

CONDENSE: Consistent 2D/3D Pre-training for Dense and Sparse Features from Multi-View Images

Xiaoshuai Zhang^{1,5*}, Zhicheng Wang⁵, Howard Zhou⁵, Soham Ghosh⁵,
Danushen Gnanapragasam⁵, Varun Jampani^{3,5*}, Hao Su^{1,4}, Leonidas Guibas^{2,5}

¹UC San Diego ²Stanford University ³Stability AI ⁴Hillbot ⁵Google Research

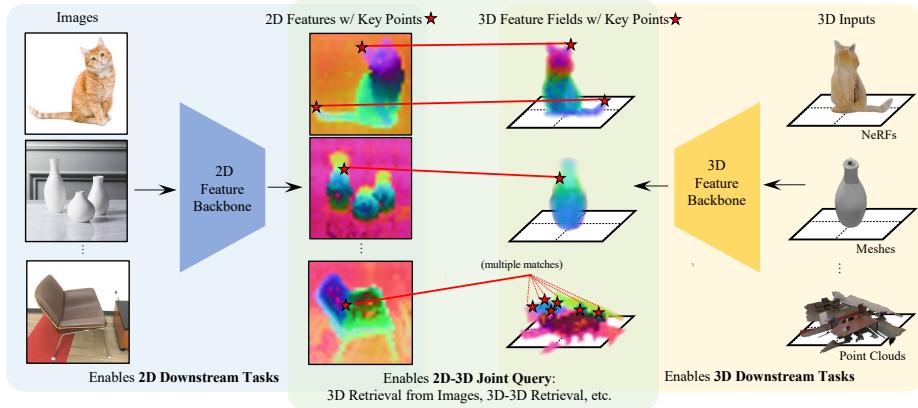


Fig. 1: CONDENSE extract co-embedded feature for 2D or 3D inputs. The model not only has improved performance over previous pre-training methods but also enables efficient cross-modality, cross-scale queries such as 3D retrieval and duplicate detection.

Abstract. To advance the state of the art in the creation of 3D foundation models, this paper introduces the CONDENSE¹ framework for 3D pre-training utilizing existing pre-trained 2D networks and large-scale multi-view datasets. We propose a novel 2D-3D joint training scheme to extract co-embedded 2D and 3D features in an end-to-end pipeline, where 2D-3D feature consistency is enforced through a volume rendering NeRF-like ray marching process. Using dense per pixel features we are able to 1) directly distill the learned priors from 2D models to 3D models and create useful 3D backbones, 2) extract more consistent and less noisy 2D features, 3) formulate a consistent embedding space where 2D, 3D, and other modalities of data (e.g., natural language prompts) can be jointly queried. Furthermore, besides dense features, CONDENSE can be trained to extract sparse features (e.g., key points), also with 2D-3D consistency – condensing 3D NeRF representations into compact sets of decorated key points. We demonstrate that our pre-trained model provides good initialization for various 3D tasks including 3D classification and segmentation, outperforming other 3D pre-training methods by

* Work conducted while at Google Research.

¹ Our project page.

a significant margin. It also enables, by exploiting our sparse features, additional useful downstream tasks, such as matching 2D images to 3D scenes, detecting duplicate 3D scenes, and querying a repository of 3D scenes through natural language – all quite efficiently and without any per-scene fine-tuning.

1 Introduction

The rapid advancement of 3D computer vision has led to significant breakthroughs in understanding and interpreting the world in three dimensions. However, achieving robust performance across a range of 3D perception tasks is challenging when we try to match the accomplishments of large pre-trained models in the natural language and 2D vision domains. The path to a 3D foundation model is hampered by the relative scarcity of 3D data compared to 2D images, and especially the increased difficulty of obtaining quality annotations in 3D. At the same time, 3D models need to co-exist and communicate with language or language-vision models, if we are to optimally use priors in perceiving, reasoning, and acting on the physical world. Motivated by these considerations we propose a novel approach for large-scale 3D pre-training that leverages the knowledge encoded in extant pre-trained 2D networks, capitalizes on the availability of large-scale multi-view datasets, and learns consistent 2D-3D features.

In this paper, we introduce a comprehensive 2D-3D joint training scheme, named CONDENSE, aimed at extracting co-embedded 2D and 3D features in an end-to-end pipeline. Our approach goes beyond conventional pre-training methods by enforcing 2D-3D feature consistency through 2D-3D consensus. This consistency is established by cross-checking the extracted 2D and 3D features via a ray-marching process inspired by Neural Radiance Fields (NeRFs), ensuring that the learned features align seamlessly in both the 2D and 3D domains.

In addition, CONDENSE represents the extracted features in two forms: a dense per-pixel representation and a sparse key point-based representation. This dual representation allows us to capitalize on the strengths of both types of features, making CONDENSE versatile and adaptable to a wide range of downstream tasks. Consider, for example, the fact the NeRFs have made the capture of 3D scenes a lightweight and widely available process. We will soon have hundreds of millions of objects and scenes in a NeRF form and the need arises to organize and search large collections of such data – and in particular to interrogate them using language and image queries. Our sparse key point representation and joint 2D-3D embeddings enable and facilitate such multi-modal cross-domain queries.

Our contributions can be summarized as follows:

- We propose CONDENSE, a novel 3D self-supervised pre-training scheme. By leveraging only large-scale 2D multi-view image datasets and 2D foundation models, we are able to achieve 3D pre-training with state-of-the-art downstream task performance, even when compared to 3D pre-training methods that utilize 3D training data.

- Our approach leads to the creation of more consistent and less noisy 2D features, enhancing the quality of existing 2D visual representations, and shows better performance than base models in various downstream 2D tasks.
- By learning sparse features jointly with the dense features for both 2D and 3D, we enable several novel tasks such as efficiently matching 2D images to larger-scale 3D scenes, or matching 3D captures of the same scene to each other.
- We establish a unified embedding space where 2D, 3D, and other modalities (e.g., natural language prompts) can be jointly queried, enabling efficient matching using either dense or efficient sparse features.

To validate the effectiveness of our large-scale pre-training approach, we conduct extensive experiments, showcasing its superior performance in various tasks. Furthermore, our pre-trained model opens up exciting possibilities for downstream applications, such as querying 3D scenes through natural language inputs or efficiently matching 2D images to 3D scenes, all without per-scene fine-tuning.

2 Related Work

2D Representation Learning and Foundation Models. Initial works on self-supervised 2D representation learning employ various pretext tasks derived from the images themselves [5, 38, 71], etc. Another line of work adopted discriminative strategies, such as instance classification [20], treating each image as a unique class and employing data augmentation for training. Recent advances in patch-based architectures, like Vision Transformers (ViTs) [17], revived interest in pretext tasks, particularly inpainting in both image and feature space. Various works find that masked-autoencoders (MAEs) provide strong initialization for downstream tasks [19]. However, all these pre-trained features require additional supervised fine-tuning. More recently, foundation models, referring to pre-trained models adept at a broad range of tasks, have seen expansive growth within the vision domain through variants that have successfully adapted to numerous vision-related tasks. Notably, the CLIP model [44] leverages contrastive learning from extensive image-text pairs to achieve zero-shot task transferability. DINO [6, 35] demonstrates the emergence of various desirable properties in its features through self-supervision, facilitating its direct application across diverse visual tasks.

3D Representation Learning and Foundation Models. Despite advances in 2D representation learning and foundation models, 3D models lag behind greatly due to dataset and architecture constraints. A major line of research proposed various pretext tasks for 3D point clouds [37, 74]. The recent success of ViTs in 2D has also spurred the exploration of their counterparts in 3D domains [29, 37, 68, 81]. However, all these models require 3D point clouds for pre-training. Most of them are pre-trained on ScanNet [14] (around 1000 scenes) and ShapeNet [7] (around 50k objects, synthetic), and are thus constrained by the limited amount of *real-world* data available.

2D to 3D Feature Distillation and Multi-Modality Embeddings. With the recent development of large-scale 2D foundation models and multi-modality embeddings (*e.g.*, CLIP for images and languages), many have tried to distill the knowledge learned from these models and extend their application to 3D data formats. PointCLIP and PartSLIP [31, 70] achieves zero-shot point cloud classification and segmentation by projecting point clouds to 2D depth maps and applying the 2D pre-trained models directly. OpenShape, ULIP, and ULIP-2 [30, 61, 62] collects text, image, and point cloud triplets and takes advantage of pre-aligned vision-language feature space to achieve alignment among the triplet modalities. Specifically, they fix the visual-language embedding space and only tune their 3D point cloud encoder to achieve this alignment. These methods focus on the task of point cloud classification and their design cannot be easily extended to other 3D tasks or 3D input formats. Several methods have been proposed for dense feature encoding in 3D [39, 78, 81]. For example, OpenScene [39] trains on point cloud-grounded multi-view datasets and learns a 3D point cloud network from multi-view aggregated 2D features. Like PointCLIP and ULIP, OpenScene requires 3D point clouds for training, which are scarce and hard to collect in the real world on a large scale, when compared to multi-view images. These works also re-use the embedding space from the 2D foundation model and only distill the 3D encoder.

More recently, Neural Radiance Fields (NeRFs) [33] and numerous subsequent follow-ups [3, 72] have gained great success in novel view synthesis. NeRF has the property of aggregating information across views. Several recent works leverage this property to improve the quality of semantic segmentation [25, 73, 75]. Many works distill features (*e.g.*, DINO [6] and CLIP [45]) into 3D and demonstrate they can be used for downstream tasks such as natural language-based query. These works require per-scene distillation and optimization. FeatureNeRF [65] proposed to distill features from 2D foundation models to 3D space via generalizable NeRFs [8, 66]. Through distillation, the learned model can lift any 2D images to continuous 3D semantic feature volumes. However, the pipeline serves more as a 2D-to-3D lifting technique and cannot handle native 3D data directly.

NeRF for Perception. The integration of Neural Radiance Fields (NeRF) into various discriminative perception tasks such as classification, detection, and segmentation has also been explored [52, 73]. These methods typically follow a reconstruction-then-detection pipeline by creating NeRFs from multi-view image data first and then designing task-specific networks and loss terms to tackle each perception task, and many of them work in a scene-by-scene manner and require re-training and optimization for each new scene. More recently, NeRF-Det [60] incorporates generalizable, feature-conditioned NeRF, and 3D detection pipelines to achieve efficient detection performance with no per-scene fine-tuning. Our work is inspired by these ideas while developing further by joining forces with 2D foundation models and creating a pre-training pipeline that is native to both 2D images and 3D formats.

Querying 3D Data. A variety of works have explored 3D to 3D similarity queries in pre-deep learning period, mostly at the object level, by constructing human-designed whole object features encoded as Euclidean embeddings or as distributions [9,36]. Later some works explored learned embedding spaces, as well as co-embeddings of images and 3D models [28]. Inspired by the bag-of-words paradigm in image search, “bags-of-features” have also been investigated in 3D to 3D search, for example [4]. However, none of these approaches is integrated into a multi-task framework, as we aim to do here and they are largely focused on object retrieval, not scenes.

3 Preliminaries

Neural Radiance Fields (NeRFs) [33] offer a novel representation of 3D scenes, capturing continuous volumetric scenes as neural networks. We briefly describe the mechanism of the NeRF and refer to [3,33] for details about related NeRF models. A NeRF \mathcal{F} maps a 3D coordinate $\mathbf{x} = (x, y, z)$ and a viewing direction $\mathbf{d} = (\theta, \phi)$ to a color $\mathbf{c} = (R, G, B)$ and density $\sigma - \mathcal{F} : (\mathbf{x}, \mathbf{d}) \mapsto (\mathbf{c}, \sigma)$, where color \mathbf{c} is related to both point location \mathbf{x} and viewing direction \mathbf{d} , recording the local appearance information, and density σ is only related to point location \mathbf{x} , recording the local geometry information.

The rendering of a 3D scene from a 2D perspective is formulated as a volume rendering problem. Given a ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, where \mathbf{o} is the camera origin and t is the distance along the viewing direction \mathbf{d} , the color $\mathbf{C}(\mathbf{r})$ of the ray is computed as:

$$\mathbf{C}(\mathbf{r}) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t))\mathbf{c}(\mathbf{r}(t), \mathbf{d}) dt, \quad T(t) = \exp\left(-\int_{t_n}^t \sigma(\mathbf{r}(s)) ds\right), \quad (1)$$

where $T(t)$ is the accumulated transmittance along the ray from t_n (near bound) to t_f (far bound). In practice, discretized approximations are used to evaluate this integral. The process is also called ray marching, which can be used as a general tool to render from any 3D feature field \mathbf{f} and get a 2D-projected feature map as shown in many previous works [52, 73, 75]. The property can be used to bridge the 3D feature field of a scene with its 2D feature maps and is the basis of our 2D-3D consensus pipeline.

2D Foundation Models are deep learning models that have been extensively pre-trained on large-scale image datasets. Let \mathcal{G} be a 2D foundation model that maps an input image \mathbf{I} to a feature representation $\mathbf{F}: \mathcal{G} : \mathbf{I} \mapsto \mathbf{F}$. The feature representation \mathbf{F} is a high-dimensional vector that captures the essential properties such as geometry and semantic information of the input image \mathbf{I} . Of particular interest, our work leverages DINO [6, 35], a self-supervised learning approach for visual representation trained on large-scale image datasets. Its latest version, DINOv2, excels in capturing both fine-grained details and global contextual information from images without any labeled data. By initializing the 2D encoder from DINOv2, we tap into a robust source of information-rich 2D dense features, and can thus kick-start our 3D encoder from 2D-3D knowledge distillation.

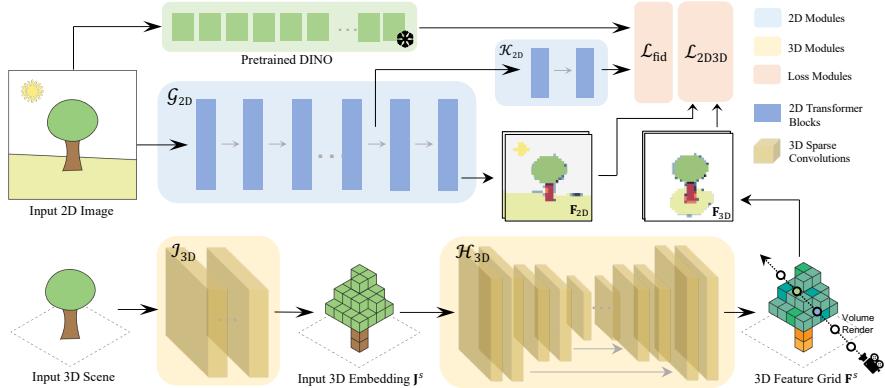


Fig. 2: Dense feature encoding: the 3D encoding module \mathcal{G}_{3D} is composed of a swappable input processing head \mathcal{J}_{3D} and a common 3D reasoning backbone \mathcal{H}_{3D} . \mathcal{J}_{3D} maps input 3D scenes of various formats into a feature \mathbf{J}^s in a unified 3D embedding space. \mathcal{H}_{3D} turns \mathbf{J}^s into a 3D feature grid \mathbf{F}^s . Through interpolation on \mathbf{F}^s and volume rendering, a 3D-projected feature map \mathbf{F}_{3D} can be obtained and compared with a 2D dense feature map \mathbf{F}_{2D} , extracted from the 2D encoding module \mathcal{G}_{2D} . The resulting 2D-3D consensus loss \mathcal{L}_{2D3D} is used as a self-supervision signal. An additional 2D fidelity loss \mathcal{L}_{fid} is introduced to make sure that the 2D-3D consensus optimized 2D feature \mathbf{F}_{2D} does not deviate too much from the original 2D feature in order to retain some of its semantics and visual richness.

4 Method

An overview of our approach is illustrated in Fig. 2 (dense feature encoding) and Fig. 3 (key point prediction). Our model is composed of two branches, encoding 2D and 3D information, respectively (Sec. 4.1 and Sec. 4.2). Both branches encode features in two forms – dense format and key-point-based sparse format (Sec. 4.4). During training, we use paired 2D-3D inputs in the form of multi-view images and the corresponding NeRF scene. The information extracted in 2D and 3D branches is compared via 2D-3D consensus (Sec. 4.3) so that information can flow both ways.

4.1 2D Encoding

The 2D encoding branch (\mathcal{G}_{2D}) of the CONDENSE framework is crucial for extracting rich visual features from multi-view images, which later are learned synergistically with the 3D encoding module. We follow the network architecture of DINOv2 [35] to use ViT [17] as the base for our 2D encoder and load the pre-trained DINO weights for the initialization of our 2D branch. The ViT architecture takes as input a grid of non-overlapping contiguous image patches of resolution $N \times N$. In this paper, we use $N = 14$ (“/14” ViT models). The patches are then passed through a linear layer to form a set of embeddings. Following

previous works [6, 17], an extra learnable [CLS] token is added to the sequence to aggregate global information. These patch tokens and the [CLS] token are fed to the standard transformer blocks and updated by the attention mechanism.

For any given input image \mathbf{I} , the 2D branch generates a dense feature map $\mathbf{F}_{2D} = \mathcal{G}_{2D}(\mathbf{I})$. This feature representation \mathbf{F}_{2D} is a high-dimensional vector encoding various essential attributions of the input image, and can already be used out-of-the-box due to its pre-training on large-scale image datasets. We will later show that combined with our 2D-3D joint training, the feature branch can be further improved for various downstream tasks.

4.2 3D Encoding

The 3D encoding branch (\mathcal{G}_{3D}) of the CONDENSE framework is aimed at extracting a 3D feature field from various data formats. It is designed to be composed of two parts: $\mathcal{G}_{3D} = \mathcal{H}_{3D} \circ \mathcal{J}_{3D}$, where \mathcal{J}_{3D} is the input processing head and \mathcal{H}_{3D} is the actual 3D reasoning backbone. Different 3D data formats have their individual input processing heads, but they share a common 3D reasoning backbone \mathcal{H}_{3D} . The major 3D inputs we are dealing with are NeRF models, while we also support other data formats such as point clouds. Here we detail the full pipeline for 3D feature field reasoning from NeRF data.

Grid Sampling from NeRF. Given any learned NeRF function $\mathcal{F} : (\sigma, \mathbf{c})$, we sample uniformly on a 3D lattice within the normalized scene bounding box $[-1, +1]$, with spacing ϵ between samples:

$$\Sigma^s = \{\sigma(x), \text{s.t. } \mathbf{x} \in [-1 : \epsilon : 1]^3\}, \quad (2)$$

$$\mathbf{C}^s = \{\mathbf{c}(x, \mathbf{d}), \text{s.t. } \mathbf{x} \in [-1 : \epsilon : 1]^3, \mathbf{d} \in \mathcal{D}\}, \quad (3)$$

where \mathcal{D} is a predefined set of directions aiming to capture as much local appearance information as possible. In order to reduce computation costs, we use the density field from the input NeRF to sparsify these grids. Specifically, we calculate sample opacity by $\alpha(\mathbf{x}) = 1 - \exp(-\sigma(\mathbf{x}))$ due to the canceled-out spacing term δ_t in the regular grid [60], and filter out the point samples \mathbf{x} with $\alpha(\mathbf{x}) < \theta$. After sparsification, the evaluated sigma value and color values are concatenated for each grid sample and fed into the input processing head, which is a small 3D sparse convolutional network:

$$\mathbf{J}^s = \mathcal{J}_{3D}^{\text{NeRF}}(\text{Concat}(\Sigma^s, \mathbf{C}^s)). \quad (4)$$

Here \mathbf{J}^s serves as the input embedding for the 3D reasoning backbone, while different input processing heads take care of mapping data from different input sources to this input embedding space. For point cloud inputs, we first voxelize the data before feeding it into a small 3D network. Please check the appendix for more details.

3D Spatial Reasoning. We apply a 3D UNet [13] implemented with sparse convolution blocks to obtain a 3D feature grid \mathbf{F}^s from the aforementioned input embedding:

$$\mathbf{F}^s = \mathcal{H}_{3D}(\mathbf{J}^s). \quad (5)$$

The module enables reasoning in 3D space. To obtain the feature for an arbitrary 3D query point $\mathbf{x} \in \mathbb{R}^3$, we interpolate within the feature grid \mathbf{F}^s with a trilinear interpolation operator:

$$\mathbf{f}_{3D}(\mathbf{x}) = \text{TriLerp}(\mathbf{x}, \mathbf{F}^s). \quad (6)$$

4.3 2D-3D Consensus with 2D Fidelity

With the proposed 2D and 3D encoders, our model can generate co-embedded dense features given 2D or 3D inputs. During training, we use paired data of multi-view 2D images $\{\mathbf{I}_k\}$ and their corresponding learned NeRF $\mathcal{F} : (\sigma, \mathbf{c})$ to jointly train the 2D and 3D branches.

Specifically, for each scene, we first generate the 3D feature field \mathbf{f}_{3D} as detailed in Sec. 4.2. Based on this feature field and the scene density σ , we can adapt the rendering equation to render 3D-projected feature maps as in [52, 75]:

$$\mathbf{F}_{3D}(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) \mathbf{f}_{3D}(\mathbf{r}(t)) dt. \quad (7)$$

In the meantime, we generate the 2D feature map with our 2D branch: $\mathbf{F}_{2D} = \mathcal{G}_{2D}(\mathbf{I}_k)$ and adopt the consistency loss between the 3D rendered feature map \mathbf{F}_{3D} and the 2D-originated counterpart \mathbf{F}_{2D} with L_2 loss:

$$\mathcal{L}_{2D3D}(\mathbf{F}_{2D}, \mathbf{F}_{3D}) = \sum_{\mathbf{r} \in \mathcal{R}} \|\mathbf{F}_{2D}(\mathbf{r}) - \mathbf{F}_{3D}(\mathbf{r})\|_2^2. \quad (8)$$

\mathcal{R} denotes all camera rays in the multi-view image set of the scene that hit at least one active voxel in the 3D feature grid \mathbf{F}^s . The loss encourages information to flow both ways – the 3D branch could learn to generate useful 3D feature fields from the 2D multi-view supervision, and the 2D branch could also benefit from consistent underlying 3D geometry and learn to extract less noisy, multi-view consistent, and 3D-informed features.

Due to the biased and scarcer nature of the existing multi-view image datasets, if we optimize the networks based only on this 2D-3D consistency loss, the feature quality may degrade due to trivial solutions and biased data distribution. To prevent this, we propose to insert an additional output head called 2D fidelity head \mathcal{K}_{2D} before the second-to-last transfer block (see Fig. 2), and apply the 2D fidelity loss to keep its output \mathbf{F}_{2D}^{fid} from deviating too much from the original DINOv2 [6] feature:

$$\mathcal{L}_{fid}(\mathbf{F}_{2D}^{fid}) = \|\mathbf{F}_{2D}^{fid} - \mathbf{F}_{2D}^t\|_2^2, \quad (9)$$

where \mathbf{F}_{2D}^t is the pre-trained DINOv2 feature. No ground-truth labels are used in this loss, and this term can be applied on any natural image collection. We adapt ImageNet-21k [47] in our pipeline.

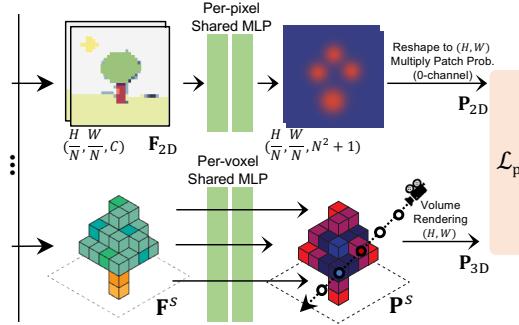


Fig. 3: Key point prediction: key points are detected in both 2D (\mathbf{P}_{2D}) and 3D (\mathbf{P}_{3D}) based on the existing feature backbones. The 2D-3D key point loss \mathcal{L}_p is used as a self-supervision signal.

4.4 Key Point Extraction

With the proposed 2D and 3D encoders and the joint training scheme, our model can generate co-embedded dense features given any input 2D image or 3D scene. This property is desired since it enables possible applications to query among 2D, 3D, and other modalities. To further facilitate these applications, we add support for sparse key point detection in both 2D and 3D based on the existing feature backbones for efficient queries across scene scales.

As shown in Fig. 3. To detect key points in 2D images, we follow a similar formulation as in [16] and decode the 2D backbone feature \mathbf{F}_{2D} into the full image-resolution interest point possibility map \mathbf{P}_{2D} with 2 MLP layers with a softmax output head (noted as \mathcal{M}_{2D}). Please refer to [16] and our Appendix for more in-depth details. For the 3D branch, we similarly use 2 MLP layers with a ReLU output head (noted as \mathcal{M}_{3D}) to decode the key point possibility on the 3D feature grid \mathbf{F}^s , and the possibility maps are rendered and compared between 2D and 3D using the aforementioned 2D-3D consensus scheme:

$$\mathbf{P}_{2D} = \mathcal{M}_{2D}(\mathbf{F}_{2D}), \quad (10)$$

$$\mathbf{P}_{3D}(\mathbf{r}) = \int_{t_n}^{t_f} T(t) \sigma(\mathbf{r}(t)) p_{3D}(\mathbf{r}(t)) dt, \quad (11)$$

$$\mathbf{P}^s = \mathcal{M}_{3D}(\mathbf{F}^s), \quad p_{3D}(\mathbf{x}) = \text{TriLerp}(\mathbf{x}, \mathbf{P}^s), \quad (12)$$

$$\mathcal{L}_p(\mathbf{P}_{2D}, \mathbf{P}_{3D}) = \sum_{\mathbf{r} \in \mathcal{R}} \|\mathbf{P}_{2D}(\mathbf{r}) - \mathbf{P}_{3D}(\mathbf{r})\|_2^2. \quad (13)$$

During test time, 3D key points are selected directly from opaque 3D grid samples. We multiply the opacity value by the predicted key point possibility and use $\alpha(x) \times p_{3D}$ as the selecting criteria for the 3D key points.

We argue that the joint 2D-3D key point detection not only helps enable various query tasks based on dense feature backbones (Sec. 5.3) but also serves as a useful technique to improve the overall feature quality (Sec. 6).

4.5 Training Details

Loss Terms. The final loss \mathcal{L} is given by:

$$\mathcal{L} = \lambda_{2D3D}\mathcal{L}_{2D3D} + \lambda_{fid}\mathcal{L}_{fid} + \lambda_p\mathcal{L}_p, \quad (14)$$

where λ_{2D3D} , λ_{fid} , and λ_p are scalars adjusted throughout the training process. Check the appendix for more details.

Datasets and Model Details. For all experiments included in the main experiments, we use MVImgNet [67], ScanNet [14], and RealEstate10k [79] as our multi-view pre-training datasets. MVImgNet is an object-centric dataset containing a total of 6.5 million frames from more than 200k video captures of diverse objects. ScanNet and RealEstate10k are indoor scene-scale datasets each containing a diverse set of scene captures in the form of video clips. Though other modalities are provided in some of these datasets (*e.g.* point clouds, semantic labels, etc.), we only use the posed images in pre-training. The scenes are fit into MipNeRF-360 [3] models individually before being used for pre-training. ImageNet-21k [47] is used to provide image samples for the 2D fidelity loss in addition to the multi-view datasets. ImageNet-21k is the superset of the commonly used ImageNet-1k dataset and contains more than 14 million images. We utilize Vision Transformers (ViT-g/14) as the backbone for the 2D branch and use 8 A100 GPUs for training. See the Appendix for more details on data pre-processing and model hyper-parameters.

Training Scheme. We bootstrap the full training process of CONDENSE in four stages. First, the 2D feature backbone \mathcal{G}_{2D} is initialized from DINOv2 [6] pre-trained weights. We then freeze its weights and fit the 2D key point detector MLPs (\mathcal{M}_{2D}) by enforcing the interest point heatmap predictions to a pre-trained, frozen SuperPoint [16] model. Then both \mathcal{G}_{2D} and \mathcal{M}_{2D} are kept frozen, while the 3D branch modules \mathcal{G}_{3D} and \mathcal{M}_{3D} are optimized from \mathcal{L}_{2D3D} and \mathcal{L}_p . In this stage, we distill the knowledge from the 2D foundation models to kick start the learning of 3D modules. For the final phase, we unfreeze all modules and jointly train all 2D and 3D modules with the loss terms defined in Eq. 14.

5 Experiments

In this section, we present extensive evaluations of our models on 1) 3D tasks including 3D classification, and 3D segmentation; 2) 2D image understanding tasks; and 3) Cross-Modality scene queries. In all experiments, we freeze weights for the feature backbones \mathcal{G}_{2D} and \mathcal{H}_{3D} unless otherwise stated. Due to the page limit, we only include the most common benchmarks in the main paper, please check the appendix for more experiments including 3D detection, 2D retrieval, and 2D depth estimation.

5.1 3D Tasks

For point-cloud-based 3D tasks, we use the 3D feature backbone \mathcal{H}_{3D} *out-of-the-box* and freeze its weights, while training a point cloud input head \mathcal{J}_{3D}^{PC} with 4

sparse convolutional layers. For a 3D point \mathbf{x} , we fetch the interpolated features in the 3D feature grid \mathbf{F}^s as the point feature (Eq. 6). Depending on the actual 3D tasks, different output heads could be added to further process these point features.

3D Classification We follow the testing protocols in the previous works [61, 62] to evaluate on ModelNet40 [58] and ScanObjectNN [51]. ModelNet40 is a synthetic dataset of CAD models containing around 10k training samples and 2.5k testing samples. ScanObjectNN is a real-world 3D dataset with around 15k objects extracted from indoor scans. We follow the same dataset setup and preparation protocols as in ULIP [61] to ensure consistent evaluation. We apply normalization on the point clouds before passing them into the input processing head and use a simple 3-layer sparse convolutional network with average pooling and 1-layer MLP output with softmax to predict the scene class. Only these two modules are trained with standard cross entropy loss on the target dataset and the 3D feature backbone is kept frozen. Results are in Tab. 1 on the left.

Table 1: Left: 3D classification results on ScanObjectNN (before slash) and ModelNet40 (after slash). **Right:** 3D segmentation results (mIoU) on ScanNet and S3DIS. For both tasks, CONDENSE outperforms all the baselines including train-from-scratch methods and pre-training methods. CONDENSE outperforms all the baselines including train-from-scratch and pre-training methods.

Model	Overall Acc	Cls-mean Acc	Model	ScanNet	S3DIS
PointNet [40]	68.2 / 89.2	63.4 / 86.0	PointNet++ [41]	53.5	54.5
PointNet++ [41]	77.9 / 90.7	75.4 / --	MinkowskiNet [12]	72.2	65.4
DGCNN [55]	78.1 / 92.9	73.6 / 90.2	PointCNN [27]	--	65.4
MVTN [18]	82.8 / 93.8	-- / 92.0	KPConv [50]	69.2	70.6
PointMLP [32]	85.7 / 94.1	84.4 / 91.3	PointNeXt [43]	71.5	74.9
PointNeXt [43]	87.5 / --	85.9 / --	PointMetaBase [29]	72.8	77.0
Point2Vec [69]	87.5 / 94.8	86.0 / 92.0	PointVector [15]	--	78.4
ULIP (w/ PointMLP) [61]	88.8 / 94.3	87.8 / 92.3	PointContrast [59]	74.1	--
ULIP-2 (w/ PointNeXt) [62]	90.8 / --	90.3 / --	MSC (w/ SparseUNet) [57]	75.5	--
ReCon [42]	90.6 / 94.7	-- / --	PPT (w/ SparseUNet) [56]	76.4	78.1
PointGPT [10]	93.4 / 94.9	-- / --	PonderV2 (w/ SparseUNet) [80]	77.0	79.9
CONDENSE (Ours)	94.1 / 95.2	93.4 / 93.1	Swin3D [63]	77.5	79.8
			CONDENSE (Ours)	79.8	80.7

3D Segmentation We follow the testing protocols in the previous works [43, 63] to evaluate on the ScanNet [14] and the S3DIS [2] datasets. ScanNet contains 1613 indoor scans with 20 semantic classes, we train on its train split and report the mean Intersection over Union (mIoU) on the validation split. S3DIS contains 272 scenes, we train on its training split and evaluate on its validation set with the 6-fold cross-validation scheme. The voxel sizes for ScanNet and S3DIS are set to 2cm and 5cm, respectively. We extract the point feature from the backbone with trilinear interpolation (Eq. 6) and use a simple linear layer with softmax to predict the point label. Only the input processing head and the linear layer are trained with standard cross entropy loss on the target dataset and the 3D feature backbone is kept frozen. Results are presented in Tab. 1 on the right.

Summarizing the results in Tab. 1, our method has demonstrated superior performance compared to other train-from-scratch and pre-training frameworks on both 3D classification and segmentation tasks. Despite only tuning the input and output heads, our approach still surpasses the performance of pre-trained methods that fine-tune the entire model. Furthermore, while many other methods heavily utilize point cloud data during pre-training [61, 62], our approach achieves remarkable results without this requirement.

5.2 2D Tasks

To evaluate the performance of the pre-trained 2D feature backbone \mathcal{G}_{2D} , we follow the settings as presented in DINOv2 [35], and compare with common self-supervised pre-trained baselines including MAE [19], DINO [6,35], and iBOT [77], as well as weakly supervised visual-language pre-trained model OpenCLIP [23]. For both classification and segmentation, we present results under the “Linear (lin.)” setting [6,35]. We include more results under the “multi-scale (+ms)” setting and more 3D benchmarks in our appendix. Results are presented in Tab. 2.

2D Classification We test the quality of the holistic image representation produced by the model on the ImageNet-1k [48] and Places205 [76] classification dataset. A linear probe is added on top of the frozen feature backbone to generate the prediction, following previous works [6,35].

2D Segmentation We test on the task of semantic image segmentation to evaluate the quality of our learned representation. A linear layer is trained to predict class logits from patch tokens and it is upscaled to obtain the final segmentation map, following previous works [6,35].

Table 2: 2D classification and segmentation results on multiple evaluation datasets with frozen features. CONDENSE improves over the DINOv2 in all benchmarks.

Model	Classification (Acc) Segmentation (mIOU)			
	ImageNet	Places205	ADE20k	PascalVOC
OpenCLIP [23]	86.2	69.8	39.3	71.4
MAE [19]	76.6	52.4	33.3	67.6
DINO [6]	79.2	60.4	31.8	66.4
iBOT [77]	82.3	64.4	44.6	82.3
DINOv2 [35]	86.5	67.5	49.0	83.0
CONDENSE	89.6	70.2	53.6	85.1

For both 2D classification and segmentation benchmarks, our 3D-informed CONDENSE shows consistent improvement over the original DINOv2 and indicate that our 2D-3D consensus training pipeline can help improve the performance of the existing 2D foundation models

5.3 Cross-Modality Scene Query

Leveraging the joint 2D-3D co-embedding property of CONDENSE, we are able to query across modalities. Here we tackle a series of matching tasks including

scene identification from 2D images and a newly proposed task – 3D scene duplication detection. The results are presented in Tab. 3. Dataset and test details are covered in the Appendix. We also include more experiments of 2D-3D joint query, including instance retrieval in real-world 3D scenes with 2D exemplars (*e.g.* CAD renderings), and query with natural languages in the Appendix.

3D Scene Retrieval with a Single Image (2D-3D). In this task, we retrieve a scene from a repository with a single view. To compare with 2D-only methods, we first render 5 views (Ren5) from a scene and compute the cosine similarity of the query image and rendered views, and then use the winner-take-all scheme to identify the scene. In Tab. 3, it can be seen CONDENSE 2D is a strong baseline not only outperforming all the other 2D methods but also performing better than ULIP-2. In 2D-3D methods, both global (using globally averaged feature) and KP (key point matching with RANSAC) variants of CONDENSE do better than the other 2D-3D method.

Table 3: 3D scene retrieval and 3D scene duplicate detection results on multiple datasets with frozen features. The upper includes 2D solutions where scenes are represented by 5 random views (Ren5), and the lower includes 2D-3D native methods.

Model	3D Retrieval (Acc)		3D Dup. Det. (AP ₇₅)	
	Objectron	ScanNet	ScanNet	Replica
OpenCLIP (Ren5) [23]	90.3	49.8	51.0	52.7
DINOv2 (Ren5) [35]	88.1	43.1	41.3	43.3
Unicom (Ren5) [1]	92.9	52.5	54.3	57.0
CONDENSE 2D (Ren5)	94.6	53.3	58.7	59.0
ULIP-2 [62]	89.7	61.7	63.0	66.6
CONDENSE-Global	91.6	70.1	65.3	66.9
CONDENSE-KP	92.9	78.4	70.7	72.0

3D Scene Duplicate Detection (3D - 3D). We further test the matching capabilities of CONDENSE at the scene level by proposing a new task of detecting duplicate scenes in a large NeRF repository. Our method is generalizable to both NeRF and Point-Cloud inputs, and we run our experiments on ScanNet and Replica [14, 49]. Results are presented in Tab. 3. Here, Ren5 methods are similarly defined as in the previous 3D scene retrieval, where scenes are rendered into images and the image embeddings are used. We use $\theta = 0.75$ as the threshold to determine if two embeddings belong to the same scene. Here we find the remarkable effectiveness of our key points. There is a large gap between 2D-3D methods for this task, showing the need to tackle this task in 3D feature space. While CONDENSE-Global performs similarly to ULIP-2, CONDENSE-KP is significantly better for scene-to-scene matching.

6 Ablation Studies and Discussions

In this section, we perform experiments to verify the effectiveness of our design. The ablation study results are presented in Tab. 4. **Freeze the 2D encoder?**

Table 4: Ablation study on removing individual components in our pre-training pipeline, evaluated on both 3D classification (Overall Acc on ScanObjectNN, before slash) and 2D classification (Acc on ImageNet-1k, after slash).

Full Model	Freeze 2D	No \mathcal{L}_p	No \mathcal{L}_{fid}	10% Data
94.1 / 89.6	91.3 / 86.5	90.5 / 87.1	89.7 / 79.9	88.7 / 79.1

When we freeze the 2D encoder, as is done in other methods [39, 61], we observed a worse performance in performance for both 2D and 3D tasks. We can also see that the features from our backbone are more 3D consistent and contains more detail. Please check the visualization in our supplementary materials.

Sparse feature helps? The sparse feature module is an integral component of our framework, not only enabling novel capabilities in 2D-3D retrieval but also serving as a strong self-supervision signal to enhance performance on individual 2D and 3D tasks. **2D fidelity helps?** The 2D fidelity loss helps prevent the 2D features from collapsing to a trivial solution or overfitting the biased data distribution. The exclusion of the 2D fidelity module has a detrimental impact on the quality of both 2D and 3D tasks, as evidenced by our experiments. Part of this loss is due to multi-view datasets being still considerably smaller than 2D image datasets, and featuring mostly man-made objects and indoor scenes. The limited size and biased distribution can cause the features to deviate significantly, leading to worse results.

7 Conclusions

In this work, we have presented CONDENSE, a framework for 3D pre-training that adeptly harmonizes 2D and 3D feature extraction using pre-trained 2D networks and multi-view datasets, in both the dense and the sparse feature regimes. Our approach not only provides a pre-trained 3D network acting in multiple tasks but also enhances the quality of 2D feature representation and establishes a unified embedding space for multi-modal data interaction. Extensive experiments demonstrate the superiority of CONDENSE over existing 3D pre-training methods in tasks like 3D classification and 3D segmentation and in new applications such as 2D image queries of 3D NeRF scenes. CONDENSE marks an advance in 3D computer vision, reducing the reliance on scarce 3D data and enabling more efficient and multi-modality queries on 3D scenes.

Nonetheless, the pipeline comes with several limitations including the relatively high cost of processing multi-view data. Exploring more efficient ways to exploit multi-view data as well as ready-to-use 3D data could be a useful future direction. Furthermore, combining techniques from contrastive learning methods [21, 59] and efficient fine-tuning methods (*e.g.* LoRA [22]) could improve the overall robustness and efficiency of the pipeline. We believe that CONDENSE opens up exciting avenues for model pre-training and 2D/3D feature backbones, and defer these to future research in this direction.

Acknowledgements

Our thanks go to Hao-Ning Wu and Bhav Ashok for their support in building large-scale pose estimation and NeRF pipeline for dataset pre-processing.

References

1. An, X., Deng, J., Yang, K., Li, J., Feng, Z., Guo, J., Yang, J., Liu, T.: Unicom: Universal and compact representation learning for image retrieval. arXiv preprint arXiv:2304.05884 (2023)
2. Armeni, I., Sener, O., Zamir, A.R., Jiang, H., Brilakis, I., Fischer, M., Savarese, S.: 3d semantic parsing of large-scale indoor spaces. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1534–1543 (2016). <https://doi.org/10.1109/CVPR.2016.170>
3. Barron, J.T., Mildenhall, B., Verbin, D., Srinivasan, P.P., Hedman, P.: Mip-nerf 360: Unbounded anti-aliased neural radiance fields. CVPR (2022)
4. Bronstein, A.M., Bronstein, M.M., Guibas, L.J., Ovsjanikov, M.: Shape google: Geometric words and expressions for invariant shape retrieval. ACM Transactions on Graphics (TOG) **30**(1), 1–20 (2011)
5. Carlucci, F.M., D’Innocente, A., Bucci, S., Caputo, B., Tommasi, T.: Domain generalization by solving jigsaw puzzles. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2229–2238 (2019)
6. Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., Joulin, A.: Emerging properties in self-supervised vision transformers. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 9650–9660 (2021)
7. Chang, A.X., Funkhouser, T., Guibas, L., Hanrahan, P., Huang, Q., Li, Z., Savarese, S., Savva, M., Song, S., Su, H., Xiao, J., Yi, L., Yu, F.: ShapeNet: An Information-Rich 3D Model Repository. Tech. Rep. arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago (2015)
8. Chen, A., Xu, Z., Zhao, F., Zhang, X., Xiang, F., Yu, J., Su, H.: Mvsnerf: Fast generalizable radiance field reconstruction from multi-view stereo. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 14124–14133 (2021)
9. Chen, D.Y., Tian, X.P., Shen, Y.T., Ouhyoung, M.: On visual similarity based 3d model retrieval. Computer Graphics Forum **22**(3), 223–232 (2003). <https://doi.org/https://doi.org/10.1111/1467-8659.00669>, <https://onlinelibrary.wiley.com/doi/abs/10.1111/1467-8659.00669>
10. Chen, G., Wang, M., Yang, Y., Yu, K., Yuan, L., Yue, Y.: Pointgpt: Auto-regressively generative pre-training from point clouds. arXiv preprint arXiv:2305.11487 (2023)
11. Chen, R., Han, S., Xu, J., Su, H.: Point-based multi-view stereo network. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 1538–1547 (2019)
12. Choy, C., Gwak, J., Savarese, S.: 4d spatio-temporal convnets: Minkowski convolutional neural networks. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 3075–3084 (2019)

13. Çiçek, Ö., Abdulkadir, A., Lienkamp, S.S., Brox, T., Ronneberger, O.: 3d u-net: learning dense volumetric segmentation from sparse annotation. In: Medical Image Computing and Computer-Assisted Intervention–MICCAI 2016: 19th International Conference, Athens, Greece, October 17–21, 2016, Proceedings, Part II 19. pp. 424–432. Springer (2016)
14. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner, M.: Scannet: Richly-annotated 3d reconstructions of indoor scenes. In: Proc. Computer Vision and Pattern Recognition (CVPR), IEEE (2017)
15. Deng, X., Zhang, W., Ding, Q., Zhang, X.: Pointvector: A vector representation in point cloud analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9455–9465 (2023)
16. DeTone, D., Malisiewicz, T., Rabinovich, A.: Superpoint: Self-supervised interest point detection and description (2018)
17. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2021)
18. Hamdi, A., Giancola, S., Ghanem, B.: Mvtn: Multi-view transformation network for 3d shape recognition. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 1–11 (2021)
19. He, K., Chen, X., Xie, S., Li, Y., Dollár, P., Girshick, R.: Masked autoencoders are scalable vision learners. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 16000–16009 (2022)
20. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
21. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 9729–9738 (2020)
22. Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W.: Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685 (2021)
23. Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., Schmidt, L.: Openclip (Jul 2021). <https://doi.org/10.5281/zenodo.5143773>, <https://doi.org/10.5281/zenodo.5143773>, if you use this software, please cite it as below.
24. Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanæs, H.: Large scale multi-view stereopsis evaluation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 406–413. IEEE (2014)
25. Kerr, J., Kim, C.M., Goldberg, K., Kanazawa, A., Tancik, M.: Lerf: Language embedded radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19729–19739 (2023)
26. Kerr, J., Kim, C.M., Goldberg, K., Kanazawa, A., Tancik, M.: Lerf: Language embedded radiance fields. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 19729–19739 (2023)
27. Li, Y., Bu, R., Sun, M., Wu, W., Di, X., Chen, B.: Pointcnn: Convolution on x-transformed points. Advances in neural information processing systems **31** (2018)
28. Li, Y., Su, H., Qi, C.R., Fish, N., Cohen-Or, D., Guibas, L.J.: Joint embeddings of shapes and images via cnn image purification. ACM transactions on graphics (TOG) **34**(6), 1–12 (2015)

29. Lin, H., Zheng, X., Li, L., Chao, F., Wang, S., Wang, Y., Tian, Y., Ji, R.: Meta architecture for point cloud analysis. arXiv:2211.14462 (2022)
30. Liu, M., Shi, R., Kuang, K., Zhu, Y., Li, X., Han, S., Cai, H., Porikli, F., Su, H.: OpenShape: Scaling Up 3D Shape Representation Towards Open-World Understanding. In: Annual Conference on Neural Information Processing Systems (NeurIPS) (2023)
31. Liu, M., Zhu, Y., Cai, H., Han, S., Ling, Z., Porikli, F., Su, H.: Partslip: Low-shot part segmentation for 3d point clouds via pretrained image-language models. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 21736–21746 (2023)
32. Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual mlp framework. arXiv preprint arXiv:2202.07123 (2022)
33. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: Nerf: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
34. Nekrasov, A., Schult, J., Litany, O., Leibe, B., Engelmann, F.: Mix3d: Out-of-context data augmentation for 3d scenes. In: 2021 International Conference on 3D Vision (3DV). pp. 116–125. IEEE (2021)
35. Oquab, M., Dariseti, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., et al.: Dinov2: Learning robust visual features without supervision. arXiv preprint arXiv:2304.07193 (2023)
36. Osada, R., Funkhouser, T., Chazelle, B., Dobkin, D.: Matching 3d models with shape distributions. In: Proceedings International Conference on Shape Modeling and Applications. pp. 154–166. IEEE (2001)
37. Pang, Y., Wang, W., Tay, F.E., Liu, W., Tian, Y., Yuan, L.: Masked autoencoders for point cloud self-supervised learning. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II. pp. 604–621. Springer (2022)
38. Pathak, D., Krahenbuhl, P., Donahue, J., Darrell, T., Efros, A.A.: Context encoders: Feature learning by inpainting. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 2536–2544 (2016)
39. Peng, S., Genova, K., Jiang, C., Tagliasacchi, A., Pollefeys, M., Funkhouser, T., et al.: Openscene: 3d scene understanding with open vocabularies. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 815–824 (2023)
40. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: 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 (2017)
41. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. Advances in neural information processing systems **30** (2017)
42. Qi, Z., Dong, R., Fan, G., Ge, Z., Zhang, X., Ma, K., Yi, L.: Contrast with reconstruct: Contrastive 3d representation learning guided by generative pretraining. arXiv preprint arXiv:2302.02318 (2023)
43. Qian, G., Li, Y., Peng, H., Mai, J., Hammoud, H., Elhoseiny, M., Ghanem, B.: Pointnext: Revisiting pointnet++ with improved training and scaling strategies. Advances in Neural Information Processing Systems **35**, 23192–23204 (2022)
44. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from

- natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
45. Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: International conference on machine learning. pp. 8748–8763. PMLR (2021)
 46. Ran, H., Liu, J., Wang, C.: Surface representation for point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 18942–18952 (2022)
 47. Ridnik, T., Ben-Baruch, E., Noy, A., Zelnik-Manor, L.: Imagenet-21k pretraining for the masses. arXiv preprint arXiv:2104.10972 (2021)
 48. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L.: ImageNet Large Scale Visual Recognition Challenge. International Journal of Computer Vision (IJCV) **115**(3), 211–252 (2015). <https://doi.org/10.1007/s11263-015-0816-y>
 49. Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J.J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H.M., Nardi, R.D., Goesele, M., Lovegrove, S., Newcombe, R.: The Replica dataset: A digital replica of indoor spaces. arXiv preprint arXiv:1906.05797 (2019)
 50. Thomas, H., Qi, C.R., Deschaud, J.E., Marcotegui, B., Goulette, F., Guibas, L.J.: Kpconv: Flexible and deformable convolution for point clouds. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 6411–6420 (2019)
 51. Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, D.T., Yeung, S.K.: Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: International Conference on Computer Vision (ICCV) (2019)
 52. Vora, S., Radwan, N., Greff, K., Meyer, H., Genova, K., Sajjadi, M.S., Pot, E., Tagliasacchi, A., Duckworth, D.: Nesf: Neural semantic fields for generalizable semantic segmentation of 3d scenes. arXiv preprint arXiv:2111.13260 (2021)
 53. Vu, T., Kim, K., Luu, T.M., Nguyen, T., Yoo, C.D.: Softgroup for 3d instance segmentation on point clouds. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2708–2717 (2022)
 54. Wang, H., Dong, S., Shi, S., Li, A., Li, J., Li, Z., Wang, L., et al.: Cagroup3d: Class-aware grouping for 3d object detection on point clouds. Advances in Neural Information Processing Systems **35**, 29975–29988 (2022)
 55. Wu, B., Liu, Y., Lang, B., Huang, L.: Dgcn: Disordered graph convolutional neural network based on the gaussian mixture model. Neurocomputing **321**, 346–356 (2018)
 56. Wu, X., Tian, Z., Wen, X., Peng, B., Liu, X., Yu, K., Zhao, H.: Towards large-scale 3d representation learning with multi-dataset point prompt training. arXiv preprint arXiv:2308.09718 (2023)
 57. Wu, X., Wen, X., Liu, X., Zhao, H.: Masked scene contrast: A scalable framework for unsupervised 3d representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9415–9424 (2023)
 58. Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J.: 3d shapenets: A deep representation for volumetric shapes. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1912–1920 (2015)
 59. Xie, S., Gu, J., Guo, D., Qi, C.R., Guibas, L., Litany, O.: Pointcontrast: Unsupervised pre-training for 3d point cloud understanding. In: Computer Vision–ECCV

- 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part III 16. pp. 574–591. Springer (2020)
60. Xu, C., Wu, B., Hou, J., Tsai, S., Li, R., Wang, J., Zhan, W., He, Z., Vajda, P., Keutzer, K., et al.: Nerf-det: Learning geometry-aware volumetric representation for multi-view 3d object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 23320–23330 (2023)
 61. Xue, L., Gao, M., Xing, C., Martín-Martín, R., Wu, J., Xiong, C., Xu, R., Niebles, J.C., Savarese, S.: Ulip: Learning a unified representation of language, images, and point clouds for 3d understanding. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1179–1189 (2023)
 62. Xue, L., Yu, N., Zhang, S., Li, J., Martín-Martín, R., Wu, J., Xiong, C., Xu, R., Niebles, J.C., Savarese, S.: Ulip-2: Towards scalable multimodal pre-training for 3d understanding. arXiv preprint arXiv:2305.08275 (2023)
 63. Yang, Y.Q., Guo, Y.X., Xiong, J.Y., Liu, Y., Pan, H., Wang, P.S., Tong, X., Guo, B.: Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. arXiv preprint arXiv:2304.06906 (2023)
 64. Yao, Y., Luo, Z., Li, S., Fang, T., Quan, L.: Mvsnet: Depth inference for unstructured multi-view stereo. In: Proceedings of the European conference on computer vision (ECCV). pp. 767–783 (2018)
 65. Ye, J., Wang, N., Wang, X.: Featurenerf: Learning generalizable nerfs by distilling foundation models. arXiv preprint arXiv:2303.12786 (2023)
 66. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: pixelnerf: Neural radiance fields from one or few images. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4578–4587 (2021)
 67. Yu, X., Xu, M., Zhang, Y., Liu, H., Ye, C., Wu, Y., Yan, Z., Liang, T., Chen, G., Cui, S., Han, X.: Mvimgnet: A large-scale dataset of multi-view images. In: CVPR (2023)
 68. Yu, X., Tang, L., Rao, Y., Huang, T., Zhou, J., Lu, J.: Point-bert: Pre-training 3d point cloud transformers with masked point modeling. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 19313–19322 (2022)
 69. Zeid, K.A., Schult, J., Hermans, A., Leibe, B.: Point2vec for self-supervised representation learning on point clouds. arXiv preprint arXiv:2303.16570 (2023)
 70. Zhang, R., Guo, Z., Zhang, W., Li, K., Miao, X., Cui, B., Qiao, Y., Gao, P., Li, H.: Pointclip: Point cloud understanding by clip. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8552–8562 (2022)
 71. Zhang, R., Isola, P., Efros, A.A.: Colorful image colorization. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part III 14. pp. 649–666. Springer (2016)
 72. Zhang, X., Bi, S., Sunkavalli, K., Su, H., Xu, Z.: Nerfusion: Fusing radiance fields for large-scale scene reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5449–5458 (2022)
 73. Zhang, X., Kundu, A., Funkhouser, T., Guibas, L., Su, H., Genova, K.: Nerflets: Local radiance fields for efficient structure-aware 3d scene representation from 2d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8274–8284 (2023)
 74. Zhang, Z., Girdhar, R., Joulin, A., Misra, I.: Self-supervised pretraining of 3d features on any point-cloud. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 10252–10263 (2021)
 75. Zhi, S., Laidlow, T., Leutenegger, S., Davison, A.J.: In-place scene labelling and understanding with implicit scene representation. In: ICCV (2021)

76. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Ghahramani, Z., Welling, M., Cortes, C., Lawrence, N., Weinberger, K.Q. (eds.) Advances in Neural Information Processing Systems. vol. 27. Curran Associates, Inc. (2014), <https://proceedings.neurips.cc/paper/2014/file/3fe94a002317b5f9259f82690aeea4cd-Paper.pdf>
77. Zhou, J., Wei, C., Wang, H., Shen, W., Xie, C., Yuille, A., Kong, T.: ibot: Image bert pre-training with online tokenizer. arXiv preprint arXiv:2111.07832 (2021)
78. Zhou, J., Wang, J., Ma, B., Liu, Y.S., Huang, T., Wang, X.: Uni3d: Exploring unified 3d representation at scale. arXiv preprint arXiv:2310.06773 (2023)
79. Zhou, T., Tucker, R., Flynn, J., Fyffe, G., Snavely, N.: Stereo magnification: Learning view synthesis using multiplane images. ACM Trans. Graph. (Proc. SIGGRAPH) **37** (2018), <https://arxiv.org/abs/1805.09817>
80. Zhu, H., Yang, H., Wu, X., Huang, D., Zhang, S., He, X., He, T., Zhao, H., Shen, C., Qiao, Y., et al.: Ponderv2: Pave the way for 3d foundataion model with a universal pre-training paradigm. arXiv preprint arXiv:2310.08586 (2023)
81. Zhu, X., Zhang, R., He, B., Guo, Z., Zeng, Z., Qin, Z., Zhang, S., Gao, P.: Pointclip v2: Prompting clip and gpt for powerful 3d open-world learning. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2639–2650 (2023)

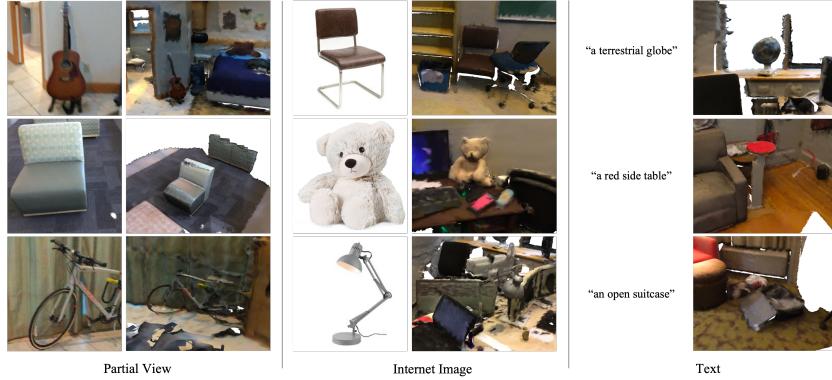


Fig. 4: Visualization of using different types of input to query the target scene repository (ScanNet). Within each pair are query inputs (left) and top-1 query results (right).

A Additional Details

A.1 Dataset Details

Generating NeRFs We use MVIImgNet [67], ScanNet [14], and RealEstate10k [79] for our 2D-3D pre-training. To generate NeRFs for our pre-training dataset, we use the MipNeRF-360 [3] official implementation to fit the scenes. For MVIImgNet, we trained 4000 steps for each scene. For ScanNet and RealEstate10k we trained 8000 steps for each scene. We half the image resolution before using it as training supervision. All the NeRF are fitted with 8 V100 GPUs. We find these settings are enough for generating NeRF with good qualities. We summarized the NeRF quality in terms of PSNR and SSIM in Tab. 5. It takes around 35000 V100 GPU hours to build these NeRFs, and we parallelize the process on around 1000 GPUs. For ScanNet, we are determining the bounding boxes with the ground-truth meshes. For MVIImgNet and RealEstate10k, we use the ray near-far values and camera locations to determine the scene bounds.

We acknowledge that fitting NeRFs on multi-view datasets requires significant computation. However, we believe this upfront cost is justified: (1) It enables 3D backbone pre-training without explicit 3D supervision, which is much more *time-consuming* to acquire. (2) Creating these datasets is a one-time process, and they can be shared among researchers to avoid repeated computation. (3) Faster rendering models have emerged since we developed the pipeline, especially the 3D Gaussian Splatting, which is even more suitable for our pipeline due to its sparsity nature. Adoption of these newer models could potentially cut the computational cost greatly.

2D Input Details For MVIImgNet [67], we use all 6.4M images in 214k scenes spreading over 238 classes for pre-training. Images are center-cropped and resized to 336×336 before being fed into 2D branch input. For ScanNet [14], we use all

Table 5: NeRF quality on the pre-training datasets with PSNR (before the slash) and SSIM (after the slash), tested on the sampled 1% images on each dataset’s native image resolution.

MVImgNet	ScanNet	RealEstate10k
30.23 / 0.939	25.78 / 0.901	27.24 / 0.922

1513 scans for training. We filter out blurred frames by calculating the variance of the Laplacian matrix and ignore them for the 2D branch inputs. Other images are randomly cropped in 336×336 before being used as 2D inputs. The same sampling and pre-processing schemes are applied for RealEstate10k [79]. For 2D fidelity loss, we sample from all 14M images in ImageNet-21k spreading over 21k classes. Images are resized in 336×336 before being processed by our 2D branch and output features were compared with the DINOv2 ViT-g/14 model [35] to enforce the 2D fidelity.

3D Input Details During training, we grid sample the NeRFs with resolution varying from $64 \times 64 \times 64$ to $256 \times 256 \times 256$ to ensure adaptability to different resolutions. These samples are then re-scaled to $128 \times 128 \times 128$, with 0.5 possibility of being sparse-dilated. The output 3D feature grid \mathbf{F}^s has the spatial resolution of $128 \times 128 \times 128$ in all experiments in this paper.

A.2 Network Details

3D Networks We follow the conventions and implementations as in MinkowskiNet [12] for all our 3D networks. Specifically, for input processing heads \mathcal{J}_{3D} , we apply 3 sparse convolution layers with “ $5 \times 5 \times 5 \times 1, 16$ ”, “ $5 \times 5 \times 5 \times 1, 32$ ” and “ $5 \times 5 \times 5 \times 1, 64$ ” configurations, similarly following the input processing of [12]. Here \times indicates a hypercubic sparse kernel. For the 3D reasoning backbone \mathcal{H}_{3D} , we apply the same architecture as MinkowskiUNet32 in [12] but remove the original input head and modify the number of output channels to match the 2D feature channels.

Key Point Prediction The process is illustrated in Fig. 3. Two different 2-layer MLPs are used for reasoning key point possibilities from 2D and 3D inputs.

A.3 Experiment Details

Computational Footprints For data preparation, we use V100 GPUs for fitting each scene. It takes around 35k V100 GPU hours. For pre-training, we use A100 GPUs and it takes around 4k A100 GPU hours, including both distillation and joint tuning stages. The total estimated power consumption is 12.1 MWh and carbon emitted is 5.8t CO₂eq.

Cross-Modality Scene Query We use 3 datasets: Objectron [51], ScanNet [14], and Replica [49] for cross-modality scene query tasks. To NeRF these scenes, we trained 4000 steps for Objectron and 8000 steps for both ScanNet and Replica on each scan. We use ground-truth bounding boxes included in these datasets. For “Ren5” baselines, we render images in the same resolution as in their corresponding datasets and the 5 views are randomly sampled from the original camera trajectories. So these 2D-Native methods are taking advantage of that queries and indices are drawn from the same trajectory, where in real-world cases this is not possible—since the original image sequences and trajectories are typically not accessible in 3D models. For ULIP-2 [62], we use its global scene embedding to perform the top-1 matching between the queries and indices. For our methods, we use 32 key points for each 2D image and 32, 64, and 64 key points for 3D input from Objectron, ScanNet, and Replica respectively. For our 2D-3D key point matching, we use the threshold 0.75 and select the 3D scene with the most number of successful matches as the query result.

3D Scene Retrieval with a Single Image (2D-3D). For Objectron [51], we sampled 1000 scenes spreading over 9 object categories. For each scene, we sample one image as the test query. We use top-1 to match between queries and keys and calculate the retrieval accuracy. For ScanNet [14], we use all 1513 scans. We sample one frame as a test query per 100 frames and at least one frame for every scene regardless of its length.

3D Scene Duplicate Detection (3D - 3D). For both ScanNet [14] and Replica [49], we sample 300 scene pairs where half of them are NeRFs from the same scene (duplicates) and half of them not. For ScanNet, the duplicates are created from different scans of the same scenes. (*e.g.* scene #0 has 3 different scans.) For Replica, the duplicates are created from the overlapping scans of the same scenes, where at least 50% of trajectory overlappings are ensured. In this way, we comprehensively test the 3D matching capability of models on either full scans or adjacent partial scans. We use 0.75 as the threshold when determining if two embeddings belong to the same scene as a way to detect duplicated scenes. AP₇₅* is calculated as the classification accuracy of duplicate detection when the threshold is 0.75.

Training and Loss Scheduling After initializing the 2D branches and fine-tuning the 2D key point detector, we train for 30k iterations with $\lambda_{\text{fid}} = 0$ while only tuning the 3D networks (distillation from 2D to 3D), where we sample 8 NeRF scenes in each iteration and sample rays within the original set of rays for supervision. After this stage, we train an additional 30k iterations with a linear warm-up of λ_{fid} and λ_p in 5k iterations where all 2D and 3D modules are jointly fine-tuned. Throughout the process, we use an AdamW optimizer, an initial LayerScale value of 1e-5, a weight decay cosine schedule from 0.02 to 0.24, a learning rate of 3.3e-4, and its warm-up of 2k iterations.

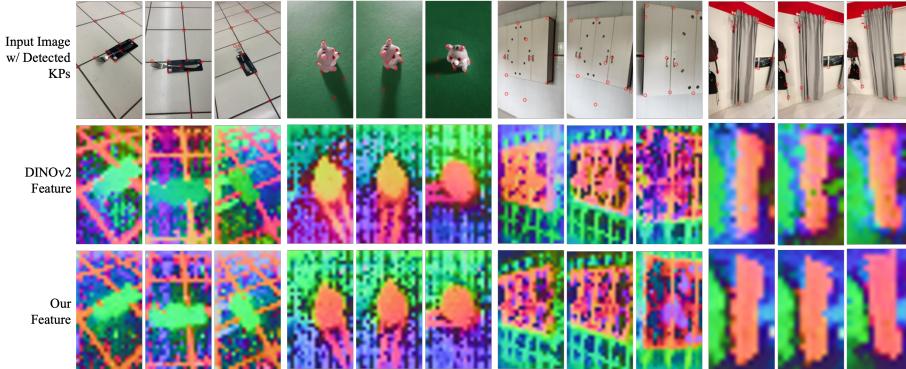


Fig. 5: Visualization of our 2D dense feature reveals its superiority over the Original DINOv2 feature in terms of consistency across multi-view images. Additionally, we present visualizations of sparse feature locations identified by our key point detector.

B Additional Experiments and Ablations

B.1 Feature Visualization

For Fig. 5, we ran PCA on the generated feature maps, both DINOv2 (2nd row) and ours (3rd row), of each scene and selected top-3 components to visualize them as red, green, and blue, respectively. We also visualize the top 10 confident (in terms of predicted probabilities) key points with positive 3D matches in this figure (1st row).

B.2 Cross-Modality Queries

We show in Fig. 4 the visualization of using different types of input to query the target 3D scene repository (ScanNet). The unique 2D-3D co-embedded embedding space with sparse key point design enables CONDENSE to effectively query objects in repositories of large scenes. Here, CONDENSE is not only able to query 3D scenes with partial views from the dataset but also able to find objects in these scenes that match the appearance from unseen internet images. In addition, by changing our backbone to a multi-scale CLIP [45] feature as in [26], we further acquire the ability to query 3D scenes with natural languages inputs, and thus build a language-image-3D co-embedded feature space with sparse key points. With this modified model, we can query scene repositories with text inputs as in Fig. 4 third row.

B.3 Additional 3D Experiments

In the main paper, we have done experiments on 3D classification and 3D segmentation and shown our performance on multiple datasets. Here we present additional 3D experiments on different tasks and more datasets.

3D Detection We show that our 3D features are also useful for 3D detection by following the settings of [63] to attach a state-of-the-art detection head CAGroup3D [54] and fine-tune the entire network for fair comparisions. The results are presented in Tab. 6. We provide an additional 2.1 and 2.9 points gain in terms of mAP@0.25 and mAP@0.5 respectively over an already strong baseline CAGroup3D. Our performance is also better than other pre-training methods [59, 63].

Table 6: 3D detection results (mAP@0.25 and mAP@0.5) on ScanNet.

Model	mAP@0.25	mAP@0.5
RepSurf [46]	71.2	54.8
SoftGroup [53]	71.6	59.4
CAGroup3D [54]	75.1	61.3
PointContrast [59]	59.2	37.3
Swin3D [63] w/ CAGroup3D	76.4	63.2
CONDENSE w/ CAGroup3D	77.2	64.2

Table 7: 2D retrieval results (top-1 Acc) on MVIImgNet and ReaEstate10k datasets.

Model	MVIImgNet	RealEstate10k
OpenCLIP [23]	70.1	63.0
MAE [19]	65.4	55.1
DINO [6]	65.6	58.9
DINOv2 [35]	70.1	61.3
CONDENSE	76.9	63.2

Table 8: 3D Classification (Acc) on MVIImgNet, Co3D, and ShapeNet. Our method is even more useful when 3D training data are scarce (ShapeNet 1%).

Method	MVIImgNet	Co3D	ShapeNet (1%)	ShapeNet (10%)	ShapeNet (Full)
From Scratch	73.5	81.1	61.2	76.9	84.9
PointContrast [59]	76.9	84.9	65.8	78.9	86.6
Point-MAE [37]	79.1	86.2	72.3	81.1	89.2
ULIP-2 [62]	82.9	86.1	74.3	81.9	89.3
Ours (Freeze 2D)	87.3	88.8	80.1	83.9	90.1
Ours	91.3	93.8	81.4	85.3	91.9

Table 9: 3D Segmentation (mIOU) on ScanNet, SemanticKitti, and S3DIS. Our method is even more useful when 3D training data are scarce (ScanNet 1%).

Method	ScanNet (1%)	ScanNet (10%)	ScanNet (100%)	SemanticKitti	S3DIS
PointNet [40]	42.2	62.1	72.2	19.6	47.6
Mix3D [34]	39.4	69.9	73.6	65.4	63.5
PointContrast [59]	52.9	70.4	74.1	71.7	75.2
Swin3D [63]	54.8	65.2	77.5	74.7	79.8
Ours (Freeze 2D)	65.6	70.0	78.1	74.6	80.6
Ours	67.3	72.3	79.8	75.1	80.7

More on 3D Classification and 3D Segmentation We tested our 3D capabilities on more datasets and also with varying amounts of 3D training data. The results are presented in Tab. 8 (classification) and Tab. 9 (segmentation). We see consistent improvement over all baselines from these results. Also, it can be seen that our method is even more useful when 3D training data is very scarce (See ShapeNet 1% and ScanNet 1%).

B.4 Additional 2D Experiments

In the main paper, we have done experiments on 2D classification and 2D segmentation and shown our performance on multiple datasets. Here we present additional 2D experiments on more tasks.

2D Retrieval We evaluate our performance for image retrieval with MVIImgNet [67] and ScanNet [14]. We follow the experiment settings of [6, 35] by freezing the features and directly applying k-NN for retrieval. On both datasets, we perform tests with 1 query image and 1 index image on each scene. The top-1 accuracy is reported in Tab. 7. Our CONDENSE clearly generates better global features suitable for retrieval tasks.

Depth Estimation We follow the settings in [35] and attach a linear classifier on top of one (lin. 1) or four (lin. 4) transformer layers to infer depth from the frozen feature backbones. It can be seen that our performance is better than all other pre-training backbones except DINOv2. The reason is that the 2D-3D consensus loss enforces the feature to be invariant across different views, and the 2D branch is thus expected to generate the same features for the same spatial point regardless of viewing angles and distances. The process of 2D-3D consensus facilitates more 3D-informed and consistent features, as can be observed from Fig. 5. Such a property could be desired or unwanted depending on the exact downstream tasks. And we defer the more in-depth study into this property to future research. To validate this, we show our results on multi-view stereo (MVS) depth estimation. Here we tested two widely used MVS depth estimation methods—MVSNet [64] and PointMVS [11], on the standard dataset (DTU [24]).

It can be seen that both backbones are boosted by our features, and CONDENSE outperforms DINoV2 by a large margin.

Table 10: Depth estimation with frozen features. We report performance when training a linear classifier on top of one (lin. 1) or four (lin. 4) transformer layers. We report the RMSE metric on the 3 datasets. Lower is better.

Method	NYUd		NYUd → SUN RGBD	
	lin. 1	lin. 4	lin. 1	lin. 4
OpenCLIP [23]	0.541	0.510	0.537	0.476
MAE [19]	0.517	0.483	0.545	0.523
DINO [6]	0.555	0.539	0.553	0.541
iBOT [77]	0.417	0.387	0.447	0.435
DINoV2 [35]	0.344	0.298	0.402	0.362
Ours	0.361	0.322	0.389	0.367

ff

Table 11: Stereo depth estimation on the DTU dataset. Backbone methods could benefit a lot from our feature initialization.

Model	Overall Err.
MVSNet [64]	0.462
PointMVS [11]	0.366
DINoV2 [35] w/ MVSNet	0.389
DINoV2 [35] w/ PointMVS	0.365
CONDENSE w/ MVSNet	0.341
CONDENSE w/ PointMVS	0.320

More on 2D Segmentation We test our performance on 2D segmentation following the “+ms” setting as proposed in [35]. The results are presented in Tab. 12.

ff

B.5 Additional Ablation Studies

Effect of NeRF Quality We observed differences in performance when using trained NeRFs of different quality. The numbers are reported on ImageNet (2D Cls) and ScanObjectNN (3D Cls) on a lighter version of the final model reported in the main paper.

Table 12: Semantic segmentation on ADE20K, CityScapes, and Pascal VOC with frozen features and a linear classifier (lin.) and with multiscale (+ms).

Method	ADE20k		CityScapes		Pascal VOC	
	lin.	+ms	lin.	+ms	lin.	+ms
OpenCLIP [23]	39.3	46.0	60.3	70.3	71.4	79.2
MAE [19]	33.3	30.7	58.4	61.0	67.6	63.3
DINO [6]	31.8	35.2	56.9	66.2	66.4	75.6
iBOT [77]	44.6	47.5	64.8	74.5	82.3	84.3
DINOv2 [35]	49.0	53.0	71.3	81.0	83.0	86.2
Ours	50.2	53.8	74.1	83.1	83.2	86.5

Table 13: Effect of NeRF Quality

Training NeRF Type	Mip-NeRF (2k steps)	Mip-NeRF (4k steps)
Quality (PSNR/SSIM)	28.54 / 0.899	30.23 / 0.939
2D Cls / 3D Cls	86.2 / 88.9	87.7 / 93.2

Results are presented in Tab. 13. We find that pre-training with data of lower quality will have an impact on performance, especially for 3D tasks. We take measures such as using different iteration numbers to ensure convergence, and filtering out low-quality frames as mentioned previously.

B.6 Commonly Used Backbones

The MinkowskiNet (MNet) is the established state-of-the-art for dense 3D tasks and has been adopted by most 3D dense task models. So we select MinkowskiNet as our backbone. Here we had an experiment comparing different backbones – results are reported below on ScanObjectNN (3D Cls) and ScanNet (3D Seg).

Table 14: Comparing different backbones.

	PointBERT	PointNeXt	MinkowskiNet
#Parameters	32.3M	41.6M	41.3M
3D Cls / 3D Seg	87.2 / -	92.1 / 77.0	93.2 / 79.1

We see better performance, especially on 3D Seg, with MNet. The reasons behind this may include (1) an overfitting tendency of the transformer-based model PointBERT, which aligns with OpenScene’s finding (their Sec. 6.4); (2) our better generalizability from training to test domains, where point distributions are different. We will include a more detailed discussion on the performance and scalability of different backbones once time permits.