP projector maps the frozen embedding into the token space of Llama-2-7B (Touvron et al., 2023), whose projected embedding is concatenated with the input prompt. We first train the projector alone, then jointly train the projector and language model, using the text annotations released by Amaduzzi et al. (2024; 2025). We evaluate brief and detailed captioning and QA. All baselines use the same pipeline, so the comparison isolates the representations. Table 4 reports sentence-level similarity using SBERT (Reimers & Gurevych, 2019) and SimCSE (Gao et al., 2021); additional NLP metrics are reported in Appendix A.6.

Table 4: **A language model conditioned on *OneField* embeddings describes either format equally well.** Sentence-level similarity between generated and reference text, via cosine similarity of SBERT and SimCSE embeddings. ***OneField*** is the only encoder that can be evaluated on a format it was not trained on, and it remains competitive with format-specific encoders.

Method	Projector trained on	Evaluated on	Brief description		Detailed description		Single-round QA	
			SBERT	SimCSE	SBERT	SimCSE	SBERT	SimCSE
nf2vec	NeRF	NeRF	75.96	77.20	78.77	81.00	81.96	82.43
OneField	NeRF	NeRF	77.93	79.03	79.74	81.75	82.51	83.08
OneField	NeRF	3DGS	**77.96**	**79.14**	**79.84**	**81.94**	**82.53**	**83.13**
ShapeSplat-GMAE	3DGS	3DGS	76.09	77.11	78.45	80.37	82.48	82.98
UniGS	3DGS	3DGS	**80.62**	**81.62**	**82.16**	**84.16**	**82.96**	**83.49**
OneField	3DGS	NeRF	77.51	78.59	79.57	81.66	82.51	83.09
OneField	3DGS	3DGS	77.95	79.09	80.20	82.17	82.55	83.14

OneField outperforms nf2vec and ShapeSplat-GMAE on every task, both on their native and cross-format evaluations. UniGS is the only competitor ahead of ***OneField***. However, UniGS uses language annotations when pre-training its 3DGS encoder and when training the projector and language model. Specifically, its encoder is pre-trained to align 3DGS embeddings with CLIP embeddings of the same textual annotations used here for language adaptation. Thus, UniGS receives language supervision in both stages, whereas ***OneField*** and the other methods see language only during projector and language-model training. Despite this, ***OneField*** remains within 2.67 SBERT points from UniGS on brief description, 1.96 on detailed description, and 0.41 on QA.

Cross-format transfer also holds for language generation, including the longest and least constrained outputs. Training the projector and language model on NeRF embeddings and evaluating on 3DGS, or vice versa, produces a maximum gap of only 0.63 SBERT points on detailed descriptions. Moreover, ***OneField*** transfers across formats while outperforming specialists on their native formats: trained on NeRFs and tested on 3DGS, it reaches 77.96 SBERT on brief descriptions versus 76.09 for ShapeSplat-GMAE; trained on 3DGS and tested on NeRFs, it reaches 77.51 versus 75.96 for nf2vec on NeRFs. Thus, even when the language pipeline is trained on one format, ***OneField*** can interpret the other without sacrificing performance relative to format-specific encoders.

5 Ablation results

The alignment objective of ***OneField*** has two terms: a SigLIP contrastive loss and a cross-modal VICReg term. To verify that both are needed, we retrain ***OneField*** with each term separately set to zero, keeping everything else fixed.

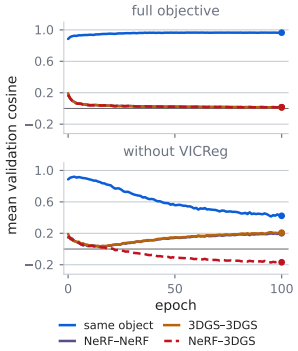


Figure 5: **VICReg ablation.**

VICReg contribution. A ranking objective can be satisfied by multiple geometries, and without VICReg the model converges to two separate but aligned spaces. As Figure 5 shows, the cosine similarity between the two embeddings of the same object decreases from 0.89 to 0.42, while random same-format pairs rise to $+0.20$ and random cross-format pairs fall to -0.17. Thus, the two formats cluster separately: a NeRF becomes closer to an unrelated NeRF than to an unrelated 3DGS, and its 3DGS counterpart is only 0.22 farther than that unrelated NeRF, versus 0.95 with the full objective.

Contrastive term. Without the contrastive term, VICReg has no mechanism to separate different objects: its alignment term acts on paired samples, while its variance and covariance terms operate independently of object identity. Alignment therefore remains intact, but neighborhoods become crowded, and R@1 on validation set drops from 92.5% to 83.2%. The gap shrinks to 3.7 points at $K = 5$ and 2.3 at $K = 10$, indicating that correspondence is preserved but the margin needed to rank the counterpart first is lost. The two terms are therefore complementary.

An additional analysis on the robustness of ***OneField*** to the views used for fitting the NeRFs and 3DGS is provided in Appendix A.10.

6 Conclusion

The format in which a radiance field is stored is a property of the pipeline that produced it, not of the object it describes, and downstream models should not depend on it. Processing NeRFs and 3DGS separately leaves two disjoint lines of work, with encoders that do not transfer and datasets that cannot supervise one another. Radiance fields should instead be treated as a single kind of data, with encoders, datasets, and benchmarks built for the field rather than for its parameterization.

OneField builds on this premise, proposing an encoder that learns a shared embedding space between these two formats. In this space, the format stops being a variable that downstream tasks must account for. A classification head trained on NeRFs can classify 3DGS. A segmentation head trained on NeRFs can label the points of a Gaussian splat, agreeing with itself on 97.7% of them. A language model conditioned on NeRF embeddings can describe objects it receives as Gaussian splats. A mixed NeRF-3DGS gallery returns the same object first, whichever format the query has. Throughout, ***OneField*** remains competitive with format-specific encoders.

Two limitations point to future work. First, compressing a field into a single global vector keeps the shared space compact but may limit its expressiveness. Tasks that need finer localization, such as dense correspondence, open-vocabulary grounding, or multi-part reasoning, likely need more capacity than one token. Second, ***OneField*** is trained and evaluated on object-centric data, whereas NeRFs and 3DGS are often fitted to scenes. A scene-level encoder must segment the field into objects, represent their spatial relations, and stay format-agnostic even though the two formats reconstruct backgrounds and unobserved regions differently.

AI USE STATEMENT

In this work, we used generative AI tools to edit software code, aid or polish writing and summarize or analyze existing literature. We have reviewed all AI-assisted work: AI-assisted code was read and tested by the authors, every number reported in this paper was produced by our own experiment code and was checked against the corresponding result file, and all AI-assisted text and references were verified by the authors.

We have not used generative AI tools to generate synthetic datasets, to develop theoretical models or conceptual frameworks, to formulate mathematical claims, to provide critical ingredients for proving them or to assist in the writing of proofs, to propose or refine the hypotheses of this work, to clean or reformat datasets, to support qualitative and thematic data analysis, or for translation. The ideas, the research question, the architecture, and the experimental design are our own.

We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Andrea Amaduzzi, Pierluigi Zama Ramirez, Giuseppe Lisanti, Samuele Salti, and Luigi Di Stefano. Llana: Large language and nerf assistant. *Advances in Neural Information Processing Systems*, 37:1162–1195, 2024.

Andrea Amaduzzi, Pierluigi Zama Ramirez, Giuseppe Lisanti, Samuele Salti, and Luigi Di Stefano. Scaling llana: Advancing nerf-language understanding through large-scale training. *arXiv preprint arXiv:2504.13995*, 2025.

Andrea Amaduzzi, Pierluigi Zama Ramirez, Giuseppe Lisanti, Samuele Salti, and Luigi Di Stefano. Spatially-aware weights tokenization for nerf-language models. *Advances in Neural Information Processing Systems*, 38:149595–149624, 2026.

Mido Assran, Adrien Bardes, David Fan, Quentin Garrido, Russell Howes, Matthew Muckley, Ammar Rizvi, Claire Roberts, Koustuv Sinha, Artem Zholus, et al. V-jepa 2: Self-supervised video models enable understanding, prediction and planning. *arXiv preprint arXiv:2506.09985*, 2025.

Francesco Ballerini, Pierluigi Zama Ramirez, Luigi Di Stefano, and Samuele Salti. Weight space representation learning on diverse nerf architectures. In *International Conference on Learning Representations*, volume 2026, pp. 122085–122108, 2026.

Satanjeev Banerjee and Alon Lavie. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pp. 65–72, 2005.

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.

Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015.

Matt Deitke, Dustin Schwenk, Jordi Salvador, Luca Weihs, Oscar Michel, Eli VanderBilt, Ludwig Schmidt, Kiana Ehsani, Aniruddha Kembhavi, and Ali Farhadi. Objaverse: A universe of annotated 3d objects. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 13142–13153, 2023.

Shuangkang Fang, I Shen, Takeo Igarashi, Yufeng Wang, ZeSheng Wang, Yi Yang, Wenrui Ding, Shuchang Zhou, et al. Nerf is a valuable assistant for 3d gaussian splatting. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 26230–26240, 2025.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 6894–6910, 2021.

Rohit Girdhar, Alaaeldin El-Nouby, Zhuang Liu, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Imagebind one embedding space to bind them all. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 15180–15190. IEEE, 2023.

Bernhard Kerbl, Georgios Kopanas, Thomas Leimkühler, George Drettakis, et al. 3d gaussian splatting for real-time radiance field rendering. *ACM Trans. Graph.*, 42(4):139–1, 2023.

Binxu Li, Yuhui Zhang, Xiaohan Wang, Weixin Liang, Ludwig Schmidt, and Serena Yeung-Levy. Closing the modality gap for mixed modality search. *arXiv preprint arXiv:2507.19054*, 2025a.

Haoyuan Li, Yanpeng Zhou, Tao Tang, Jifei Song, Yihan Zeng, Michael Kampffmeyer, Hang Xu, and Xiaodan Liang. Unigs: Unified language-image-3d pretraining with gaussian splatting. In *International Conference on Learning Representations*, volume 2025, pp. 86880–86900, 2025b.

Yue Li, Qi Ma, Runyi Yang, Huapeng Li, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Ender Konukoglu, Theo Gevers, et al. Scenesplat: Gaussian splatting-based scene understanding with vision-language pretraining. In *2025 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4961–4972. IEEE, 2025c.

Yue Li, Qi Ma, Runyi Yang, Mengjiao Ma, Bin Ren, Nikola Popovic, Nicu Sebe, Theo Gevers, Luc Van Gool, Danda Pani Paudel, et al. Chorus: Multi-teacher pretraining for holistic 3d gaussian scene encoding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 21431–21442, 2026.

Victor Weixin Liang, Yuhui Zhang, Yongchan Kwon, Serena Yeung, and James Y Zou. Mind the gap: Understanding the modality gap in multi-modal contrastive representation learning. *Advances in neural information processing systems*, 35:17612–17625, 2022.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Minghua Liu, Ruoxi Shi, Kaiming Kuang, Yinhao Zhu, Xuanlin Li, Shizhong Han, Hong Cai, Fatih Porikli, and Hao Su. Openshape: Scaling up 3d shape representation towards open-world understanding. *Advances in neural information processing systems*, 36:44860–44879, 2023.

Qi Ma, Yue Li, Bin Ren, Nicu Sebe, Ender Konukoglu, Theo Gevers, Luc Van Gool, and Danda Pani Paudel. A large-scale dataset of gaussian splats and their self-supervised pretraining. In *2025 International Conference on 3D Vision (3DV)*, pp. 145–155. IEEE, 2025.

Dawid Malarz, Weronika Smolak-Dyżewska, Jacek Tabor, Sławomir Tadeja, and Przemysław Spurek. Gaussian splatting with nerf-based color and opacity. *Computer Vision and Image Understanding*, 251:104273, 2025.

Ben Mildenhall, Pratul P Srinivasan, Matthew Tancik, Jonathan T Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. *Communications of the ACM*, 65(1):99–106, 2021.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311–318, 2002.

Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 652–660, 2017a.

Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017b.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Pierluigi Zama Ramirez, Luca De Luigi, Daniele Sirocchi, Adriano Cardace, Riccardo Spezialetti, Francesco Ballerini, Samuele Salti, and Luigi Di Stefano. Deep learning on object-centric 3d neural fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(12):9940–9956, 2024.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3982–3992, 2019.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Le Xue, Mingfei Gao, Chen Xing, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, and Silvio Savarese. Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 1179–1189. IEEE, 2023.

Le Xue, Ning Yu, Shu Zhang, Artemis Panagopoulou, Junnan Li, Roberto Martín-Martín, Jiajun Wu, Caiming Xiong, Ran Xu, Juan Carlos Niebles, et al. Ulip-2: Towards scalable multimodal pre-training for 3d understanding. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 27081–27091. IEEE, 2024.

Xumin Yu, Lulu Tang, Yongming Rao, Tiejun Huang, Jie Zhou, and Jiwen Lu. Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 19291–19300. IEEE, 2022.

Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 11941–11952. IEEE, 2023.

Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 16259–16268, 2021.

A Appendix

A.1 Fitting 3DGS

All the 3DGS presented in our dataset were fitted by us to match the NeRFs available in the dataset released by Amaduzzi et al. (2024; 2025). We fit the 3DGS on the same training views used for the NeRFs. Every object is fitted independently for $30{,}000$ steps using the photometric objective of Kerbl et al. (2023): an L1 term, and a D-SSIM term weighted 0.8 and 0.2. The points of each 3DGS are initialized from $5{,}000$ points randomly sampled from the original mesh of the object, as in Ma et al. (2025).

A.2 Robustness to splat resolution

The number of Gaussians an object is fitted with is a free parameter in our architecture, and one that the encoder does not control: objects holding more than $5{,}000$ splats are randomly subsampled to $5{,}000$ when they are loaded, and a batch is then formed at the smallest count among the objects it contains, $N = \min_b \min(M_b, 5{,}000)$, with every object being randomly subsampled to that size. Both subsamplings are drawn afresh each time an object is loaded, so an object above the batch size is seen with a different subset of its primitives at every epoch.

We therefore measure how much of the splat is needed for its embedding to still align with its NeRF counterpart. We take 500 ShapeNet test objects that hold more than $5{,}000$ Gaussians, subsample each splat uniformly to a given number, re-encode it, and compare the result against the NeRF embedding of the same object, which is left unchanged throughout.

Table 5 reports the results considering a varying number of primitives. Alignment saturates well below the maximum resolution: at 512 Gaussians the NeRF–3DGS similarity is already 98.6% of the value obtained with the full $5{,}000$, and from $1{,}024$ on, five times fewer primitives than the cap, it is indistinguishable from it. The spread shrinks in the same way, from 0.256 at 32 Gaussians to 0.066 from $1{,}024$ onward, so the agreement holds object by object and is not an average over a wide distribution. Only below roughly 128 Gaussians, the splat embedding drifts away from its NeRF counterpart, which is also the point at which too few primitives can no longer adequately describe the object.

Table 5: **Cross-modal alignment against the number of Gaussians.** Each object's 3DGS is subsampled to the given number of Gaussians and re-encoded, while its NeRF embedding is left unchanged; we report the mean per-object cosine similarity between the two over 500 ShapeNet test objects that hold more than $5{,}000$ Gaussians. The curve saturates at roughly $1{,}024$ Gaussians, five times fewer than the $5{,}000$ the encoder is trained with.

Gaussians	32	64	128	256	512	1024	2048	3072	4096	5000
Cosine sim.	0.410	0.665	0.828	0.896	0.927	0.938	0.940	0.941	0.940	0.941
Std	0.256	0.216	0.145	0.096	0.078	0.066	0.067	0.066	0.068	0.066

A.3 Implementation details

We train on 4 nodes with 4 NVIDIA A100 64 GB each, 16 GPUs in total. Every GPU holds 12 objects, so one step covers $|\mathcal{B}| = 192$ objects. The learning rate is set per GPU and scaled linearly with the number of GPUs, 10^{-6} each, and so 1.6×10^{-5} in total. We optimize with AdamW for 200 epochs of $1{,}531$ steps each. An epoch takes 24.6 minutes, so the full training time is about 82 hours or $1{,}312$ GPU-hours.

Gradients are averaged over the 16 GPUs. The embeddings are gathered across them before computing the two loss terms, so that $\mathcal{L}_{\text{siglip}}$ scores all 192^2 pairs and the variance and covariance of $\mathcal{L}_{\text{vic}}$ are estimated over 192 objects instead of 12; the gather is differentiable, so each gradient returns to the GPU that produced its embedding.

A.4 TRAINING OBJECTIVES

Section 3.5 introduced the four terms of the training objective. Here we provide their full formulations. In the subsequent notation, $\mathcal{B}$ represents a batch of objects, D the dimension of the embedding space, and $\boldsymbol{e}_{\mathrm{nf},i}$ and $\boldsymbol{e}_{\mathrm{gs},i}$ the two embeddings of the i-th object.

Alignment. The SigLIP loss (Zhai et al., 2023) scores each of the $|\mathcal{B}|^2$ NeRF–3DGS pairs of a batch:

$$\mathcal{L}_{\mathrm{siglip}} = -\frac{1}{|\mathcal{B}|}\sum_{i=1}^{|\mathcal{B}|}\sum_{j=1}^{|\mathcal{B}|}\ln\frac{1}{1+e^{-z_{ij}(\tau\,\boldsymbol{e}_{\mathrm{nf},i}\cdot\boldsymbol{e}_{\mathrm{gs},j}+\beta)}}, \tag{6}$$

where $z_{ij}=+1$ if $i=j$ and -1 otherwise, τ and β are learnable scalars. The two embeddings are ℓ_2-normalized before the dot product. Before computing the loss, the embeddings of all GPUs are gathered into a single batch of size $|\mathcal{B}|=192$.

We also employ a cross-format VICReg term (Bardes et al., 2022) to prevent the two formats from settling into two separate regions of the space. Let $\boldsymbol{E}_{\mathrm{nf}}, \boldsymbol{E}_{\mathrm{gs}} \in \mathbb{R}^{|\mathcal{B}|\times D}$ be the stack of embeddings for a batch; VICReg combines an invariance term s that pulls matched pairs together, a variance term v that keeps every dimension from collapsing, and a covariance term c that decorrelates dimensions:

$$s(\boldsymbol{E},\boldsymbol{E}') = \frac{1}{|\mathcal{B}|\,D}\sum_{i=1}^{|\mathcal{B}|}\left\|\boldsymbol{e}_i-\boldsymbol{e}'_i\right\|_2^2, \tag{7}$$

$$v(\boldsymbol{E}) = \frac{1}{D}\sum_{j=1}^{D}\max\Big(0,\ \gamma-\underbrace{\sqrt{\mathrm{Var}(\boldsymbol{E}^j)+\epsilon}}_{\text{standard deviation}}\Big), \tag{8}$$

$$c(\boldsymbol{E}) = \frac{1}{D}\sum_{k\neq l}\big[\mathrm{Cov}(\boldsymbol{E})\big]_{k,l}^2, \tag{9}$$

where $\boldsymbol{e}_i\in\mathbb{R}^D$ is the embedding of the i-th object, $\boldsymbol{E}^j$ is the j-th column of the embeddings' stack, and $\mathrm{Cov}(\boldsymbol{E}) = \frac{1}{|\mathcal{B}|-1}\sum_i(\boldsymbol{e}_i-\bar{\boldsymbol{e}})(\boldsymbol{e}_i-\bar{\boldsymbol{e}})^\top$ is the covariance matrix of the batch, $\bar{\boldsymbol{e}}$ its mean, $\gamma=1$ the target standard deviation and ϵ a small constant for numerical stability.

Only the invariance term is cross-format; variance and covariance are computed within each format:

$$\mathcal{L}_{\mathrm{vic}} = \lambda_s\, s(\boldsymbol{E}_{\mathrm{nf}},\boldsymbol{E}_{\mathrm{gs}}) + \frac{\lambda_v}{2}\big[v(\boldsymbol{E}_{\mathrm{nf}})+v(\boldsymbol{E}_{\mathrm{gs}})\big] + \frac{\lambda_c}{2}\big[c(\boldsymbol{E}_{\mathrm{nf}})+c(\boldsymbol{E}_{\mathrm{gs}})\big], \tag{10}$$

with $(\lambda_s,\lambda_v,\lambda_c)=(25,25,1)$ as in Bardes et al. (2022).

Reconstruction. Let $\mathcal{I}$ be the set of views an object was fitted on and $\mathcal{S}(I)$ a subset of pixels sampled from view $I\in\mathcal{I}$. For a pixel $\boldsymbol{p}$, let $C_{\mathcal{N}}(\boldsymbol{p})$ be the color rendered by the object's own NeRF and $C_{\mathrm{D}}(\boldsymbol{e},\boldsymbol{p})$ the color rendered by the decoder conditioned on an embedding $\boldsymbol{e}$ of that object. The rendering loss of that embedding is:

$$\mathcal{L}_{\mathrm{R}}(\boldsymbol{e}) = \sum_{I\in\mathcal{I}}\sum_{\boldsymbol{p}\in\mathcal{S}(I)}\mathrm{smoothL1}\big(C_{\mathcal{N}}(\boldsymbol{p}),\,C_{\mathrm{D}}(\boldsymbol{e},\boldsymbol{p})\big). \tag{11}$$

As in Ramirez et al. (2024), this loss is computed twice: $\mathcal{L}_{\mathrm{R}}^{\mathrm{fg}}$ on foreground pixels, and $\mathcal{L}_{\mathrm{R}}^{\mathrm{bg}}$ on points sampled from cells the occupancy grid marks as empty, whose target is the background color. The latter is what tells the decoder where the object is *not*, since the rays of the former only ever sample occupied cells.

We apply these two reconstruction losses to both embeddings of the object and average the two, so that $\boldsymbol{e}_{\mathrm{nf}}$ and $\boldsymbol{e}_{\mathrm{gs}}$ are supervised against the same target.

Full training objective. The loss terms detailed above are combined as follows:

$$\mathcal{L} = w_{\mathrm{fg}}\,\mathcal{L}_{\mathrm{R}}^{\mathrm{fg}} + w_{\mathrm{bg}}\,\mathcal{L}_{\mathrm{R}}^{\mathrm{bg}} + w_{\mathrm{c}}\,\mathcal{L}_{\mathrm{siglip}} + w_{\mathrm{v}}\,\mathcal{L}_{\mathrm{vic}}, \tag{12}$$

with $w_{\mathrm{fg}}=0.8$, $w_{\mathrm{bg}}=0.2$, $w_{\mathrm{c}}=0.02$ and $w_{\mathrm{v}}=0.01$. These weights were determined empirically.

A.5 Downstream heads and language projector

The classification and the part segmentation heads share the same two-layer MLP architecture where each hidden layer is followed by batch normalization, a ReLU, and dropout. The classifier head reads the embeddings and has an output dim of 13, which is the number of categories in ShapeNet. The segmentation head reads the embedding concatenated with the coordinates of a point and outputs the parts of one category: four for Airplane, Car, Chair and Lamp, and three for Table.

The language projector is a two-layer MLP, $1024 \rightarrow 2048 \rightarrow 4096$, which maps the embedding into the token space of Llama-2-7B. Its output is a single token, enclosed by two tokens `<point_start>` and `<point_end>`, added to the vocabulary.

A.6 Extended language metrics

Tables 6 to 8 report BLEU-1 (Papineni et al., 2002), ROUGE-L (Lin, 2004), and METEOR (Banerjee & Lavie, 2005) metrics for brief description, detailed description, and question answering, next to the sentence-similarity metrics of Table 4. These three metrics count the words the generated text shares with the reference, so a description that is right about the object but worded differently is penalized, while one that borrows the phrasing of the annotations is rewarded whether or not it is right.

Table 6: **Language evaluation on ShapeNeRF-Text – Brief Description (%).**

Method	Projector trained on	BLEU-1		ROUGE-L		METEOR		SBERT		SimCSE	
		NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS
nf2vec	NeRF	44.94	—	45.74	—	44.37	—	75.96	—	77.20	—
ShapeSplat-GMAE	3DGS	—	45.61	—	46.24	—	45.03	—	76.09	—	77.11
UniGS	3DGS	—	**48.72**	—	**50.13**	—	**48.95**	—	**80.62**	—	**81.62**
OneField	NeRF	46.57	46.53	**47.67**	47.91	**46.33**	46.42	**77.93**	77.96	**79.03**	79.14
OneField	3DGS	**46.59**	46.77	47.59	47.91	46.19	46.66	77.51	77.95	78.59	79.09

Table 7: **Language evaluation on ShapeNeRF-Text – Detailed Description (%).**

Method	Projector trained on	BLEU-1		ROUGE-L		METEOR		SBERT		SimCSE	
		NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS
nf2vec	NeRF	42.85	—	37.69	—	33.94	—	78.77	—	81.00	—
ShapeSplat-GMAE	3DGS	—	42.78	—	37.99	—	33.95	—	78.45	—	80.37
UniGS	3DGS	—	**44.58**	—	**40.02**	—	**35.83**	—	**82.16**	—	**84.16**
OneField	NeRF	**43.45**	43.32	**38.76**	38.52	**34.65**	34.60	**79.74**	79.84	**81.75**	81.94
OneField	3DGS	43.20	43.50	38.53	38.85	34.40	34.75	79.57	80.20	81.66	82.17

Table 8: **Language evaluation on ShapeNeRF-Text – Single-Round QA (%).**

Method	Projector trained on	BLEU-1		ROUGE-L		METEOR		SBERT		SimCSE	
		NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS	NeRF	3DGS
nf2vec	NeRF	48.04	—	55.12	—	52.03	—	81.96	—	82.43	—
ShapeSplat-GMAE	3DGS	—	48.57	—	55.87	—	52.46	—	82.48	—	82.98
UniGS	3DGS	—	**49.14**	—	**56.51**	—	**53.00**	—	**82.96**	—	**83.49**
OneField	NeRF	**48.91**	48.57	**56.09**	55.90	**52.72**	52.51	**82.51**	82.53	83.08	83.13
OneField	3DGS	48.78	48.91	56.08	56.19	52.61	52.79	**82.51**	82.55	**83.09**	83.14

Against the encoders that do not have language supervision before the projector stage, nf2vec on NeRF and ShapeSplat-GMAE on 3DGS, ***OneField*** is ahead on every task and every metric. The margin is widest on brief description and narrowest on question answering, where the answers are short and the three encoders end up within a point of one another.

UniGS preserves the benefit of its caption pre-training, mirroring the trend seen at the sentence-level metrics: a clear lead on brief description, a narrower one on detailed description, and near parity on question answering. As discussed in Section 4.6, its encoder is aligned to CLIP embeddings of the same annotations used here for language adaptation, so it is exposed to language in both stages, whereas ***OneField*** sees it only in the second.

Cross-format transfer holds on these metrics as well. For a given ***OneField*** row, the two evaluation columns stay closer together than the corresponding sentence-similarity columns, and ***OneField*** trained on one format and evaluated on the other still outscores the specialist built for that format.

Which representation the embedding is read from barely changes what the language model can say about the object.

A.7 SEGMENTATION QUALITATIVE RESULTS

Figure 6 compares part segmentation across methods on the same five objects, one for each ShapeNetPart category available in both formats.

The three baselines fail in three different ways, and the difference is informative because the head is the same in all cases and sees nothing but a global embedding and a coordinate. UniGS collapses on Car: the whole shape is predicted as a single part, roof and wheels included. nf2vec fails on Lamp: the boundary is in a plausible place, but the parts on either side of it get the wrong identity. ShapeSplat-GMAE is imprecise: on Chair it recovers back, seat and legs, but the seat spills over the tops of the legs and the seat-back junction sits too low.

OneField is the closest to the ground truth on four of the five objects. Its mistakes are slight boundary misalignments rather than collapsed or mislabeled parts.

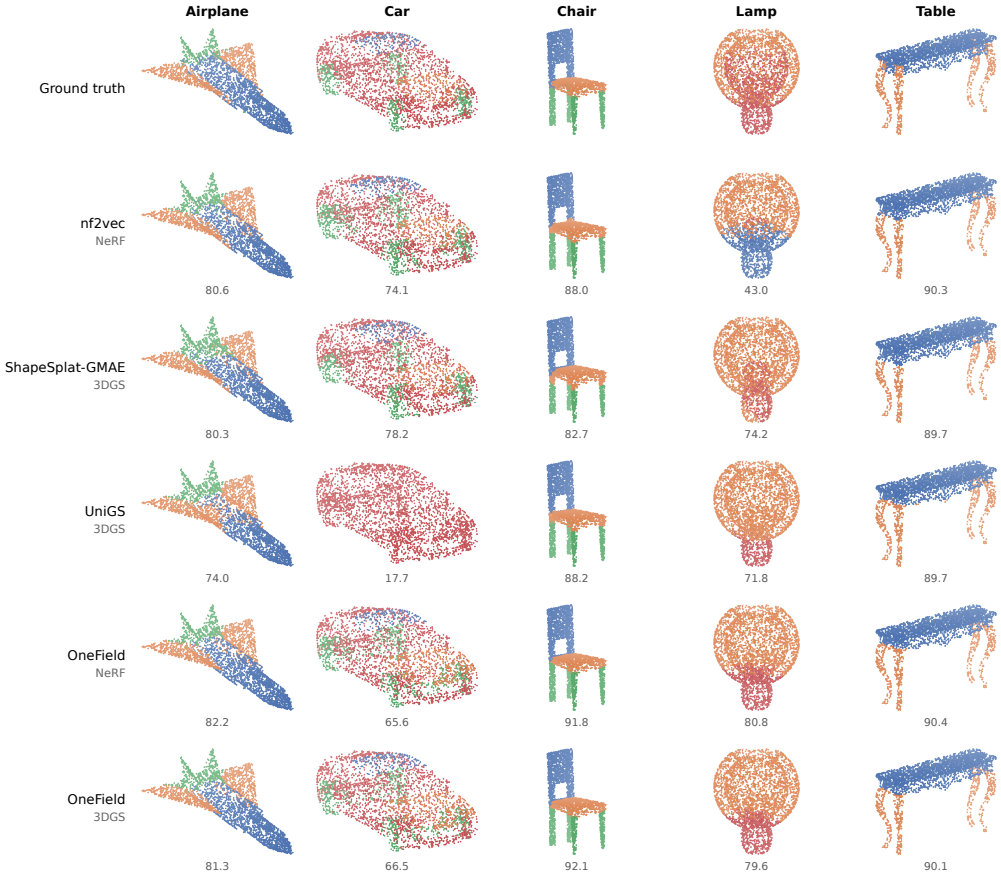


Figure 6: **Part segmentation, every encoder on the same objects.** Each row is a per-category head trained on that encoder's embeddings; numbers are per-shape mIoU. ***OneField*** uses one NeRF-trained head for both of its rows, so the 3DGS row is segmented by a head that never saw that format.

A.8 MORE EXAMPLES FROM RETRIEVAL GALLERY

Figure 7 shows an extended retrieval gallery for ***OneField***: five objects drawn from five different categories, each queried from both of its modalities against the same mixed gallery used in Section 4.4. What Figure 3 shows is not particular to that object. In all ten rows, rank 1 is the counterpart of

the query in the other modality, with a cosine between 0.95 and 0.99 and with a visible margin over rank 2, and the two rows of an object are almost the same list in almost the same order. Roughly half of the retrieved tiles are NeRFs and half are 3DGS in every block, so the ranking is not sorted by format.



Figure 7: **Extended visualization of the retrieval gallery:** five test objects from five categories, each queried from its NeRF and from its 3DGS embedding against a mixed gallery containing both modalities. Tiles are ordered left to right by cosine similarity. The top-ranked item is always the same object recovered in the other modality, at a cosine of 0.95 or above and clearly separated from the rest. Below rank 1, the two rows of an object list almost the same neighbors in almost the same order, the two modalities interleave throughout, and the retrieved objects repeat the geometry of the query and, for the car and the sofa, its color as well. The out-of-category hits (red) are a long low table among the sofas and two stretch limousines among the watercraft.

A.9 QUALITATIVE EXAMPLES OF NERF AND 3DGS

Figure 8 presents some qualitative examples of what our NeRFs and 3DGS look like next to one of the views they were fitted on. Each object appears from one of its 36 training views: the dataset image, then its NeRF and its 3DGS rendered from the camera that produced that image, so the three tiles of a row differ only in representation. All fits are faithful to their ground truth.



Figure 8: **NeRF and 3DGS rendered views:** Twelve random objects; each view is from one of its own training cameras.

A.10 VIEW SAMPLING ROBUSTNESS

Every object in our dataset is fitted twice from the same 36 images, so the NeRF and the 3DGS of an object are reconstructions of exactly the same views. We therefore verify that the encoder places the two formats close to each other because they describe the same object, and not because they were fitted from the same pixels.

We refit both representations from scratch on 100 random objects of the ShapeNet test split from controlled subsets of the 36 views, and measure the cosine similarity between the NeRF and the 3DGS embedding of each object. Table 9 reports the three settings. When both formats are refitted

on the 18 even views, the similarity is 0.829. When the NeRFs fitted on the even views are compared to the 3DGS refitted on the 18 odd views, so that the two formats share no views at all, the similarity is 0.832. The paired per-object difference is -0.003 ± 0.002.

Making the two view sets disjoint therefore costs nothing: the drop from the 0.936 of the full 36 views is caused by halving the number of views, which degrades both reconstructions equally, and not by the two formats seeing different images.

Table 9: **View-sampling robustness.** Each object's NeRF and 3DGS are refit from controlled subsets of its 36 views; we report mean per-object cosine similarity between the two embeddings

Scenario	Training views		Cosine sim.
	NeRF	3DGS	
Full density, same views	36 even+odd	36 even+odd	0.936 ± 0.110
Half density, same views	18 even	18 even	0.829 ± 0.172
Half density, disjoint views	18 even	18 odd	0.832 ± 0.169