BEAST3D: Animal behavioral analysis and neural encoding from multi-view video via Gaussian splatting

Yanchen Wang
Columbia University
yw4503@columbia.edu

Lenny Aharon
Columbia University
lenny.aharon@columbia.edu

Wangshu Zhu
Columbia University
wz2708@columbia.edu

Kyle Daruwalla
Cold Spring Harbor
daruwal@cshl.edu

Linghua Zhang
Cold Spring Harbor
lingzhan@cshl.edu

Jiaru Zou
Stanford University
jiaru@stanford.edu

Selmaan Chettih
Columbia University
sc4551@columbia.edu

Helen Hou
Cold Spring Harbor
hou@cshl.edu

Liam Paninski
Columbia University
lmp2107@columbia.edu

Matthew R Whiteway
Columbia University
m.whiteway@columbia.edu

Abstract

Multi-view video recordings are increasingly used to capture the 3D movements of animals in experimental settings, yet extracting rich 3D representations from these recordings remains challenging. Supervised pose estimation requires extensive manual annotation, while general-purpose 3D reconstruction models trained on generic scene datasets fail on the specialized imagery and sparse-view setting of laboratory experiments. We address these limitations with BEAST3D, a self-supervised pretraining framework that learns 3D visual representations from unlabeled, calibrated multi-view video. BEAST3D uses a vision transformer to predict 3D Gaussian splats that reconstruct held-out views through differentiable rendering, while simultaneously segmenting the animal from the background. BEAST3D reconstructs 3D structure with as few as four views by conditioning directly on known camera parameters—unlike general-purpose models, which must estimate camera geometry from dense overlapping viewpoints that are seldom available in lab settings. Through comprehensive evaluation across four species, we demonstrate that BEAST3D produces rich, viewpoint-invariant features that transfer effectively to three downstream tasks: *novel view synthesis*, which validates the quality of the learned 3D representations; *multi-view pose estimation*, which provides the sparse keypoint trajectories widely used in behavioral analysis; and *neural encoding*, which relates 3D behavioral features to simultaneously recorded neural activity. BEAST3D thus establishes a versatile framework for behavioral analysis that leverages 3D structure in modern multi-view laboratory recordings. Our project page and code: https://ppwangyc.github.io/projects/beast3d.

1 Introduction

Advances in neuroscience increasingly depend on precise quantification of animal movement, yet the single-view video recordings that dominate current practice are fundamentally limited: self-occlusions

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

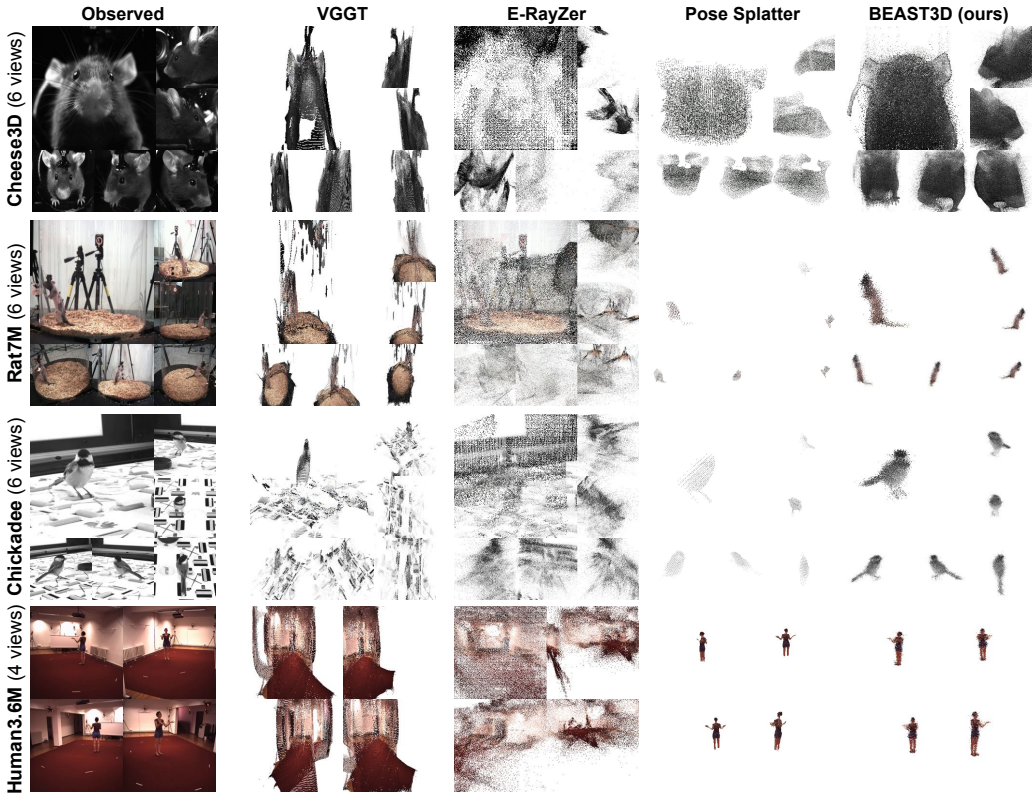

Figure 1: **3D point clouds from BEAST3D and leading baselines.** An example scene from diverse datasets (*left* column; Cheese3D [18], Rat7M [2], Chickadee [3], Human3.6M [19]) is encoded into a 3D point cloud by general-purpose models (VGGT [20], E-RayZer [21]) and tailored per-dataset models (Pose Splatter [22], BEAST3D). BEAST3D achieves strong performance while simultaneously providing foreground segmentation of the subject. The point cloud positions are predicted by each model, and each point is colored by the corresponding color on each pixel.

obscure body parts, and 2D observations cannot recover the full 3D kinematics of movement [1]. A growing number of laboratories address these limitations by recording with multiple synchronized and calibrated cameras, enabling accurate 3D measurements of constrained and freely moving animals [2, 3, 4, 5, 6]. The resulting multi-view data support a range of 3D analysis tasks, from sparse keypoint estimation that describes posture and joint kinematics [7, 8, 9, 10, 11, 12, 13, 14] to dense 3D meshes that capture the full surface geometry and articulation of the subject [15, 16, 17]. These explicit 3D representations offer a natural basis for relating behavior to neural activity, providing an interpretable and low-dimensional description of animal movement independent of camera viewpoint.

Despite the richness of multi-view recordings, current approaches to extracting 3D representations face important limitations. Supervised pose estimation methods require extensive manual annotation of keypoints, which is labor-intensive and must be repeated for each new experimental setup or species. Mesh-based approaches such as SMAL [15] provide richer surface representations, but require a species-specific template mesh and costly per-frame optimization, limiting their scalability across species and long recordings [16, 23] (though see [24]). Meanwhile, recent 3D computer vision models such as VGGT [20] and E-RayZer [21] can produce dense 3D reconstructions from images in a single forward pass, but these methods are trained on generic scene datasets and perform poorly on the specialized imagery of laboratory recordings: close-up views of animals under controlled lighting from fixed camera rigs (Fig. 1). Furthermore, these models are designed for settings with dense, highly overlapping viewpoints, and devote substantial capacity to estimating unknown camera parameters, which can fail with limited view overlap (Fig. 7). Neither assumption holds in typical laboratory camera rigs, which use a small number of widely spaced cameras with known calibration.

We address these gaps with BEAST3D, a self-supervised pretraining framework that learns 3D visual features from calibrated multi-view animal behavior videos. Given only unlabeled, synchronized multi-view images and corresponding camera parameters, BEAST3D predicts a set of 3D Gaussian

splats via a transformer trained to reconstruct held-out views. The model simultaneously segments the animal from the background by distilling masks from a video segmentation model, producing clean foreground representations. The self-supervised pretraining produces a flexible backbone that can be fine-tuned for multiple analytical needs. We demonstrate the quality of BEAST3D's features by comparing against state-of-the-art models across four species (mouse, rat, chickadee and human), showing substantially improved reconstructions with as few as four views (Fig. 1). We evaluate these features on three downstream tasks: (i) *novel view synthesis*, where a 3D scene constructed from a subset of views must reconstruct held-out views; (ii) *multi-view pose estimation*, where a lightweight head trained on BEAST3D features predicts a sparse set of keypoints; and (iii) *neural encoding*, where BEAST3D's features predict neural activity in the mouse and chickadee datasets. In all cases BEAST3D achieves competitive or superior performance compared to baselines including VGGT and DINOv3 [25]. These results establish BEAST3D as a versatile framework for extracting rich 3D features from multi-view laboratory recordings, bridging the gap between general-purpose 3D computer vision and the specialized demands of behavioral neuroscience.

2 Related Work

Self-supervised visual representation learning. Self-supervised pretraining has become the dominant paradigm for learning transferable visual features. Masked image modeling approaches, exemplified by MAE [26], train a vision transformer [27] to reconstruct randomly masked pixel patches. An alternative family of methods avoids pixel-level reconstruction. I-JEPA [28] predicts latent representations of masked regions from context, avoiding the low-level pixel bias of MAE-style objectives. The DINO family [29, 30, 25] learns representations through self-distillation with momentum teachers, producing features that capture semantic structure without explicit reconstruction. These methods learn powerful 2D features but have no mechanism for reasoning about 3D structure.

3D scene modeling. 3D Gaussian Splatting (3DGS; [31]) represents a scene as a set of 3D Gaussians—each with its own position, shape and color parameters—rendered via differentiable rasterization, achieving high visual fidelity. A key limitation of 3DGS (as well as earlier neural radiance field approaches, [32]) is the need for per-scene optimization: each new scene requires minutes of gradient descent to fit the representation. Recent work has sought to amortize this cost through feed-forward prediction, demonstrating the inductive bias of 3D Gaussian representations can be combined with the generalization capacity of transformers to enable single-forward-pass reconstruction [33, 34, 35]. Pose Splatter [22] applies feed-forward 3DGS to multi-view animal behavior videos, using "shape carving" to initialize a voxel grid that is refined by a 3D U-Net. While Pose Splatter demonstrates accurate 3D reconstructions, it differs from BEAST3D in key respects. First, Pose Splatter treats construction of a 3D latent representation as the end goal; BEAST3D instead uses novel view synthesis as a pretext task to learn a general-purpose backbone whose features transfer directly to downstream tasks. Second, Pose Splatter requires foreground segmentation as a preprocessing step for shape carving, whereas BEAST3D distills segmentation masks from a foundation model during training, so that inference requires only the raw multi-view images. Finally, BEAST3D emphasizes multi-session training that generalizes better to new subjects.

3D-aware pretraining from multi-view data. A separate line of work uses 3D geometry not as an end in itself (like 3DGS) but as an inductive bias for learning transferable visual features. RayZer [36] introduces ray-conditioned scene representations for novel view synthesis, using estimated Plücker ray coordinates [37] to encode camera geometry alongside image features. E-RayZer [21] extends this framework with an end-to-end architecture that jointly predicts camera poses and 3D Gaussians. VGGT [20] is a supervised model that predicts camera parameters, depth maps, and point clouds from a collection of images using alternating within- and across-view attention. These methods demonstrate that explicitly reasoning about 3D geometry during pretraining produces features that outperform purely 2D pretraining on geometry-sensitive downstream tasks. However, they are designed for general internet imagery where camera parameters are unknown and must be estimated jointly; models learn to do this by training on specialized datasets with dense camera coverage (often 10+ views, e.g., [38, 39]). In laboratory settings, sparse scene coverage with 3-6 cameras is more common, and accurate camera calibration is readily available. BEAST3D exploits this by removing the camera parameter estimation branch entirely and instead conditions on ground-truth camera parameters. This simplification reduces the number of trainable parameters, reduces the number of required views, and allows the model to focus capacity on learning appearance and geometry features.

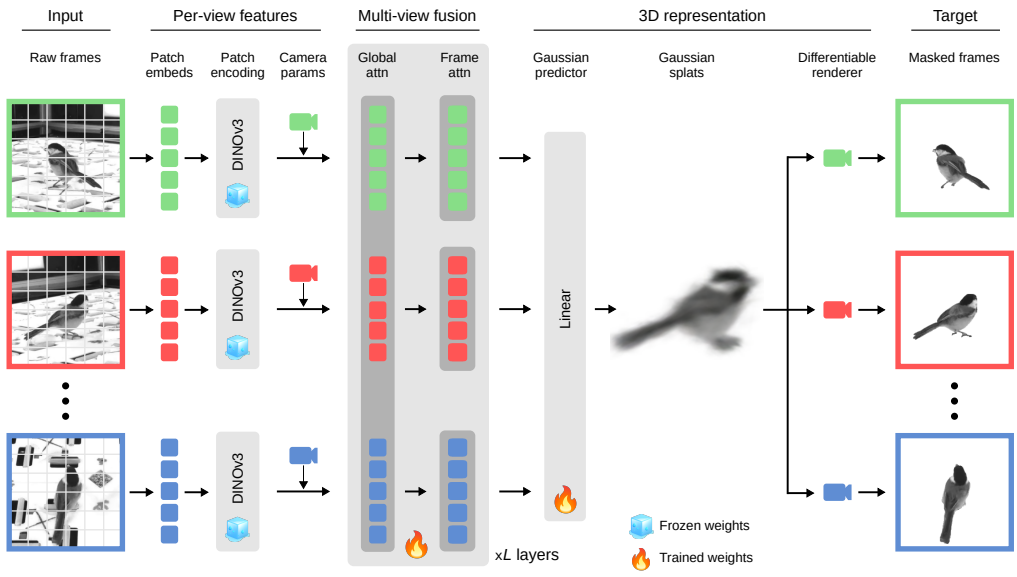

Figure 2: **BEAST3D framework.** BEAST3D is a masked autoencoder that uses 3D Gaussian splats as the intermediate representation. During training, one view is removed from the input and reconstructed through differentiable rendering of the 3D Gaussian splats inferred by the remaining views.

Self-supervised pretraining for behavior analysis. Selfee [40] constructs composite frames from grayscale video sequences and applies standard contrastive learning techniques, demonstrating effectiveness on downstream tasks like action segmentation and anomaly detection. Mueller et al. [41] and AnimalJEPA [42] adapt video-based joint-embedding predictive architectures to primate and mouse behavior classification, respectively. These methods produce trajectory- or clip-level representations suited to action recognition but do not target multi-camera data. Most closely related to our work, BEAST [43] combines masked autoencoding with temporal contrastive learning to pretrain a vision transformer on unlabeled videos from a single experimental setup, demonstrating improvements over general-purpose baselines on downstream neural encoding and pose estimation tasks. However, BEAST operates entirely in 2D image space and has no mechanism for reasoning about 3D geometry. BEAST3D extends this line of work by introducing an explicit 3D inductive bias: rather than reconstructing masked image patches, it reconstructs entire held-out viewpoints through differentiable 3D Gaussian splatting. This formulation forces the representation to encode the full 3D structure of the subject—its shape, articulation, and appearance from arbitrary viewpoints—rather than merely interpolating 2D texture patterns.

3 Method

We present BEAST3D, a self-supervised framework that learns 3D visual representations from multi-view animal behavior videos. Given a set of synchronized, calibrated camera views of a behaving animal, BEAST3D is trained to reconstruct held-out viewpoints via differentiable 3D Gaussian splatting. The resulting encoder captures rich geometric and appearance features that transfer effectively to downstream tasks. Fig. 2 provides an overview of our approach.

3.1 Problem Setting

V synchronized cameras observe a subject from different viewpoints. At each timestep we have a set of images $\{\mathbf{I}_v \in \mathbb{R}^{H\times W\times 3}\}_{v=1}^{V}$, the corresponding camera parameters, and foreground segmentation masks $\{\mathbf{M}_v \in \{0,1\}^{H\times W}\}_{v=1}^{V}$. Our goal is to learn an encoder that maps images to a 3D scene representation, with no keypoint annotations. At each training step, we randomly partition the V views into a reference set $\mathcal{R}$ and a target set $\mathcal{T}$, with $|\mathcal{T}| \geq 1$. The model observes only the reference views and camera parameters, predicts a 3D Gaussian representation of the scene, and renders images for all views. The training signal comes from reconstruction errors from the held-out target views.

3.2 Multi-view dataset construction

To construct training datasets we first uniformly sample one *scene* (synchronized frames across cameras) per second from each recording. To isolate the animal from the background, BEAST3D learns to predict per-pixel alpha values; by supervising this channel with "ground truth" segmentation masks during training, a separate segmentation model is not required at inference time. We generate these masks offline using SAM3 [44]: an initial bounding box per view is detected automatically from a simple text prompt ("mouse", "bird", etc.) and propagated across subsequent frames. Running this process on every new recording at inference time would be impractical; SAM3 is computationally expensive, and mask propagation can fail on challenging frames (e.g., severe occlusions), requiring manual correction (see Fig. 6 for examples of SAM3 successes and failures). We therefore run it once carefully during dataset construction, using a lightweight GUI to correct propagation failures, and amortize this cost into BEAST3D's trained weights.

3.3 Architecture

BEAST3D builds on the 3DGS framework of E-RayZer [21], but replaces the learned image tokenizer and camera pose predictor with a frozen pretrained vision encoder (as in VGGT) and ground truth calibration data, respectively (Fig. 2). These choices leverage the strong visual features of modern foundation models while exploiting the accurate camera geometry available in controlled lab settings.

Image tokenization. We use a frozen DINOv3 [25] ViT-B/16 as the image encoder. Each reference image $\mathbf{I}_v \in \mathbb{R}^{H\times W\times 3}$ is first normalized with ImageNet statistics and then passed through DINOv3 to obtain patch-level features $\mathbf{Z}_v \in \mathbb{R}^{N\times d}$, where N is the number of spatial tokens and $d = 768$ is the feature dimension. We retain the `[CLS]` and register tokens in the DINOv3 processing, allowing them to interact with the spatial tokens through self-attention; however, we discard the `[CLS]` and register tokens from the DINOv3 output, as these are not associated with any unique spatial patch, and would require a separate global-token pathway in the geometry transformer described below. All DINOv3 parameters remain frozen throughout training.

Camera tokenization. To provide BEAST3D with geometric grounding, each pixel is associated with the ray that emanates from the camera center through that pixel's location in 3D space. We represent each such ray in Plücker coordinates [37], a compact 6D descriptor that jointly encodes the ray's direction in world space and its offset from the origin, and is invariant to position along the ray. The resulting Plücker map $\mathbf{\Pi}_v \in \mathbb{R}^{H\times W\times 6}$ for view v is tokenized into patches using a linear projection to produce camera tokens $\mathbf{P}_v \in \mathbb{R}^{N\times d}$.

Token fusion. For each reference view $v \in \mathcal{R}$, we augment both image and camera tokens by adding positional embeddings (fixed 2D sinusoidal positional embeddings passed through a separate two-layer MLP for each modality). We concatenate these augmented tokens $\mathbf{Z}_v^{'}$ and $\mathbf{P}_v^{'}$ along the feature dimension and fuse them with another simple two-layer MLP.

Geometry transformer. The fused tokens from all reference views are collected into a single sequence $\mathbf{F} = [\mathbf{F}_{v_1}; \ldots; \mathbf{F}_{v_{|\mathcal{R}|}}] \in \mathbb{R}^{|\mathcal{R}|N\times d}$ and processed by a pretrained VGGT geometry transformer [20]. The transformer consists of L layers with self-attention alternating between two attention patterns: even layers apply frame attention, where each view attends only to its own tokens, while odd layers apply global attention across all views jointly. This alternating strategy allows the model to interleave 2D appearance processing with 3D multi-view aggregation.

3D Gaussian prediction. The output tokens from the geometry transformer are decoded into per-patch 3D Gaussian parameters via a linear head. Each spatial token predicts one 3D Gaussian $\mathbf{g}_i$, yielding $|\mathcal{R}| \cdot N$ Gaussians in total. Each Gaussian is described by 27 parameters that define position, shape and view-dependent color information (Appendix C).

Differentiable rendering. Given the predicted 3D Gaussians $\{\mathbf{g}_i\}_{i=1}^{|\mathcal{R}|N}$, we render images from both reference and target views using GSplat [45]. For each view, pixel colors are computed by accumulating Gaussian contributions along each viewing ray such that Gaussians closer to the camera contribute first, with each one partially occluding those behind it. We apply an additional "frustum constraint" that culls Gaussians that fall outside the intersection of all views.

3.4 Training

The training loss $\mathcal{L}$ is computed only on the held-out target views $\mathcal{T}$ and combines three terms: (1) a photometric loss $\mathcal{L}_{\ell_2}$ that penalizes pixel-level differences between the ground truth and rendered images; (2) a perceptual loss $\mathcal{L}_{\text{perc}}$ that penalizes abstract feature-level differences between ground truth and rendered images; and (3) a mask loss $\mathcal{L}_{\text{mask}}$ that penalizes differences between the rendered alpha channel and the SAM3 segmentation masks. The final loss is a weighted combination of each:

$$\mathcal{L} = \lambda_{\ell_2}\,\mathcal{L}_{\ell_2} + \lambda_{\text{perc}}\,\mathcal{L}_{\text{perc}} + \lambda_{\text{mask}}\,\mathcal{L}_{\text{mask}}. \tag{1}$$

In our experiments we set $\lambda_{\ell_2} = 1.0$, $\lambda_{\text{perc}} = 0.2$, and $\lambda_{\text{mask}} = 0.1$ across all datasets (Appendix C).

We train BEAST3D using the AdamW optimizer [46] with a base learning rate of 5×10^{-5}, weight decay of 0.05, and a cosine learning rate schedule with 15% warmup for 800 epochs. Training uses bf16 mixed precision and is distributed across 8 GPUs. During each forward pass, we randomly sample $V_{\text{ref}} = V - 1$ reference views and hold out $V_{\text{tgt}} = 1$ target view. The input image resolution is 256×256, producing $N = 256$ patch tokens per view.

4 Results

We first assess 3D reconstruction capabilities through a novel view synthesis task, which tests the models' ability to generate coherent 3D representations outside of the views it is prompted with. We then demonstrate the versatility of BEAST3D through two downstream neuro-behavioral tasks: pose estimation, a dense prediction task which assesses the model's ability to extract 3D keypoints; and neural encoding, which challenges the model to extract features that predict patterns in neural activity.

Datasets. We evaluate BEAST3D on four datasets spanning species, environments and camera configurations (Fig. 1; Appendix A): (1) a head-fixed mouse recorded from six views (Cheese3D; [18]); (2) a freely moving rat recorded from six views (Rat7M; [2]); (3) a seed-caching chickadee recorded from six views [3]; (4) a human performing everyday tasks recorded from four views (Human3.6M [19]). Together, these datasets span both unconstrained naturalistic behavior and controlled lab settings.

4.1 Novel view synthesis

Novel view synthesis (NVS) is the task of rendering a scene from a camera view that is withheld from the model at inference time (even though that same view may have been seen during training): for a given timestep, one or more views are held out and the model must reconstruct them from the remaining views alone. This provides a direct window into the quality of the model's internal 3D representation: a model that generates geometrically consistent, photorealistic renderings of held-out views must have learned to infer the 3D structure of the scene rather than merely memorizing 2D appearance statistics. We use NVS performance to benchmark BEAST3D's 3D reconstruction capabilities against competing methods before turning to more specialized downstream tasks.

Baselines. We compare against three baselines: E-RayZer (zero-shot), the pretrained E-RayZer [21] model applied directly without any fine-tuning; E-RayZer (fine-tuned), the same model fine-tuned on each of our training datasets; and Pose Splatter [22], a feed-forward Gaussian splat model specifically designed for multi-view animal behavior recordings. We note that Pose Splatter is extensively evaluated against per-scene optimization approaches—including 3DGS [31], FSGS [47], and GO [48]—and was found to perform comparably or better overall; we therefore omit those baselines here. We also omit VGGT [20], as it produces 3D point clouds but not the accompanying color, size, and opacity information for each point required to perform the NVS task.

Evaluation. E-RayZer (fine-tuned), Pose Splatter, and BEAST3D are each trained on data pooled across multiple individuals. We evaluate in two complementary regimes: (1) within-subject, in which test frames are drawn from sessions included in training, but the specific test frames are held out; this matches the evaluation protocol of Pose Splatter. (2) Cross-subject, in which test frames are drawn from sessions and individuals not seen during training, assessing generalization to novel subjects. In each setting we report three standard image quality metrics. Peak signal-to-noise ratio (PSNR) quantifies pixel-level reconstruction performance relative to the dynamic range of pixel intensities. Structural similarity index measure (SSIM) captures perceptual similarity of two images by comparing luminance, contrast, and structure [49]. Learned Perceptual Image Patch Similarity

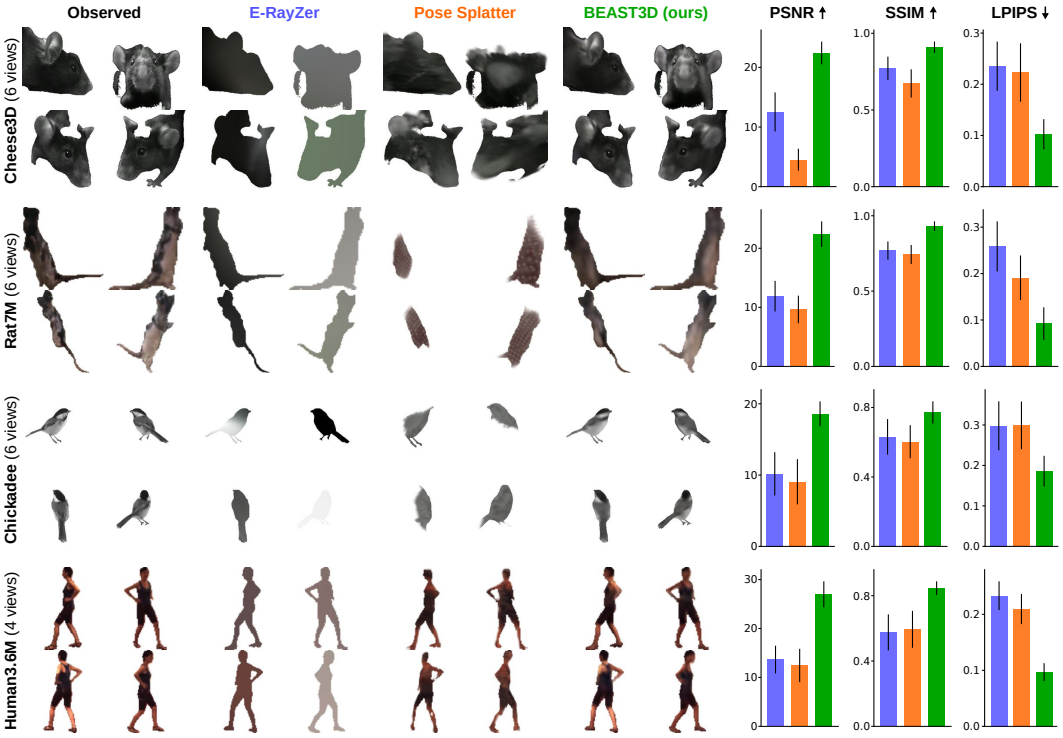

Figure 3: **BEAST3D performs high-fidelity novel view synthesis.** *Left*: example within-subject, held-out target views from each dataset and the corresponding reconstructions from E-RayZer, Pose Splatter, and BEAST3D, each conditioned on the remaining views from the same timestep. Reconstructions are masked by the SAM3 outputs; within these masked regions, E-RayZer often produces empty renderings, indicating that its predicted Gaussians fail to populate the subject's location. *Right*: per-dataset PSNR, SSIM, and LPIPS (described in text). We assess significance with two-sided Wilcoxon signed-rank tests on paired image metrics, Holm-corrected across the nine contrasts within each dataset (three pairwise model contrasts times three metrics). BEAST3D significantly outperforms both baselines across all metrics and datasets at $p < 10^{-10}$.

(LPIPS) measures perceptual dissimilarity between image patches using deep feature activations [50]. To enable fair comparisons across baselines, all metrics are computed over foreground pixels only, identified using SAM3 segmentation masks: E-RayZer reconstructs both foreground and background by default and must be masked accordingly, while Pose Splatter uses masks internally for shape carving. BEAST3D results evaluated without masks at inference time are reported in Appendix C.

Results. Qualitatively, BEAST3D produces sharp reconstructions of held-out viewpoints that recover the subject's silhouette and fine appearance details across all four datasets (Fig. 3; Supplementary Videos). With the ground truth masks applied, E-RayZer renderings clearly do not capture structure and appearance details of the subjects, while Pose Splatter outputs exhibit visible artifacts characteristic of shape-carving failures. Quantitatively, BEAST3D improves metrics across all datasets in the within-subject evaluation setting (Fig. 3; Tables 3, 5). Across-subjects, BEAST3D improves PSNR and LPIPS over the baselines on every dataset, and achieves the best SSIM on all but Chickadee, where E-RayZer is marginally better (Tables 4, 6). E-RayZer fine-tuning yields no improvement, and occasionally degrades performance, relative to zero-shot (Appendix); we attribute this to its joint camera-pose estimation branch failing in the sparse view regime tested here (Fig. 7).

Ablations (Tables 3–6). Following VGGT [20], which employs a frozen DINOv2 encoder, we use DINOv3 as our per-view image encoder. Removing DINOv3 degrades NVS performance across all datasets except for Human3.6M, demonstrating the multi-view transformer is not by itself sufficient to learn high-quality geometric representations, and benefits from strong pretrained visual features. Reinforcing this conclusion, the DINOv3 ablation *substantially* degrades pose estimation performance (Fig. 10), described in more detail in the next section. We also ablate the frustum constraint, which culls Gaussians not contained within the intersection of all viewpoints. This constraint has minimal effect on most datasets but is critical for Chickadee, where the bird is cropped from large images using a per-frame bounding box.

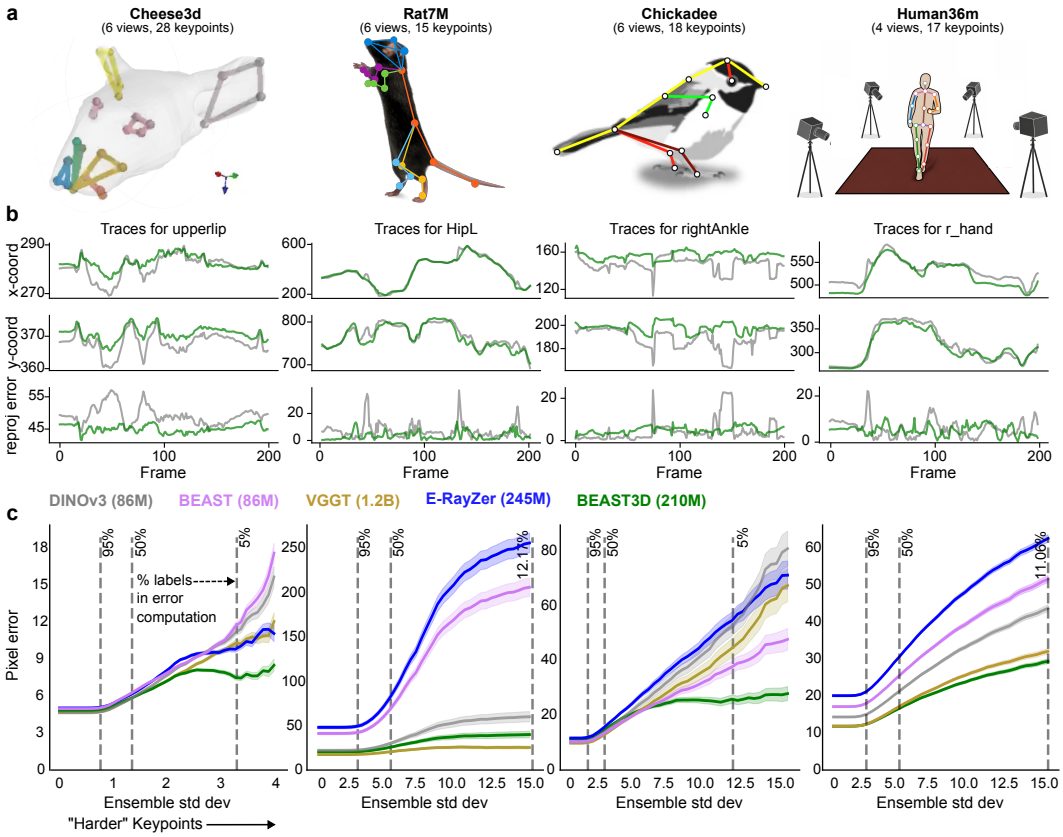

Figure 4: **BEAST3D improves pose estimation. a:** Experimental setups and keypoint skeletons for all datasets. **b:** *Top*: representative keypoint traces from a single view. *Bottom*: corresponding 3D reprojection error for ViT-B DINOv3 (gray) and BEAST3D (green). Because reprojection error leverages known camera geometry to measure agreement across views, it serves as a label-free proxy for prediction quality. Across all datasets, BEAST3D traces are visibly smoother in time and incur substantially lower reprojection error. **c:** Pixel error as a function of keypoint difficulty (lower is better), computed on frames from held-out test subjects [54]. Dashed vertical lines indicate the fraction of data retained; error bands show s.e.m. across included keypoints. Results are shown for all datasets trained with subsets of 100 labeled instances.

4.2 Pose Estimation

Pose estimation underpins the quantitative study of behavior in neuroscience and ethology [51, 52]. Although multi-view pose estimation is inherently 3D, most state-of-the-art pipelines process each view independently and triangulate 2D predictions into 3D as a separate post-processing step [53, 10]. In this work, we ask whether backbones that learn 3D structure during pretraining yield better pose estimators, particularly in the low-data regime typical of scientific applications. We evaluate on each dataset introduced in the novel view synthesis task.

Models. All pose estimators are trained with the Lightning Pose package [54, 14]. As a strong 2D baseline we use a ViT-B/16 backbone pretrained with DINOv3 [25]; against this baseline we evaluate four backbones: BEAST [43], a ViT-B/16 pretrained with masked autoencoding on single-view behavioral videos; VGGT [20]; E-RayZer [21]; and BEAST3D.

Evaluation. We adopt the limited-data evaluation protocol of the original BEAST paper [43], which is designed to reflect the annotation budgets of typical labs, where exhaustive labeling is impractical. For each backbone we train three models on random subsets of only 100 labeled instances and evaluate on held-out test subjects. Because pose estimation error varies substantially across keypoints—some are more inherently ambiguous or occluded—we report results using the difficulty-stratified error curves of ref. [54], which reveal how each method handles keypoints of varying difficulty rather than collapsing performance into a single number. "Difficulty" is quantified by the standard deviation of predictions across models and seeds for an individual keypoint on an individual frame (ensemble standard deviation, e.s.d.; higher e.s.d. indicates higher disagreement among models). Each error

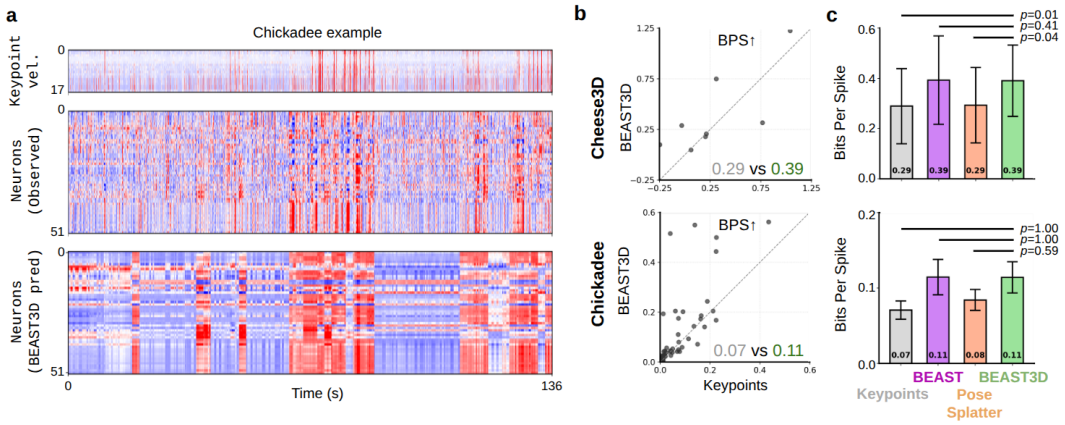

Figure 5: **BEAST3D features improve neural encoding. a**: Example session from Chickadee. *Top*: z-scored 3D keypoint velocities. *Middle*: observed neural activity. *Bottom*: activity predicted from BEAST3D Gaussian splats on held-out timepoints. **b**: Per-neuron BPS for BEAST3D vs. keypoints; each dot is a neuron. Session-averaged BPS shown in bottom-right. **c**: Average BPS across keypoints, BEAST, Pose Splatter, and BEAST3D, with error bars showing S.E.M. across neurons. We assess significance with two-sided Wilcoxon signed-rank tests on paired per-neuron ΔBPS, Holm-corrected across the three contrasts within each dataset (BEAST3D versus each of the baselines).

curve plots mean pixel error (y-axis) against an e.s.d. threshold (x-axis), where each point includes only keypoints whose e.s.d. exceeds that threshold. The leftmost point thus summarizes all keypoints, while moving rightward progressively restricts to more "difficult" ones.

Results. We find that BEAST3D achieves the best performance across nearly all datasets (Figs. 4, 11), with the exception of Rat7M where VGGT—a model nearly an order of magnitude larger—performs best. Notably, BEAST3D consistently outperforms BEAST across all datasets, and across a range of label budgets (Fig. 12), demonstrating the benefit of multi-view pretraining. However, a striking finding is that multi-view models more broadly (VGGT and E-RayZer) do not consistently outperform single-view DINOv3 and BEAST baselines, despite extensive fine-tuning and hyperparameter search (Appendix G). This suggests that a multi-view architecture alone is insufficient to guarantee strong pose estimation performance; how a model is pretrained matters at least as much as its architectural capacity for 3D reasoning. BEAST3D succeeds where other multi-view models fall short precisely because its novel view synthesis pretraining objective provides representations that transfer well to the pixel-level demands of pose estimation.

4.3 Neural encoding

A growing body of work probes the relationship between brain and behavior by predicting neural activity directly from behavior videos [55, 56, 57]. The standard approach relies on keypoints [58, 59, 60, 18], potentially missing critical behavioral features that are not well captured by tracking a sparse set of points. BEAST [57] addressed this limitation by showing that learned 2D image representations outperform keypoints for neural encoding; however, these representations take the form of a single high-dimensional `CLS` token per frame, offering no spatial structure that would allow neuroscientists to interpret which aspects of the animal's appearance or posture drive neural activity. BEAST3D's Gaussian splat representation is denser and more expressive than a sparse set of keypoints while remaining spatially grounded, making it more interpretable than opaque `CLS` tokens.

Datasets. We evaluate neural encoding on two datasets: Cheese3D, with eight neurons recorded from the mouse facial motor nucleus during spontaneous behavior [18]; and Chickadee, with 52 neurons recorded from hippocampus during seed-caching behavior [3]. See Appendix A for details.

Models. We use a variety of feature representations for neural encoding: (1) sparse 3D keypoints from pose estimation; (2) BEAST `CLS` tokens; (3) Pose Splatter Gaussian splats; and (4) BEAST3D Gaussian splats. For the Gaussian splat models we keep the centroid of each Gaussian and discard the remaining parameters that are used for rendering (e.g., opacity, color, and scale), which we find do not contain meaningful information for this encoding task (Table 8). For the keypoint- and splat-based representations we fit a PointNet transformer model [61] applied per timepoint, followed

by a spatiotemporal transformer that aggregates information over a fixed temporal window; for BEAST we fit a temporal convolution network following [43]. See Appendix H for details.

Evaluation. We split neural activity from each session into non-overlapping 2 s chunks, and randomly assign each chunk to a train, val or test split (70%/15%/15%). We tune model hyperparameters on the train/val splits, and report results on the test split using the Bits Per Spike (BPS) metric [62].

Results. BEAST3D features outperform 3D keypoints for predicting neural activity in both datasets (Fig. 5), and per-neuron comparisons confirm that this gain is consistent across the population rather than driven by a small subset of well-predicted neurons. This demonstrates that dense 3D representations capture behavioral information that sparse keypoints discard. BEAST3D also outperforms Pose Splatter, indicating that its splats capture more neurally-relevant behavioral structure. BEAST3D matches the performance of BEAST `CLS` tokens, recovering the predictive power of an opaque high-dimensional vector while preserving spatial structure. Because each Gaussian is anchored to a localized region of the subject, post-hoc analyses can ask which body parts or appearance details drive a given neuron's activity—a question that `CLS` tokens cannot answer. Notably, these trends hold across both datasets despite substantial differences in species (mouse vs. bird) and brain region (facial motor nucleus vs. hippocampus).

5 Conclusion

We presented BEAST3D, a self-supervised framework that learns dense 3D representations from multi-view animal behavior videos by reconstructing held-out viewpoints through differentiable Gaussian splatting. Across species and recording setups, we showed that BEAST3D's representations support strong 3D awareness through novel view synthesis; improve standard pose estimation pipelines in the low-annotation regime; and provide a spatially-grounded, interpretable representation for predicting neural activity. In each setting, BEAST3D outperforms general-purpose 3D computer vision models such as VGGT and E-RayZer, which are designed for the dense-view, uncalibrated regime and struggle in the sparse-view laboratory setting. BEAST3D also generalizes more reliably across subjects than Pose Splatter, a model tailored to multi-view animal recordings.

A current limitation of BEAST3D is its compute requirements: pretraining on a single dataset takes roughly 32 hours on 8 A100 GPUs, placing it out of reach for many labs (see Appendix F for details on inference compute). Identifying architectural choices that can reduce parameter count while preserving representation quality is an important avenue for future work. The framework naturally accommodates joint training across multiple datasets, which could yield a single backbone that labs use out-of-the-box without dataset-specific pretraining. In a first experiment, a single model trained jointly on three datasets matched the per-dataset models on held-out frames of training subjects but generalized worse to new subjects (Appendix E.1; Tables 3 and 4), so more careful training protocols will be needed to realize this direction.

Several extensions of BEAST3D suggest themselves as natural next steps. On the representation side, BEAST3D does not currently produce a canonical shape space: each timestep's Gaussian splats are predicted independently, with no explicit correspondence across frames or individuals. Designing a mechanism for a canonical body representation that is shared across frames—much as in the mesh-based literature [63, 15, 64, 65, 66]—while remaining species-agnostic and template-free, is a promising direction for future work. A related but more constrained extension would fit an articulated skeleton or mesh model to BEAST3D's dense per-frame observations, though this would require species-specific body templates [67, 17, 5]. On the application side, BEAST3D's fixed-coordinate geometry is naturally suited to egocentric alignment and depth-sensitive posture analysis, which raises the possibility of using BEAST3D embeddings for automated behavior discovery [68, 69, 70, 71, 22]; our datasets lack ground-truth ethograms, however, so we present this as a motivated direction for future evaluation. A further extension is to multi-animal social settings, where prior work has found 3D representations to be particularly informative for analyzing interactions [72, 5, 73]. Together, these directions promise to lower the barrier to adoption and bring self-supervised 3D representation learning into broader use across the behavioral neuroscience community.

Acknowledgments and Disclosure of Funding

We thank Qitao Zhao for discussions about the E-RayZer model, and Jack Goffinet for discussions about the Pose Splatter model. This work was supported by the following: Gatsby Charitable Foundation GAT3708, NIH 1R50NS145433, NIH U19NS123716, NSF 1707398, the National Science Foundation and DoD OUSD (R&E) under Cooperative Agreement PHY-2229929 (The NSF AI Institute for Artificial and Natural Intelligence), Simons Foundation, Wellcome Trust 216324, and funds from Zuckerman Institute Team Science. The authors declare no competing interests.

References

[1] Jesse D Marshall, Tianqing Li, Joshua H Wu, and Timothy W Dunn. Leaving flatland: Advances in 3d behavioral measurement. *Current Opinion in Neurobiology*, 73:102522, 2022.

[2] Jesse D Marshall, Diego E Aldarondo, Timothy W Dunn, William L Wang, Gordon J Berman, and Bence P Ölveczky. Continuous whole-body 3d kinematic recordings across the rodent behavioral repertoire. *Neuron*, 109(3):420–437, 2021.

[3] Selmaan N Chettih, Emily L Mackevicius, Stephanie Hale, and Dmitriy Aronov. Barcoding of episodic memories in the hippocampus of a food-caching bird. *Cell*, 187(8):1922–1935, 2024.

[4] Jonas Håkansson, Brooke L Quinn, Abigail L Shultz, Sharon M Swartz, and Aaron J Corcoran. Application of a novel deep learning–based 3d videography workflow to bat flight. *Annals of the new York Academy of Sciences*, 1536(1):92–106, 2024.

[5] Ugne Klibaite, Tianqing Li, Diego Aldarondo, Jumana F Akoad, Bence P Ölveczky, and Timothy W Dunn. Mapping the landscape of social behavior. *Cell*, 188(8):2249–2266, 2025.

[6] Emine Zeynep Ulutas, Amartya Pradhan, Dorothy Koveal, and Jeffrey E Markowitz. High-resolution in vivo kinematic tracking with customized injectable fluorescent nanoparticles. *Science Advances*, 11(40):eadu9136, 2025.

[7] Semih Günel, Helge Rhodin, Daniel Morales, João Campagnolo, Pavan Ramdya, and Pascal Fua. Deepfly3d, a deep learning-based approach for 3d limb and appendage tracking in tethered, adult drosophila. *Elife*, 8:e48571, 2019.

[8] Praneet C Bala, Benjamin R Eisenreich, Seng Bum Michael Yoo, Benjamin Y Hayden, Hyun Soo Park, and Jan Zimmermann. Automated markerless pose estimation in freely moving macaques with openmonkeystudio. *Nature communications*, 11(1):4560, 2020.

[9] Timothy W Dunn, Jesse D Marshall, Kyle S Severson, Diego E Aldarondo, David GC Hildebrand, Selmaan N Chettih, William L Wang, Amanda J Gellis, David E Carlson, Dmitriy Aronov, et al. Geometric deep learning enables 3d kinematic profiling across species and environments. *Nature methods*, 18(5):564–573, 2021.

[10] Pierre Karashchuk, Katie L Rupp, Evyn S Dickinson, Sarah Walling-Bell, Elischa Sanders, Eiman Azim, Bingni W Brunton, and John C Tuthill. Anipose: A toolkit for robust markerless 3d pose estimation. *Cell reports*, 36(13), 2021.

[11] Arne Monsees, Kay-Michael Voit, Damian J Wallace, Juergen Sawinski, Edyta Charyasz, Klaus Scheffler, Jakob H Macke, and Jason ND Kerr. Estimation of skeletal kinematics in freely moving rodents. *Nature methods*, 19(11):1500–1509, 2022.

[12] Yaning Han, Ke Chen, Yunke Wang, Wenhao Liu, Zhouwei Wang, Xiaojing Wang, Chuanliang Han, Jiahui Liao, Kang Huang, Shengyuan Cai, et al. Multi-animal 3d social pose estimation, identification and behaviour embedding with a few-shot learning framework. *Nature machine intelligence*, 6(1):48–61, 2024.

[13] Chaoqun Cheng, Zijian Huang, Ruiming Zhang, Guozheng Huang, Han Wang, Likai Tang, and Xiaoqin Wang. A real-time, multi-subject three-dimensional pose tracking system for the behavioral analysis of non-human primates. *Cell Reports Methods*, 5(2), 2025.

[14] Lenny Aharon, Matthew R Whiteway, Karan Sikka, Keemin Lee, Yanchen Wang, Selmaan Chettih, Benjamin Midler, Ilana B Witten, Dmitriy Aronov, International Brain Laboratory, et al. Lightning pose 3d: an uncertainty-aware framework for data-efficient multi-view animal pose estimation. *bioRxiv*, pages 2026–04, 2026.

[15] Silvia Zuffi, Angjoo Kanazawa, David W Jacobs, and Michael J Black. 3d menagerie: Modeling the 3d shape and pose of animals. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6365–6373, 2017.

[16] Marc Badger, Yufu Wang, Adarsh Modh, Ammon Perkes, Nikos Kolotouros, Bernd G Pfrommer, Marc F Schmidt, and Kostas Daniilidis. 3d bird reconstruction: a dataset, model, and shape recovery from a single view. In *European conference on computer vision*, pages 1–17. Springer, 2020.

[17] James P Bohnslav, Mohammed Abdal Monium Osman, Akshay Jaggi, Sofia Soares, Caleb Weinreb, Sandeep Robert Datta, and Christopher D Harvey. Armo: An articulated mesh approach for mouse 3d reconstruction. *bioRxiv*, pages 2023–02, 2023.

[18] Kyle Daruwalla, Irene Nozal Martin, Linghua Zhang, Diana Naglič, Andrew Frankel, Catherine Rasgaitis, Rubin Zhao, Xinyan Zhang, Zainab Ahmad, Jeremy C Borniger, et al. Cheese3d enables sensitive detection and analysis of whole-face movement in mice. *Nature Neuroscience*, pages 1–12, 2026.

[19] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE transactions on pattern analysis and machine intelligence*, 36(7):1325–1339, 2013.

[20] Jianyuan Wang, Minghao Chen, Nikita Karaev, Andrea Vedaldi, Christian Rupprecht, and David Novotny. Vggt: Visual geometry grounded transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 5294–5306, 2025.

[21] Qitao Zhao, Hao Tan, Qianqian Wang, Sai Bi, Kai Zhang, Kalyan Sunkavalli, Shubham Tulsiani, and Hanwen Jiang. E-rayzer: Self-supervised 3d reconstruction as spatial visual pre-training. *arXiv preprint arXiv:2512.10950*, 2025.

[22] Jack Goffinet, Youngjo Min, Carlo Tomasi, and David Carlson. Pose splatter: A 3d gaussian splatting model for quantifying animal pose and appearance. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[23] Chen Li and Gim Hee Lee. Coarse-to-fine animal pose and shape estimation. *Advances in Neural Information Processing Systems*, 34:11757–11768, 2021.

[24] Xuyi Hu, Jin Lyu, Jiuming Liu, Yebin Liu, Silvia Zuffi, Liang An, and Stefan Goetz. Sam 3d animal: Promptable animal 3d reconstruction from images in the wild. *arXiv preprint arXiv:2605.07604*, 2026.

[25] Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

[26] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[27] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[28] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15619–15629, 2023.

[29] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021.

[30] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[31] 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.

[32] 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.

[33] David Charatan, Sizhe Lester Li, Andrea Tagliasacchi, and Vincent Sitzmann. pixelsplat: 3d gaussian splats from image pairs for scalable generalizable 3d reconstruction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 19457–19467, 2024.

[34] Yuedong Chen, Haofei Xu, Chuanxia Zheng, Bohan Zhuang, Marc Pollefeys, Andreas Geiger, Tat-Jen Cham, and Jianfei Cai. Mvsplat: Efficient 3d gaussian splatting from sparse multi-view images. In *European conference on computer vision*, pages 370–386. Springer, 2024.

[35] Kai Zhang, Sai Bi, Hao Tan, Yuanbo Xiangli, Nanxuan Zhao, Kalyan Sunkavalli, and Zexiang Xu. Gs-lrm: Large reconstruction model for 3d gaussian splatting. In *European Conference on Computer Vision*, pages 1–19. Springer, 2024.

[36] Hanwen Jiang, Hao Tan, Peng Wang, Haian Jin, Yue Zhao, Sai Bi, Kai Zhang, Fujun Luan, Kalyan Sunkavalli, Qixing Huang, et al. Rayzer: A self-supervised large view synthesis model. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4918–4929, 2025.

[37] Julius Plücker. Xvii. on a new geometry of space. *Philosophical Transactions of the Royal Society of London*, (155):725–791, 1865.

[38] Tinghui Zhou, Richard Tucker, John Flynn, Graham Fyffe, and Noah Snavely. Stereo magnification: Learning view synthesis using multiplane images. *arXiv preprint arXiv:1805.09817*, 2018.

[39] Lu Ling, Yichen Sheng, Zhi Tu, Wentian Zhao, Cheng Xin, Kun Wan, Lantao Yu, Qianyu Guo, Zixun Yu, Yawen Lu, et al. Dl3dv-10k: A large-scale scene dataset for deep learning-based 3d vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 22160–22169, 2024.

[40] Yinjun Jia, Shuaishuai Li, Xuan Guo, Bo Lei, Junqiang Hu, Xiao-Hong Xu, and Wei Zhang. Selfee, self-supervised features extraction of animal behaviors. *Elife*, 11:e76218, 2022.

[41] Felix B Mueller, Timo Lueddecke, Richard Vogg, and Alexander S Ecker. Domain-adaptive pretraining improves primate behavior recognition. *arXiv preprint arXiv:2509.12193*, 2025.

[42] Chengjie Zheng, Tewodros Mulugeta Dagnew, Liuyue Yang, Wei Ding, Shiqian Shen, Changning Wang, and Ping Chen. Animal-jepa: Advancing animal behavior studies through joint embedding predictive architecture in video analysis. In *2024 IEEE International Conference on Big Data (BigData)*, pages 1909–1918. IEEE, 2024.

[43] Yanchen Wang, Han Yu, Ari Blau, Yizi Zhang, Liam Paninski, Cole Lincoln Hurwitz, Matthew R Whiteway, et al. Animal behavioral analysis and neural encoding with transformer-based self-supervised pretraining. In *The Fourteenth International Conference on Learning Representations*, 2026.

[44] Nicolas Carion, Laura Gustafson, Yuan-Ting Hu, Shoubhik Debnath, Ronghang Hu, Didac Suris, Chaitanya Ryali, Kalyan Vasudev Alwala, Haitham Khedr, Andrew Huang, et al. Sam 3: Segment anything with concepts. *arXiv preprint arXiv:2511.16719*, 2025.

[45] Vickie Ye, Ruilong Li, Justin Kerr, Matias Turkulainen, Brent Yi, Zhuoyang Pan, Otto Seiskari, Jianbo Ye, Jeffrey Hu, Matthew Tancik, et al. gsplat: An open-source library for gaussian splatting. *Journal of Machine Learning Research*, 26(34):1–17, 2025.

[46] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

[47] Zehao Zhu, Zhiwen Fan, Yifan Jiang, and Zhangyang Wang. Fsgs: Real-time few-shot view synthesis using gaussian splatting. In *European conference on computer vision*, pages 145–163. Springer, 2024.

[48] Chen Yang, Sikuang Li, Jiemin Fang, Ruofan Liang, Lingxi Xie, Xiaopeng Zhang, Wei Shen, and Qi Tian. Gaussianobject: High-quality 3d object reconstruction from four views with gaussian splatting. *arXiv preprint arXiv:2402.10259*, 2024.

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

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

[51] Sandeep Robert Datta, David J Anderson, Kristin Branson, Pietro Perona, and Andrew Leifer. Computational neuroethology: a call to action. *Neuron*, 104(1):11–24, 2019.

[52] Talmo D Pereira, Joshua W Shaevitz, and Mala Murthy. Quantifying behavior to understand the brain. *Nature neuroscience*, 23(12):1537–1549, 2020.

[53] Tanmay Nath, Alexander Mathis, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie Weygandt Mathis. Using deeplabcut for 3d markerless pose estimation across species and behaviors. *Nature protocols*, 14(7):2152–2176, 2019.

[54] Dan Biderman, Matthew R Whiteway, Cole Hurwitz, Nicholas Greenspan, Robert S Lee, Ankit Vishnubhotla, Richard Warren, Federico Pedraja, Dillon Noone, Michael M Schartner, et al. Lightning pose: improved animal pose estimation via semi-supervised learning, bayesian ensembling and cloud-native open-source tools. *Nature methods*, 21(7):1316–1328, 2024.

[55] Simon Musall, Matthew T Kaufman, Ashley L Juavinett, Steven Gluf, and Anne K Churchland. Single-trial neural dynamics are dominated by richly varied movements. *Nature neuroscience*, 22(10):1677–1686, 2019.

[56] Carsen Stringer, Marius Pachitariu, Nicholas Steinmetz, Charu Bai Reddy, Matteo Carandini, and Kenneth D Harris. Spontaneous behaviors drive multidimensional, brainwide activity. *Science*, 364(6437):eaav7893, 2019.

[57] Ziyue Aiden Wang, Balint Kurgyis, Susu Chen, Byungwoo Kang, Feng Chen, Yi Liu, Dave Liu, Karel Svoboda, Nuo Li, and Shaul Druckmann. Brain-wide analysis reveals movement encoding structured across and within brain areas. *Nature Neuroscience*, 29(1):147–158, 2026.

[58] Atika Syeda, Lin Zhong, Renee Tung, Will Long, Marius Pachitariu, and Carsen Stringer. Facemap: a framework for modeling neural activity based on orofacial tracking. *Nature neuroscience*, 27(1):187–195, 2024.

[59] International Brain Laboratory, Kush Banga, Julius Benson, Jai Bhagat, Dan Biderman, Daniel Birman, Niccolò Bonacchi, Sebastian A Bruijns, Kelly Buchanan, Robert AA Campbell, et al. Reproducibility of in vivo electrophysiological measurements in mice. *Elife*, 13:RP100840, 2025.

[60] International Brain Laboratory, Dora Angelaki, Brandon Benson, Julius Benson, Daniel Birman, Niccolò Bonacchi, Kcénia Bougrova, Sebastian A Bruijns, Matteo Carandini, Joana A Catarino, et al. A brain-wide map of neural activity during complex behaviour. *Nature*, 645(8079):177–191, 2025.

[61] 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*, pages 652–660, 2017.

[62] Felix Pei, Joel Ye, David Zoltowski, Anqi Wu, Raeed H Chowdhury, Hansem Sohn, Joseph E O'Doherty, Krishna V Shenoy, Matthew T Kaufman, Mark Churchland, et al. Neural latents benchmark'21: evaluating latent variable models of neural population activity. *arXiv preprint arXiv:2109.04463*, 2021.

[63] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *European Conference on Computer Vision*, pages 386–402. Springer, 2018.

[64] Shangzhe Wu, Ruining Li, Tomas Jakab, Christian Rupprecht, and Andrea Vedaldi. Magicpony: Learning articulated 3d animals in the wild. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 8792–8802. IEEE, 2023.

[65] Zizhang Li, Dor Litvak, Ruining Li, Yunzhi Zhang, Tomas Jakab, Christian Rupprecht, Shangzhe Wu, Andrea Vedaldi, and Jiajun Wu. Learning the 3d fauna of the web. In *2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9752–9762. IEEE, 2024.

[66] Yuan Zhong, Jingxiang Sun, Zhongbin Zhang, Liang An, and Yebin Liu. Moremouse: Monocular reconstruction of laboratory mouse. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pages 13548–13556, 2026.

[67] Silvia Zuffi, Angjoo Kanazawa, Tanya Berger-Wolf, and Michael Black. Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild". In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5358–5367. IEEE, 2019.

[68] Alexander B Wiltschko, Matthew J Johnson, Giuliano Iurilli, Ralph E Peterson, Jesse M Katon, Stan L Pashkovski, Victoria E Abraira, Ryan P Adams, and Sandeep Robert Datta. Mapping sub-second structure in mouse behavior. *Neuron*, 88(6):1121–1135, 2015.

[69] Alexander I Hsu and Eric A Yttri. B-soid, an open-source unsupervised algorithm for identification and fast prediction of behaviors. *Nature communications*, 12(1):5188, 2021.

[70] Kevin Luxem, Petra Mocellin, Falko Fuhrmann, Johannes Kürsch, Stephanie R Miller, Jorge J Palop, Stefan Remy, and Pavol Bauer. Identifying behavioral structure from deep variational embeddings of animal motion. *Communications Biology*, 5(1):1267, 2022.

[71] Caleb Weinreb, Jonah E Pearl, Sherry Lin, Mohammed Abdal Monium Osman, Libby Zhang, Sidharth Annapragada, Eli Conlin, Red Hoffmann, Sofia Makowska, Winthrop F Gillis, et al. Keypoint-moseq: parsing behavior by linking point tracking to pose dynamics. *Nature methods*, 21(7):1329–1339, 2024.

[72] Jathushan Rajasegaran, Georgios Pavlakos, Angjoo Kanazawa, Christoph Feichtenhofer, and Jitendra Malik. On the benefits of 3d pose and tracking for human action recognition. In *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 640–649. IEEE, 2023.

[73] Lucas Maximilian Martini, Alex Lappe, Anna Bognár, Rufin Vogels, and Martin Giese. Bigmaq: a big macaque motion and animation dataset bridging image and 3d pose representations. In *International Conference on Learning Representations*, volume 2026, pages 117144–117175, 2026.

[74] Catalin Ionescu, Fuxin Li, and Cristian Sminchisescu. Latent structured models for human pose estimation. In *2011 International Conference on Computer Vision*, pages 2220–2227. IEEE, 2011.

[75] Alex Henry, Prudhvi Raj Dachapally, Shubham Shantaram Pawar, and Yuxuan Chen. Query-key normalization for transformers. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4246–4253, 2020.

[76] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *European conference on computer vision*, pages 694–711. Springer, 2016.

[77] Yufei Xu, Jing Zhang, Qiming Zhang, and Dacheng Tao. Vitpose: Simple vision transformer baselines for human pose estimation. *Advances in neural information processing systems*, 35:38571–38584, 2022.

[78] Alexander Mathis, Pranav Mamidanna, Kevin M Cury, Taiga Abe, Venkatesh N Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. Deeplabcut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience*, 21(9):1281–1289, 2018.

Supplementary Material

BEAST3D: Animal behavioral analysis and neural encoding from multi-view video via Gaussian splatting

Contents

A Datasets 16

B Dataset construction 18

C BEAST3D 20

D Baselines 21

E Novel view synthesis 24

F Inference compute cost 28

G Pose estimation 28

H Neural encoding 31

I Broader impacts 32

A Datasets

For all datasets we define a set of "In-Distribution" (InD) sessions used for training and a separate set of "Out-of-Distribution" (OOD) sessions comprising *new subjects* for evaluation. Table 1 documents the number of sessions, subjects and BEAST3D training/evaluation frames for each dataset; Appendix B details how these frames are selected. For the pose estimation task, training frames are drawn exclusively from InD sessions and test frames exclusively from OOD sessions.

A.1 Cheese3D

Behavior data. A head-fixed mouse behaves spontaneously, captured by six cameras at 100 Hz [18]. Frame sizes are 640 × 512 pixels.

For the pose estimation task, we did not have the required labeled dataset where each keypoint is labeled across all views for a given instance in time. We instead constructed a pseudo-labeled dataset from an initial set of 665 instances where each keypoint is labeled in a subset of views depending on anatomical visibility. Using this data, we trained an ensemble of three single-view transformer pose estimation models [14] using a DINOv2-pretrained ViT-B backbone (Fig. 9; step 2).

Ensemble inference. We ran inference with the ensemble across a set of full-length videos. For each time point, view and keypoint, we computed the median across the ensemble as our 2D prediction. Because training labels covered only a subset of views per keypoint, predictions in unlabeled views were systematically unreliable and received low likelihoods.

Frame filtering and triangulation. We applied a confidence filter requiring that at least two views report a median likelihood exceeding 0.6 for a given keypoint to contribute to triangulation. Keypoints for which no view ever exceeded this threshold within a given recording session—arising from anatomical occlusion or a complete absence of training annotations across all views—were excluded from the per-frame acceptance criterion and appear as `NaN` in the final pseudo-labeled output. A frame was retained if all eligible keypoints were confidently predicted in at least two views simultaneously. For retained frames, we performed multi-view triangulation using

camera parameters to recover 3D keypoint locations. These 3D estimates were then reprojected into all six views, producing geometrically consistent 2D pseudo-labels even in cameras that lacked direct confident predictions.

Frame selection. To obtain a compact yet pose-diverse pseudo-labeled set, we applied k-means clustering in the space of flattened 3D keypoints for each session. The frame closest to each cluster centroid was selected as the representative. We targeted 50–55 representative frames per session.

Reprojection error quality filtering. As a final quality control step, we computed a per-frame reprojection error as the mean Euclidean distance between the original ensemble median 2D predictions and the triangulation-derived reprojected labels, averaged over all confident keypoints and views. We discarded the worst 25% of frames by this metric. Pseudo-labels are provided for all six views per retained frame, with `NaN` entries where triangulation was not possible. The final pose estimation training/test sets consist of 450/150 instances, respectively.

Neural data. A 32-channel single-shank silicon probe was inserted into the facial motor nucleus. The probe was coated with lipophilic dyes to reveal the probe track post hoc. Recordings began at least 15 min after probe insertion to ensure recording stability. Voltage signals were acquired at 30 kHz. After the recording, single electrical pulses were delivered to all sites on the probe to induce facial movements to verify probe placement location. For this specific recording session, ear, whisker pad, nose and mouth movements were observed following electric stimulation.

Spikes were sorted into 8 well-isolated single units and binned at the downsampled video frame rate (50 Hz). The full $\sim$20 min session (61,697 frames) was segmented into non-overlapping 2.0 s windows of 100 frames each, yielding 616 windows. Windows were shuffled with a fixed seed and partitioned into 70% / 15% / 15% train / val / test splits (431 / 92 / 93 windows). All 8 units were retained without firing-rate thresholding.

A.2 Rat7M

A single freely moving rat behaves spontaneously in a circular arena, captured by six cameras at 120 Hz [2]. Frame sizes are 1328 $\times$ 1048 pixels.

Rat7M is a marker-based motion capture dataset with a large number of ground truth pose labels spanning multiple subjects. We retain 15 of the original keypoints in each view, omitting those tracking the head stage of the rat. Rather than retain the 30,000+ labeled instances in the original dataset, we chose to curate a smaller subset of instances matched in size with our other datasets. For each session we first filter out any instances with missing data. We then filter out potentially problematic points using skeleton distances (`ElbowR-ElbowL` distance in $[40, 60]$; `KneeR-ShinR` and `KneeL-ShinL` distances in $[10, 1000]$). We then run k-means clustering on the remaining 3D poses (using 100 clusters per session) and select one example per cluster. Finally, we performed manual inspection of the resulting labels and excluded any instances where ground truth keypoints were incorrect due to camera syncing issues. The pose estimation training/test sets consist of 455/177 instances, respectively.

The dataset is available at `https://doi.org/10.6084/m9.figshare.c.5295370` under the CC-BY 4.0 license.

A.3 Chickadee

Behavior data. Freely moving chickadees engage in seed caching behavior in a large arena, captured by six cameras at 60 Hz [3]. Frame sizes vary by view but are approximately 3000 $\times$ 1500 pixels. Given the small size of the bird relative to the arena, we produced a set of cropped videos for model training. Using previously collected pose estimation labels, we trained a detector network on full resolution frames downsampled to 256 $\times$ 256 pixels to localize the bird within each frame. We computed a bounding box from the pose estimation output, and use these cropped videos for training all downstream models.

For the pose estimation task, we created a cropped dataset using the ground truth labels to define a bounding box around the bird, and reshaped the cropped frames to 320 $\times$ 320 pixels. Eighteen keypoints on the chickadee's body are labeled in each view. The training/test sets consist of 433/143 instances, respectively.

Neural data. Large-scale silicon-probe recordings of one chickadee yielded 132 spike-sorted units across a multi-hour free-behavior session [3]. Spikes were binned at the video frame rate (60 Hz). To focus the encoding analysis on a behaviorally rich and neurally active window, we performed a sliding-window search over the full recording for the contiguous 15-minute interval that maximized the number of units firing at $\geq$1 Hz; this interval (54,000 frames at 60 Hz, in which 53 of 132 units exceed 1 Hz mean firing rate) is used for all subsequent encoding analyses. After cutting the videos and spike trains to this range, we segmented the spike trains into non-overlapping 2.0 s windows of 120 frames each (450 windows total), shuffled with a fixed seed, and split 70% / 15% / 15% into train / val / test (315 / 67 / 68 windows). For neural encoding, we additionally drop units whose mean spike count per training window is below 2 (i.e. $<$1 Hz), retaining 52 units; this filter is computed on training windows only and applied to all three splits to avoid data leakage.

Table 1: Dataset statistics for In-Distribution (InD) and Out-of-Distribution (OOD) splits.

	In-Distribution				Out-of-Distribution		
Dataset	Subjects	Sessions	Train frames	Test frames	Subjects	Sessions	Test frames
Cheese3D [18]	6	11	44370	8382	2	4	13890
Rat7M [2]	3	5	53970	7650	2	2	21588
Chickadee [3]	6	12	64728	7650	2	4	21558
Human3.6M [19]	5	75	78596	15472	2	30	27660

A.4 Human3.6M

Human subjects perform a range of everyday activities captured by four synchronized cameras at 50 Hz [74, 19]. Frame sizes are 1000×1002 pixels.

Human3.6M is a marker-based motion capture dataset with ground truth pose labels spanning multiple subjects and 15 activity types (actions 2–16). We retain 17 keypoints in each view. Rather than retain the 3.6M labeled instances in the original dataset, similar to Rat7M we chose to curate a smaller subset of instances to benchmark the pose estimation models. Only the first subaction of each action is used; subaction variants and action 13 of subject 9 are excluded due to corrupted 3D coordinates. For each subject-action pair, we first filter out any instances with missing 3D keypoint data. We then run k-means clustering on the remaining 3D poses using 15 clusters and select the single instance closest to each cluster center, yielding up to 15 representative frames per session. We additionally require that selected frame indices are separated by at least 3 frames. The pose estimation training/test sets consist of 1125/425 instances, respectively.

The dataset is available at `http://vision.imar.ro/human3.6m` under the Human3.6M license.

B Dataset construction

All datasets are processed by a single, configurable pipeline that turns raw multi-view recordings into the calibrated *(frame, mask, camera)* tuples consumed by BEAST3D. The pipeline runs five stages in order: *cut*, *downsample*, *segment*, *assemble* and *resize*. Each stage writes its output to disk and the next stage picks it up automatically, so individual steps can be re-run without redoing earlier work. The same pipeline is invoked three times per dataset, with three different configurations, to produce the pretraining set, the in-distribution (InD) test set, and the out-of-distribution (OOD) test set.

B.1 Processing pipeline

Cut. Many of the source recordings are very long (tens of minutes to several hours) and captured at 50–120 Hz, which is far denser than what is needed for self-supervised pretraining of a static-frame model. The optional *cut* step trims each video to a user-specified inclusive frame range using `ffmpeg`'s frame-accurate `trim` filter, so that downstream steps see only the portion of the recording that contains the behavior of interest. Whenever bounding-box CSVs accompany the videos, they are filtered and re-indexed to the same range so that bbox-frame alignment is preserved.

Downsample. The *downsample* step then resamples each video to a target frame rate via `ffmpeg -vf fps=...` and re-encodes with libx264 (CRF 18, audio dropped). Both *cut* and *downsample* steps process video files in parallel through a process pool to amortize cluster I/O.

Segment. For each downsampled video we run SAM3 [44] in tracking mode with a dataset-specific text prompt (e.g. `mouse` for Cheese3D, `bird` for Chickadee) to obtain a per-frame binary foreground mask. When SAM3's text grounding fails for a particular video—usually because the subject is small, occluded in the first frame, or visually atypical—the failure is recorded in `failed_videos.json`. We developed a small Gradio app to manually click two corners of a bounding box on the first frame of each failed video; a follow-up *retry* stage then re-runs SAM3 on those videos with the labeled boxes supplied, which bypasses text grounding and resolves all remaining failures.

Figure 6 shows representative SAM3 masks for all four datasets. In the large majority of frames SAM3 produces a single connected mask that tightly follows the subject's silhouette. The typical failure mode involves occlusion: the mask is truncated or split into fragments where the subject touches or is partially hidden by scene objects (feeders, perches, cables, a chair), or it leaks onto the occluding object itself. We also observe that masks are qualitatively more reliable on Human3.6M than on the animal datasets (mouse, rat, and bird), likely because human imagery is far better represented than laboratory animal footage in the training data of the segmentation foundation model.

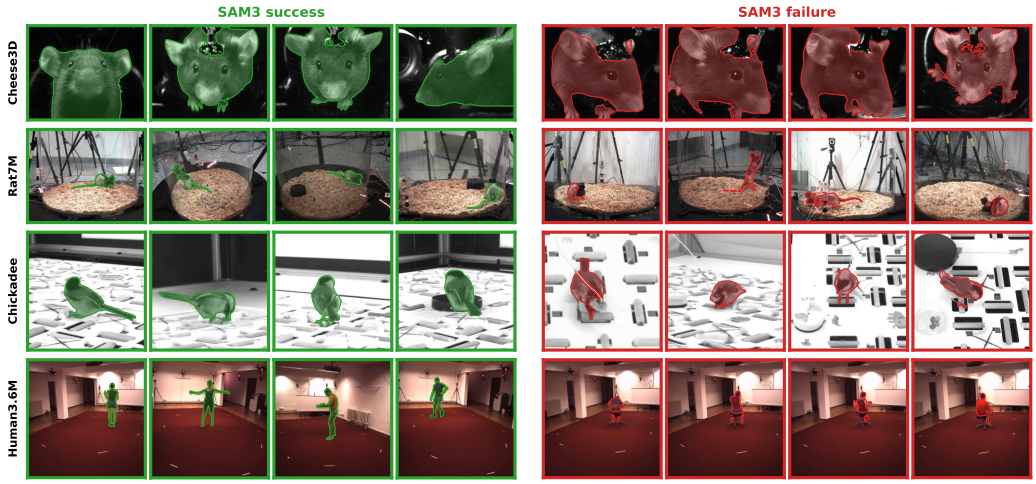

Figure 6: **SAM3 mask quality.** Representative SAM3 foreground masks (colored overlay and contour) for each dataset: successes (*green*, left) and failures (*red*, right). Failures are mostly truncated, fragmented, or leaking masks caused by occlusion or contact with scene objects (feeders, cables, a chair).

Assemble. The *assemble* stage groups videos by session and camera, loads the corresponding intrinsics and extrinsics from the calibration file (Anipose `toml` or DeepLabCut `pickle`, depending on the dataset), and writes a single self-contained directory per session of the form `<session>/<cam>/img{idx}.png` together with a corresponding `img{idx}.npy` that holds the per-frame camera parameters (and bounding box, when available). The matching SAM3 mask is copied alongside as `mask{idx}.png`.

Resize. A final *resize* stage rescales the assembled images and masks so that the shorter side is 256 pixels, matching the input resolution expected by the DINOv3 ViT-B/16 backbone used in BEAST3D (Appendix C). Calibration intrinsics are scaled by the same factor.

B.2 Dataset splits

Pretraining set. The pretraining configuration uses the InD sessions of each dataset and aggressively downsamples each video to a low effective frame rate—1 Hz for Cheese3D and Chickadee, and a comparable rate for Rat7M and Human3.6M—under the assumption that consecutive seconds are visually near-redundant for the purpose of single-frame multi-view geometry learning. After the cut, downsample, segment, assemble, and resize stages, every retained frame becomes a training example with up to six paired views, foreground masks, and calibrated cameras. This procedure produces the training-frame counts reported in Table 1.

OOD test set. The OOD test set is constructed by re-running the exact same pipeline on a held-out set of sessions from *new subjects* that never appear during pretraining. Apart from pointing at a different input directory, the configuration is identical to the pretraining one—same target frame rate, same SAM3 prompt, same calibration loader, same resize—so the resulting frames are statistically comparable to the training frames in resolution, masking quality, and temporal density. This isolates the effect of subject identity on novel view synthesis and pose estimation performance.

InD test set. The InD test set is more delicate: it must come from the same sessions as pretraining, yet contain frames that are guaranteed to be unseen during training. We achieve this by re-running the downsample stage on the same InD videos with a *phase-shifted* frame selector. Concretely, if the source video is captured at f_{src} Hz and the pretraining target is f_{tgt} Hz, the pretraining pipeline keeps source frames at indices $\{0, S, 2S, \ldots\}$ with stride $S = \text{round}(f_{\text{src}}/f_{\text{tgt}})$. The InD test pipeline shifts this lattice by an offset K with $1 \leq K < S$, keeping source frames $\{K, K+S, K+2S, \ldots\}$. We use $K = S/2$ (e.g. $K = 50$ for Cheese3D's $f_{\text{src}} = 100$ Hz, $f_{\text{tgt}} = 1$ Hz pretraining setup), which places each test frame roughly half a sampling interval away from the nearest training frame. Because $1 \leq K < S$, the training and InD test frame sets are guaranteed to be disjoint at the source-frame level, while still being drawn from the same underlying distribution of behavior, lighting, and subjects.

C BEAST3D

For each timestep, for a dataset with V views, we have access to a set of images $\{\mathbf{I}_v \in \mathbb{R}^{H\times W\times 3}\}_{v=1}^{V}$, the corresponding camera-to-world transformations $\{\mathbf{T}_v \in \mathrm{SE}(3)\}_{v=1}^{V}$ and intrinsic parameters $\{f_x^v, f_y^v, c_x^v, c_y^v\}_{v=1}^{V}$, and foreground segmentation masks $\{\mathbf{M}_v \in \{0,1\}^{H\times W}\}_{v=1}^{V}$ via SAM3.

At each training step, we randomly partition the V views into a reference set $\mathcal{R}$ and a target set $\mathcal{T}$, with $|\mathcal{T}| \geq 1$. Our goal is to learn an encoder that maps reference views and their camera parameters to a 3D scene representation which is capable of reconstructing the target views.

C.1 Architecture details

Image tokenization. We use a frozen DINOv3 [25] ViT-B/16 as the image encoder. Each reference image $\mathbf{I}_v \in \mathbb{R}^{H\times W\times 3}$ is first normalized with ImageNet statistics and then passed through DINOv3 to obtain patch-level features:

$$\mathbf{Z}_v = \mathrm{DINOv3}(\mathbf{I}_v) \in \mathbb{R}^{N\times d}, \quad N = \left(\frac{H}{p}\right)^2, \tag{2}$$

where $p = 16$ is the patch size, N the number of spatial tokens, and $d = 768$ the feature dimension. We discard the `[CLS]` token and any register tokens, retaining only the spatial patch tokens. All DINOv3 parameters remain frozen throughout training.

Camera tokenization. For a pixel at location (u, w) in view v with intrinsics (f_x, f_y, c_x, c_y) and camera-to-world matrix $\mathbf{T}_v = [\mathbf{R}_v \mid \mathbf{t}_v]$, we first compute the ray direction in world coordinates:

$$\mathbf{d}_{u,w} = \mathbf{R}_v \cdot \begin{pmatrix} (u + 0.5 - c_x)/f_x \\ (w + 0.5 - c_y)/f_y \\ 1 \end{pmatrix}, \quad \hat{\mathbf{d}}_{u,w} = \frac{\mathbf{d}_{u,w}}{\|\mathbf{d}_{u,w}\|}, \tag{3}$$

and the ray origin $\mathbf{o}_v = \mathbf{t}_v$. The 6D Plücker representation is then:

$$\boldsymbol{\pi}_{u,w} = [\mathbf{o}_v \times \hat{\mathbf{d}}_{u,w};\ \hat{\mathbf{d}}_{u,w}] \in \mathbb{R}^6, \tag{4}$$

where the first three components are the moment vector and the last three are the direction.

The Plücker coordinate map $\mathbf{\Pi}_v \in \mathbb{R}^{H\times W\times 6}$ for view v is tokenized into patches using a linear projection (analogous to the image tokenizer in DINOv3) to produce camera tokens $\mathbf{P}_v \in \mathbb{R}^{N\times d}$.

Token fusion. For each reference view $v \in \mathcal{R}$, we augment both image and camera tokens with positional embeddings, which are fixed 2D sinusoidal positional embeddings passed through a separate two-layer MLP for each modality. Each MLP consists of a linear layer, SiLU activation, and another linear layer. We concatenate the augmented tokens $\mathbf{Z}_v^{'}$ and $\mathbf{P}_v^{'}$ along the feature dimension and fuse them with a two-layer MLP:

$$\mathbf{F}_v = \mathrm{MLP}_{\mathrm{fuse}}\big([\mathbf{Z}_v^{'};\ \mathbf{P}_v^{'}]\big) \in \mathbb{R}^{N\times d}, \tag{5}$$

where $\mathrm{MLP}_{\mathrm{fuse}} : \mathbb{R}^{2d} \to \mathbb{R}^d$ consists of layer normalization, a linear layer, SiLU activation, and another linear layer.

Geometry transformer. The fused tokens from all reference views are collected into a single sequence $\mathbf{F} = [\mathbf{F}_{v_1}; \ldots; \mathbf{F}_{v_{|\mathcal{R}|}}] \in \mathbb{R}^{|\mathcal{R}|N\times d}$ and processed by a geometry transformer. Following VGGT [20], the transformer consists of L layers with QK-normalized multi-head self-attention [75] that alternate between two attention patterns:

- **Frame attention** (even layers): attention is computed independently within each view, allowing the model to reason about local 2D structure. Tokens are reshaped to $(|\mathcal{R}| \cdot B) \times N \times d$, where B is the batch size, so that each view attends only to its own tokens.
- **Global attention** (odd layers): attention spans all views jointly, enabling cross-view reasoning about 3D geometry. Tokens are reshaped to $B \times (|\mathcal{R}| \cdot N) \times d$ so that each token attends across all reference views.

This alternating strategy allows the model to interleave 2D appearance processing with 3D multi-view aggregation, balancing computational efficiency with geometric expressiveness. Layer weights are initialized with a depth-dependent standard deviation $\sigma_\ell = 0.02/\sqrt{2(\ell+1)}$ to stabilize training of deep transformers [20].

3D Gaussian prediction. The output tokens from the geometry transformer are decoded into per-patch 3D Gaussian parameters via a linear head. Each spatial token predicts one 3D Gaussian, yielding $|\mathcal{R}| \cdot N$ Gaussians in total. Specifically, the decoder predicts the following attributes per Gaussian:

$$\mathbf{g}_i = (\Delta\mathbf{x}_i,\ \mathbf{c}_i,\ \mathbf{s}_i,\ \mathbf{q}_i,\ \alpha_i), \tag{6}$$

where $\Delta\mathbf{x}_i \in \mathbb{R}^3$ is a position offset, $\mathbf{c}_i \in \mathbb{R}^{n_{\text{SH}} \times 3}$ are spherical harmonic (SH) coefficients with $n_{\text{SH}} = (\ell_{\max}+1)^2$ and $\ell_{\max} = 3$ for view-dependent color, $\mathbf{s}_i \in \mathbb{R}^3$ is the log-scale, $\mathbf{q}_i \in \mathbb{R}^4$ is the rotation quaternion, and $\alpha_i \in \mathbb{R}$ is the pre-sigmoid opacity. The log-scale is clamped to $[s_{\min}, s_{\max}]$ for numerical stability.

Hard pixel alignment. To anchor the predicted Gaussians in 3D space, we apply a hard pixel-alignment strategy [36]. The position offset $\Delta\mathbf{x}_i$ is first mapped to a depth value $\delta_i \in [\delta_{\text{near}}, \delta_{\text{far}}]$ via a linear range function. The final 3D position is obtained by marching along the corresponding camera ray:

$$\mathbf{x}_i = \mathbf{o}_v + \delta_i \cdot \hat{\mathbf{d}}_i, \tag{7}$$

where $\mathbf{o}_v$ is the ray origin and $\hat{\mathbf{d}}_i$ is the normalized ray direction at the spatial location of token i. This constrains each Gaussian to lie on the ray cast from its corresponding pixel, which provides a strong geometric prior and improves convergence.

C.2 Loss details

The BEAST3D loss is the weighted sum of three losses: photometric, perceptual, and mask losses.

Photometric loss. We use a foreground-weighted mean squared error between the rendered image $\hat{\mathbf{I}}_v$ and an augmented ground-truth target image $\mathbf{I}_v^*$ for each target view $v \in \mathcal{T}$:

$$\mathcal{L}_{\ell_2} = \frac{1}{|\mathcal{T}|} \sum_{v \in \mathcal{T}} \left[w_{\text{fg}} \cdot \frac{\sum (\hat{\mathbf{I}}_v - \mathbf{I}_v^*)^2 \odot \mathbf{M}_v}{\sum \mathbf{M}_v} + \frac{\sum (\hat{\mathbf{I}}_v - \mathbf{I}_v^*)^2 \odot (1 - \mathbf{M}_v)}{\sum (1 - \mathbf{M}_v)} \right], \tag{8}$$

where $w_{\text{fg}} = 5$ up-weights the foreground region to focus learning on the subject rather than the (augmented) background. During training, a random background color $\mathbf{b} \sim \text{Uniform}([0,1]^3)$ is composited into the initial ground truth image $\mathbf{I}_v$ via $\mathbf{I}_v^* \leftarrow \mathbf{I}_v \odot \mathbf{M}_v + \mathbf{b} \odot (1 - \mathbf{M}_v)$ to prevent the model from memorizing a fixed background.

Perceptual loss. Following E-RayZer [21], we add a VGG-based perceptual loss [76] that extracts multi-scale features from a pretrained VGG-19 network:

$$\mathcal{L}_{\text{perc}} = \frac{1}{|\mathcal{T}|} \sum_{v \in \mathcal{T}} \sum_{l=0}^{5} w_l \cdot \|\phi_l(\hat{\mathbf{I}}_v) - \phi_l(\mathbf{I}_v^*)\|_1, \tag{9}$$

where $\{\phi_l\}$ are the feature maps at selected VGG layers and $\{w_l\}$ are normalization weights. This loss encourages perceptually plausible reconstructions and reduces blurriness.

Mask loss. We supervise the rendered alpha channel $\hat{\mathbf{A}}_v$ (accumulated opacity from Gaussian splatting) against the ground-truth foreground mask:

$$\mathcal{L}_{\text{mask}} = \frac{1}{|\mathcal{T}|} \sum_{v \in \mathcal{T}} \|\hat{\mathbf{A}}_v - \mathbf{M}_v\|_2^2. \tag{10}$$

We use MSE rather than binary cross-entropy to avoid excessively large gradients in background regions, which typically occupy the majority of the image.

D Baselines

D.1 VGGT

VGGT [20] is a feed-forward 3D foundation model that, given an unordered set of multi-view images, jointly predicts per-pixel world-space points (with a per-point confidence), per-view camera extrinsics, and per-view intrinsics in a single forward pass. We use VGGT strictly as an off-the-shelf reference: we do *not* fine-tune it on any of our animal datasets and instead use the public pretrained checkpoint released by the authors[1]. At inference we resize each input view to 224×224 pixels (VGGT's training resolution), run a single forward pass over all available views with bf16 autocast, and recover (i) per-view camera extrinsics and intrinsics, and (ii) a

[1] https://github.com/facebookresearch/vggt

single shared point cloud by concatenating the per-pixel world points across views. To suppress low-confidence background floaters we remove points whose confidence falls in the bottom 50% of all predicted points; the remaining points retain their per-pixel RGB colors and form the VGGT point cloud used in Fig. 1. No alignment to ground-truth cameras is performed for the point cloud itself; renders shown in Fig. 1 are rasterized either through the VGGT-predicted cameras directly (Cheese3D, Rat7M, Chickadee) or through the ground-truth cameras (Human3.6M) after a per-frame Procrustes alignment of the ground truth camera pose into VGGT's predicted world frame (the same alignment used for the NVS evaluation in Appendix E).

D.2 E-RayZer

E-RayZer [21] is a 3D-aware transformer that consumes multi-view images and predicts a set of per-pixel 3D Gaussians, which are then rendered into target views via Gaussian splatting[2]. Unlike VGGT, we treat E-RayZer as a fine-tunable baseline: we initialize from the public pretrained checkpoint and continue self-supervised pretraining on each of our datasets independently. Concretely, the model first tokenizes each input image with a ViT-style image tokenizer, runs a VGGT-style alternating frame/global attention encoder, and decodes per-view camera extrinsics and intrinsics from a learned camera token. These *predicted* per-view cameras are converted into Plücker ray embeddings, fused with the image tokens via an MLP, processed by a geometry transformer, and decoded into per-pixel Gaussian parameters with hard pixel alignment along the corresponding camera ray; the same predicted cameras are then used as the rendering cameras for Gaussian splatting, both during pretraining and for novel-view synthesis at inference.

The four main differences from BEAST3D are: (i) E-RayZer predicts its own cameras rather than consuming the calibrated ground truth cameras as input, and renders through those predicted cameras, whereas BEAST3D consumes and renders through ground truth cameras; (ii) E-RayZer uses a simple patch tokenizer before sending patch embeddings into the geometry transformer, whereas BEAST3D uses a frozen DINOv3 backbone as a much more powerful patch tokenizer, similar to VGGT; (iii) E-RayZer does not enforce the frustum constraint that BEAST3D uses to anchor Gaussians within the visible volume of the reference cameras; and (iv) E-RayZer is trained to reconstruct its inputs with no foreground supervision, whereas BEAST3D adds a foreground-weighted reconstruction term and a mask MSE term that supervise the rendered alpha against the SAM3 segmentation mask (Appendix C). The first three differences shift what *geometry* the model can recover, while the loss difference shifts what part of the image the model is encouraged to fit.

A practical caveat is that E-RayZer was originally pretrained with 10 views per scene split into 5 reference + 5 target, a regime that assumes dense, highly overlapping coverage of the same object (as in scene-scale datasets such as DL3DV [39]). Animal behavior recordings in our lab datasets only provide 4–6 widely separated cameras whose fields of view share a small common volume, and in this sparse-camera regime E-RayZer's pose prediction collapses — the predicted cameras are essentially uncorrelated with the true poses (Fig. 7), which propagates to the Plücker ray embeddings and prevents the geometry transformer from converging.

Fine-tuning. E-RayZer was fine-tuned for 200 epochs with AdamW (lr 5×10^{-5}, weight decay 0.05) at a global batch size of 256, retaining E-RayZer's original loss weights.

D.3 Pose Splatter

Pose Splatter (PS) [22] is a feed-forward multi-view 3D Gaussian splatting model designed for multi-view animal datasets. Given calibrated cameras and per-view foreground masks, PS constructs a coarse voxel occupancy volume via multi-view silhouette intersection ("shape carving"), refines this representation with a 3D U-Net, and decodes occupied voxels into Gaussian parameters for differentiable rendering. Unlike generalizable reconstruction models, PS is trained per dataset.

PS differs from BEAST3D along several design dimensions that affect how 3D structure is inferred. (i) PS uses foreground masks as direct inputs to shape carving at inference time, whereas BEAST3D uses foreground masks only as supervision during training and predicts geometry from unmasked RGB images at inference time. (ii) PS initializes geometry via multi-view silhouette intersection before network refinement, introducing an explicit geometric constraint induced by mask consistency and camera alignment. BEAST3D instead predicts Gaussians directly from learned multi-view image features conditioned on camera rays. (iii) PS is trained to optimize photometric reconstruction of rendered views, while BEAST3D uses held-out-view reconstruction with an additional perceptual loss to combat vision transformer patch artifacts. These differences reflect distinct inductive biases in how geometry and appearance are represented and optimized.

Training. We use the public implementation[3] and train one model from scratch per dataset, following the provided preprocessing pipeline to produce geometry-normalized inputs with the same SAM3-generated foreground masks as those used in BEAST3D training. Models are trained for 50 epochs using Adam with a

[2] https://github.com/QitaoZhao/E-RayZer
[3] https://github.com/jackgoffinet/pose-splatter

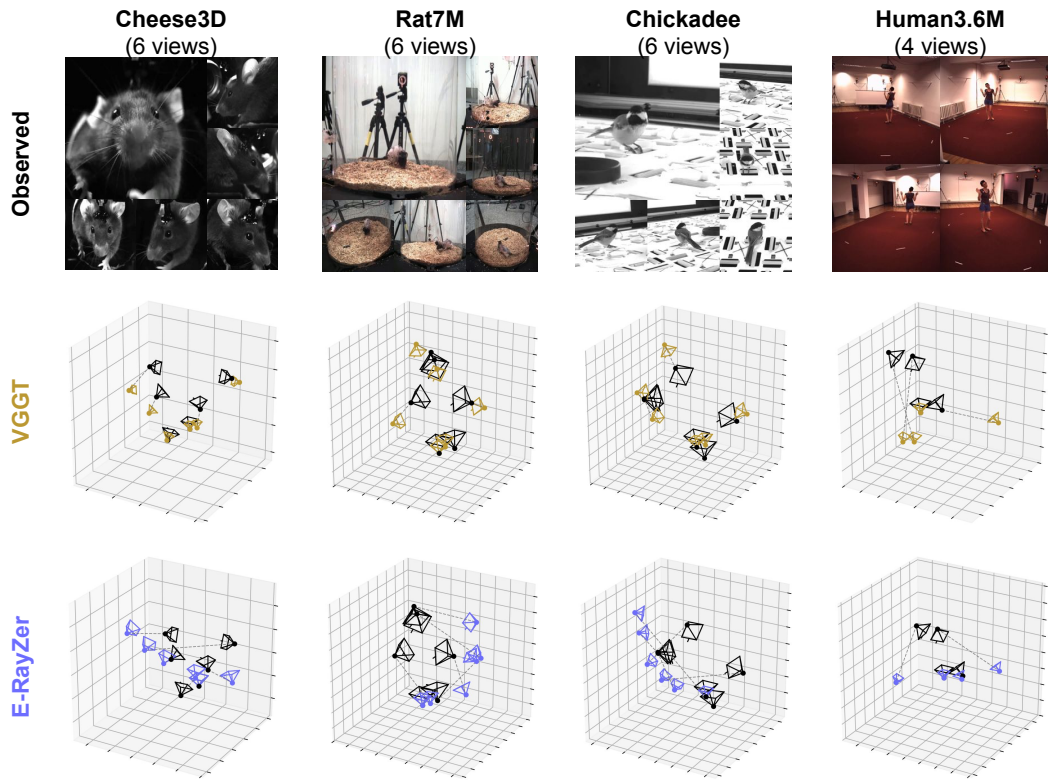

Figure 7: **Camera pose prediction collapses in the sparse-view regime.** *Top:* representative input views. *Middle:* VGGT-predicted cameras (*colored*) paired with ground truth cameras (*black*) via dashed lines. *Bottom:* E-RayZer-predicted cameras, also paired with ground truth via dashed lines. VGGT's predictions stay close to the ground truth, but deviate more strongly on the Human3.6M dataset which only has four views. E-RayZer—which learns camera poses fully unsupervised—produces poses that are nearly uncorrelated with the true cameras under our 4–6 widely separated views.

learning rate of 1×10^{-4} and batch size 1. The optimizer is used without weight decay, following the official implementation. We otherwise follow the official dataset-specific training configurations, including a voxel grid resolution of 64^3 for Cheese3D, Rat7M, and Human3.6M, and 112^3 for Chickadee, with dataset-specific cropping volumes aligned to the canonical object frame.

Evaluation. For evaluation, we compare two shape carving protocols. The *original* protocol follows the PS evaluation setup, where all available views contribute to shape carving. The *leave-one-out* (LOO) protocol matches our held-out target-view setting: for target view v, the shape carver uses only the remaining $V-1$ views at evaluation time, and the model is scored by rendering view v. Table 2 reports this comparison on the InD split with ground truth mask scoring. The original protocol improves PSNR on all datasets. We use the LOO protocol in Tables 3–6.

Table 2: **Effect of Pose Splatter carving protocol on InD GT-mask NVS.** Original uses all available views for shape carving; LOO excludes the rendered target view from the carver at evaluation time.

	Cheese3D			Rat7M			Chickadee			Human3.6M		
Protocol	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓	PSNR↑	SSIM↑	LPIPS↓
Original	13.703	0.600	0.420	8.360	0.699	0.194	12.366	0.630	0.309	10.020	0.742	0.187
LOO	12.443	0.594	0.419	4.519	0.673	0.223	9.064	0.603	0.299	9.648	0.743	0.191

D.4 BEAST

BEAST [43] is a single-view, self-supervised pretraining procedure for animal behavior video. It pretrains a ViT-B/16 backbone with masked autoencoding (MAE) [26] on individual frames, treating each camera view as an independent image with no cross-view fusion or camera input. We pretrain one backbone per dataset using the public implementation[4], with the default recipe (mask ratio 0.75, AdamW optimizer, cosine learning

[4] https://github.com/paninski-lab/beast

rate schedule, 800 epochs); the optional InfoNCE contrastive term is disabled so that pretraining reduces to pure single-view MAE. Because BEAST does not see 3D structure or cameras, it serves both as the natural 2D baseline against BEAST3D and as a fine-tunable backbone for the single-view pose estimation models in Appendix G.

E Novel view synthesis

We report full novel view synthesis (NVS) results across all four datasets (Cheese3D, Rat7M, Chickadee, Human3.6M) and six methods (E-RayZer [21] zero-shot (ZS), E-RayZer fine-tuned (FT) per dataset, Pose Splatter [22], BEAST3D without the frustum constraint, BEAST3D without the DINOv3 backbone, and the full BEAST3D). For each method–dataset pair we evaluate two regimes that match Section 4.1: the In-Distribution Test (InD Test) split, in which held-out frames come from training subjects, and the Out-of-Distribution Test (OOD Test) split, in which test frames come from unseen subjects. We also evaluate each setting under two masking conventions: foreground masked with the "ground truth" SAM3 segmentation mask (Tables 3 and 4), and foreground masked with the rendered alpha channel of each method (Tables 5 and 6). The latter additionally lets us report foreground-segmentation intersection over union (IoU) against the "ground truth" mask. Numbers are mean $\pm$ standard deviation across all evaluation frames; the best result per dataset and metric is highlighted in bold (ties at the displayed precision are bolded jointly). E-RayZer (ZS) and E-RayZer (FT) do not produce a confident foreground alpha and are therefore omitted from the predicted-mask tables. Tables 3 and 4 additionally include BEAST3D (joint), a single model trained on Cheese3D, Rat7M and Human3.6M and evaluated only on those datasets (Appendix E.1).

We provide example InD Test videos for each dataset, which show the synchronized camera views (*top left*), distilled foreground segmentation mask (*top right*), predicted 3D Gaussian splats for each view (*bottom left*), and a free-viewpoint orbit of the reconstructed subject demonstrating novel view synthesis capabilities (*bottom right*): Cheese3D, Rat7M, Chickadee, Human3.6M.

E.1 Joint training across datasets

The other BEAST3D models in this paper are trained separately on each dataset, whereas BEAST3D (joint) is a single model trained on the union of the Cheese3D, Rat7M and Human3.6M InD training sets (Table 1), which together contain 14 subjects from three species. We leave Chickadee out of this initial experiment to keep the setup simple, since its frames are cropped from a large arena with a per-frame bounding box (Appendix A). The camera rigs differ (six cameras for Cheese3D and Rat7M, four for Human3.6M), so each mini-batch is drawn from a single dataset and uses all of that dataset's views. Batches from the three datasets are shuffled together, so each dataset is sampled in proportion to its size. All other settings follow the per-dataset recipe in Section 3.4, except that the first 261 of the 800 epochs ran on 4 GPUs (global batch size 128) instead of 8 (global batch size 256) because of compute availability. We evaluate the last saved checkpoint, at epoch 781.

Tables 3 and 4 report BEAST3D (joint) next to the per-dataset BEAST3D. We compare the two with two-sided Wilcoxon signed-rank tests on paired image metrics, Holm-corrected across the six contrasts within each dataset (two splits times three metrics). All six contrasts are significant at $p < 10^{-20}$ on every dataset. Since thousands of paired frames make even small differences significant, we also check whether each difference is consistent across recording sessions. On the InD Test split, the joint model improves SSIM and LPIPS on all three datasets and in nearly every session. Its PSNR is on par with the per-dataset models, marginally higher on average but not consistently across sessions.

On the OOD Test split, where all subjects are new, the joint model has lower PSNR than the per-dataset models on all three datasets and in every session. On Cheese3D it still improves SSIM and LPIPS in all four OOD sessions, while on Rat7M and Human3.6M it is worse on all three metrics. Rat7M has only two OOD sessions, and most of its gap comes from one of them.

The InD Test results show that one set of weights can reconstruct head-fixed mice, freely moving rats and humans recorded with different camera rigs as well as the per-dataset models do, with consistent gains in SSIM and LPIPS. We view this as encouraging for a cross-subject, cross-species 3D foundation model for animal behavior. Pooling does not, however, improve generalization to new individuals, plausibly because it adds species and camera rigs but not more individuals per species (three to six per dataset). We expect better cross-subject reconstruction to require multi-view training data with many more subjects per species.

Table 3: Novel view synthesis on the In-Distribution Test (InD Test) set (foreground masked with the GT mask). BEAST3D (joint) is a single model trained on Cheese3D, Rat7M and Human3.6M (Appendix E.1) and is not evaluated on Chickadee. ↑ / ↓: higher / lower is better.

Dataset	Model	PSNR↑	SSIM↑	LPIPS↓
Cheese3D	E-RayZer (ZS)	13.645 ± 2.834	0.575 ± 0.110	0.467 ± 0.051
	E-RayZer (FT)	13.131 ± 2.876	0.565 ± 0.098	0.473 ± 0.057
	Pose Splatter	12.443 ± 3.404	0.594 ± 0.113	0.419 ± 0.054
	BEAST3D (no frustum)	26.578 ± 2.698	0.837 ± 0.043	0.224 ± 0.038
	BEAST3D (no DINOv3)	26.388 ± 2.457	0.832 ± 0.044	0.208 ± 0.031
	BEAST3D (joint)	$\mathbf{27.040} \pm 3.031$	$\mathbf{0.855} \pm 0.041$	$\mathbf{0.177} \pm 0.034$
	BEAST3D	26.990 ± 2.659	0.846 ± 0.042	0.194 ± 0.032
Rat7M	E-RayZer (ZS)	12.536 ± 3.271	0.772 ± 0.075	0.235 ± 0.048
	E-RayZer (FT)	12.398 ± 3.314	0.764 ± 0.081	0.237 ± 0.049
	Pose Splatter	4.519 ± 1.845	0.673 ± 0.092	0.223 ± 0.057
	BEAST3D (no frustum)	22.005 ± 1.934	0.900 ± 0.038	0.113 ± 0.032
	BEAST3D (no DINOv3)	22.260 ± 1.947	0.906 ± 0.038	0.105 ± 0.030
	BEAST3D (joint)	$\mathbf{22.468} \pm 1.950$	$\mathbf{0.914} \pm 0.038$	$\mathbf{0.092} \pm 0.029$
	BEAST3D	22.423 ± 1.902	0.909 ± 0.036	0.102 ± 0.029
Chickadee	E-RayZer (ZS)	10.173 ± 3.041	0.631 ± 0.102	0.298 ± 0.060
	E-RayZer (FT)	9.537 ± 3.062	0.628 ± 0.098	0.298 ± 0.058
	Pose Splatter	9.064 ± 3.172	0.603 ± 0.095	0.299 ± 0.059
	BEAST3D (no frustum)	16.760 ± 1.273	0.708 ± 0.070	0.239 ± 0.043
	BEAST3D (no DINOv3)	18.293 ± 1.783	0.767 ± 0.069	0.187 ± 0.037
	BEAST3D	$\mathbf{18.592} \pm 1.751$	$\mathbf{0.771} \pm 0.064$	$\mathbf{0.186} \pm 0.038$
Human3.6M	E-RayZer (ZS)	11.893 ± 2.591	0.769 ± 0.061	0.259 ± 0.054
	E-RayZer (FT)	13.231 ± 1.733	0.786 ± 0.053	0.258 ± 0.056
	Pose Splatter	9.648 ± 2.335	0.743 ± 0.063	0.191 ± 0.048
	BEAST3D (no frustum)	22.206 ± 2.080	0.929 ± 0.031	0.098 ± 0.036
	BEAST3D (no DINOv3)	$\mathbf{22.701} \pm 2.007$	$\mathbf{0.934} \pm 0.029$	0.091 ± 0.035
	BEAST3D (joint)	22.450 ± 2.380	$\mathbf{0.934} \pm 0.032$	$\mathbf{0.086} \pm 0.034$
	BEAST3D	22.386 ± 2.169	0.931 ± 0.031	0.092 ± 0.035

Table 4: Novel view synthesis on the Out-of-Distribution Test (OOD Test) set (foreground masked with the GT mask). BEAST3D (joint) is a single model trained on Cheese3D, Rat7M and Human3.6M (Appendix E.1) and is not evaluated on Chickadee.

Dataset	Model	PSNR↑	SSIM↑	LPIPS↓
Cheese3D	E-RayZer (ZS)	12.981 ± 2.848	0.578 ± 0.101	0.467 ± 0.041
	E-RayZer (FT)	12.307 ± 2.752	0.563 ± 0.082	0.480 ± 0.047
	Pose Splatter	12.452 ± 2.873	0.592 ± 0.095	0.420 ± 0.043
	BEAST3D (no frustum)	20.175 ± 1.852	0.712 ± 0.050	0.298 ± 0.033
	BEAST3D (no DINOv3)	$\mathbf{21.220} \pm 2.060$	$\mathbf{0.732} \pm 0.049$	0.281 ± 0.025
	BEAST3D (joint)	20.476 ± 2.059	0.728 ± 0.051	$\mathbf{0.257} \pm 0.028$
	BEAST3D	20.736 ± 2.000	0.720 ± 0.047	0.273 ± 0.030
Rat7M	E-RayZer (ZS)	12.322 ± 2.650	0.767 ± 0.078	0.235 ± 0.048
	E-RayZer (FT)	12.889 ± 2.682	0.751 ± 0.087	0.259 ± 0.066
	Pose Splatter	5.188 ± 2.200	0.682 ± 0.094	0.221 ± 0.058
	BEAST3D (no frustum)	$\mathbf{16.265} \pm 2.569$	$\mathbf{0.811} \pm 0.061$	0.153 ± 0.043
	BEAST3D (no DINOv3)	15.936 ± 2.510	0.810 ± 0.060	0.151 ± 0.042
	BEAST3D (joint)	13.883 ± 2.574	0.752 ± 0.065	0.186 ± 0.046
	BEAST3D	16.006 ± 2.360	0.804 ± 0.061	$\mathbf{0.147} \pm 0.044$
Chickadee	E-RayZer (ZS)	10.500 ± 3.110	$\mathbf{0.667} \pm 0.105$	0.283 ± 0.063
	E-RayZer (FT)	9.454 ± 2.959	0.664 ± 0.104	0.287 ± 0.066
	Pose Splatter	8.601 ± 1.868	0.651 ± 0.090	0.253 ± 0.052
	BEAST3D (no frustum)	8.664 ± 1.618	0.523 ± 0.125	0.329 ± 0.080
	BEAST3D (no DINOv3)	9.771 ± 1.668	0.547 ± 0.109	0.297 ± 0.065
	BEAST3D	$\mathbf{13.726} \pm 1.746$	0.650 ± 0.105	$\mathbf{0.235} \pm 0.050$
Human3.6M	E-RayZer (ZS)	11.613 ± 3.623	0.724 ± 0.083	0.274 ± 0.056
	E-RayZer (FT)	12.795 ± 2.495	0.749 ± 0.064	0.275 ± 0.062
	Pose Splatter	8.560 ± 2.472	0.698 ± 0.068	0.225 ± 0.047
	BEAST3D (no frustum)	17.214 ± 1.662	0.806 ± 0.051	0.179 ± 0.037
	BEAST3D (no DINOv3)	$\mathbf{18.712} \pm 1.984$	$\mathbf{0.830} \pm 0.052$	$\mathbf{0.159} \pm 0.039$
	BEAST3D (joint)	16.447 ± 1.687	0.784 ± 0.056	0.192 ± 0.038
	BEAST3D	17.107 ± 1.697	0.801 ± 0.052	0.180 ± 0.037

Table 5: Novel view synthesis and foreground segmentation on the In-Distribution Test (InD Test) set (rendered alpha as the foreground mask).

Dataset	Model	PSNR↑	SSIM↑	LPIPS↓	IoU↑
Cheese3D	Pose Splatter	15.255 ± 2.602	0.580 ± 0.109	0.440 ± 0.052	0.740 ± 0.079
	BEAST3D (no frustum)	26.087 ± 2.856	0.831 ± 0.046	0.228 ± 0.039	$\mathbf{0.933} \pm 0.059$
	BEAST3D (no DINOv3)	25.969 ± 2.649	0.827 ± 0.047	0.212 ± 0.033	0.899 ± 0.058
	BEAST3D	$\mathbf{26.623} \pm 2.838$	$\mathbf{0.842} \pm 0.044$	$\mathbf{0.197} \pm 0.033$	0.903 ± 0.058
Rat7M	Pose Splatter	9.430 ± 2.127	0.580 ± 0.103	0.377 ± 0.064	0.129 ± 0.149
	BEAST3D (no frustum)	21.956 ± 1.965	0.900 ± 0.039	0.113 ± 0.032	0.500 ± 0.087
	BEAST3D (no DINOv3)	22.160 ± 1.994	0.905 ± 0.038	0.106 ± 0.030	0.548 ± 0.097
	BEAST3D	$\mathbf{22.327} \pm 1.945$	$\mathbf{0.908} \pm 0.037$	$\mathbf{0.103} \pm 0.030$	$\mathbf{0.551} \pm 0.093$
Chickadee	Pose Splatter	10.256 ± 2.491	0.499 ± 0.134	0.395 ± 0.075	0.259 ± 0.164
	BEAST3D (no frustum)	16.640 ± 1.366	0.706 ± 0.071	0.242 ± 0.043	0.547 ± 0.055
	BEAST3D (no DINOv3)	18.029 ± 1.945	0.764 ± 0.072	0.191 ± 0.039	$\mathbf{0.692} \pm 0.085$
	BEAST3D	$\mathbf{18.314} \pm 1.910$	$\mathbf{0.768} \pm 0.066$	$\mathbf{0.189} \pm 0.039$	0.681 ± 0.081
Human3.6M	Pose Splatter	13.998 ± 1.688	0.731 ± 0.068	0.224 ± 0.064	0.581 ± 0.120
	BEAST3D (no frustum)	22.102 ± 2.126	0.928 ± 0.032	0.098 ± 0.036	0.695 ± 0.055
	BEAST3D (no DINOv3)	$\mathbf{22.624} \pm 2.046$	$\mathbf{0.933} \pm 0.029$	$\mathbf{0.091} \pm 0.035$	$\mathbf{0.718} \pm 0.056$
	BEAST3D	22.272 ± 2.223	0.930 ± 0.032	0.093 ± 0.035	0.710 ± 0.054

Table 6: Novel view synthesis and foreground segmentation on the Out-of-Distribution Test (OOD Test) set (rendered alpha as the foreground mask).

Dataset	Model	PSNR↑	SSIM↑	LPIPS↓	IoU↑
Cheese3D	Pose Splatter	14.300 ± 2.710	0.572 ± 0.095	0.447 ± 0.044	0.740 ± 0.062
	BEAST3D (no frustum)	19.569 ± 1.750	0.700 ± 0.050	0.305 ± 0.032	$\mathbf{0.886} \pm 0.039$
	BEAST3D (no DINOv3)	$\mathbf{20.607} \pm 2.068$	$\mathbf{0.720} \pm 0.051$	0.292 ± 0.026	0.874 ± 0.043
	BEAST3D	20.209 ± 1.879	0.709 ± 0.047	$\mathbf{0.280} \pm 0.031$	0.877 ± 0.044
Rat7M	Pose Splatter	9.928 ± 2.165	0.607 ± 0.100	0.356 ± 0.062	0.175 ± 0.136
	BEAST3D (no frustum)	$\mathbf{15.853} \pm 2.716$	$\mathbf{0.802} \pm 0.064$	0.170 ± 0.049	0.343 ± 0.135
	BEAST3D (no DINOv3)	15.330 ± 2.617	0.798 ± 0.062	0.174 ± 0.047	0.355 ± 0.157
	BEAST3D	15.507 ± 2.453	0.792 ± 0.065	$\mathbf{0.169} \pm 0.051$	$\mathbf{0.400} \pm 0.112$
Chickadee	Pose Splatter	9.943 ± 1.747	0.560 ± 0.105	0.355 ± 0.062	0.382 ± 0.130
	BEAST3D (no frustum)	7.933 ± 1.443	0.494 ± 0.129	0.389 ± 0.081	0.105 ± 0.080
	BEAST3D (no DINOv3)	8.779 ± 1.460	0.514 ± 0.115	0.359 ± 0.067	0.148 ± 0.074
	BEAST3D	$\mathbf{13.016} \pm 1.817$	$\mathbf{0.634} \pm 0.107$	$\mathbf{0.258} \pm 0.054$	$\mathbf{0.462} \pm 0.113$
Human3.6M	Pose Splatter	15.281 ± 1.569	0.685 ± 0.073	0.265 ± 0.064	0.562 ± 0.151
	BEAST3D (no frustum)	16.888 ± 1.731	0.794 ± 0.056	0.185 ± 0.040	0.635 ± 0.067
	BEAST3D (no DINOv3)	$\mathbf{18.427} \pm 2.018$	$\mathbf{0.822} \pm 0.055$	$\mathbf{0.163} \pm 0.041$	0.625 ± 0.068
	BEAST3D	16.741 ± 1.764	0.788 ± 0.057	0.187 ± 0.040	$\mathbf{0.640} \pm 0.068$

F Inference compute cost

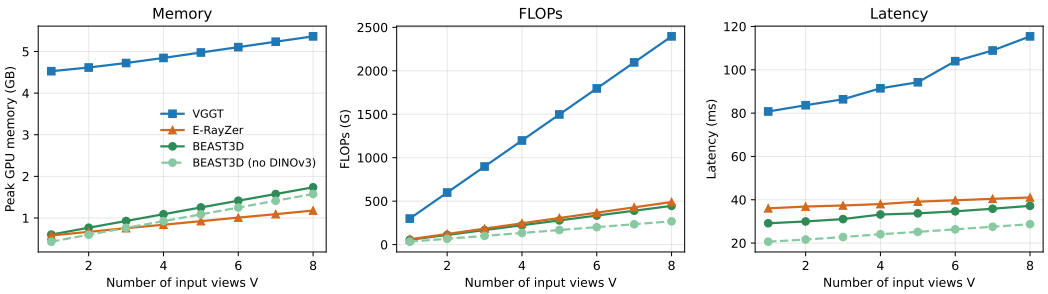

Figure 8: **Inference compute cost vs. number of input views.** Peak GPU memory (left), FLOPs (middle), and median latency over 20 timed iterations after 5 warmup iterations (right) for VGGT, E-RayZer, and BEAST3D (with and without DINOv3), swept over $V \in \{1, \ldots, 8\}$ at batch size 1, 256×256 input. Default deployment config, single GPU, bfloat16 autocast.

We benchmark inference cost of BEAST3D, VGGT, and E-RayZer as the number of input views grows (Fig. 8). All models run in their default deployment configuration on a single GPU with synthetic 256×256 inputs in bfloat16. VGGT is the most expensive along every axis (roughly $5\times$ the FLOPs and latency of the lightweight models). BEAST3D and E-RayZer have comparable compute and memory profiles, with the DINOv3 frontend in BEAST3D adding a roughly constant per-view memory and latency overhead.

G Pose estimation

G.1 Model training

We evaluate five backbone architectures spanning both single-view and 3D-aware multi-view models. All models share a common training framework built on Lightning Pose [54, 14], with architecture-specific modifications described below. For all models, backbone features are transformed into per-view keypoint heatmaps using a lightweight head consisting of transposed convolutions, following ViTPose [77], which demonstrated that ViT backbones are sufficiently expressive that simple linear/deconvolution heads are sufficient for strong pose estimation performance. Standard DLC-style augmentations [78]—including rotation, scaling, and cropping—are applied during training. Images are resized to 256×256 pixels and heatmaps are downsampled by a factor of 4, yielding 64×64 target heatmaps per keypoint per view. All models are trained for 300 epochs with the Adam optimizer and a multi-step learning rate schedule that decays the rate by a factor of 0.5 at epochs 150, 200, and 250. We employ a two-stage training strategy: backbone weights are frozen for the first 20 epochs to allow the randomly initialized prediction head to adapt to the backbone's feature space, after which the entire network is fine-tuned end-to-end. We use 95% of labeled frames for training and 5% for validation, and sample 100 labeled frames across all views per dataset for all experiments. We monitor the validation loss every 5 epochs and retain the checkpoint with the lowest validation loss as the final model.

Single-view heatmap models. The first group of methods processes each camera view independently in the same way (Fig. 9) through a shared view-agnostic backbone followed by the heatmap head. We consider two backbones in this category:

- **ViT-B DINOv3**: A ViT-B pretrained with DINOv3 self-supervised learning [25], which learns semantically rich patch-level representations without explicit supervision.
- **BEAST**: A ViT-B pretrained using the BEAST self-supervised pretraining procedure on the target dataset [43], then fine-tuned for pose estimation. This two-stage approach provides dataset-specific feature initialization, and was previously shown to outperform convolution-based architectures (i.e. ResNet-50) as well as other transformer-based architectures (e.g., DINOv1 and DINOv2).

In both single-view models, per-view heatmaps are predicted independently and the final 2D keypoint locations are obtained via soft-argmax on each heatmap. These models use a training batch size of 8.

Multi-view 3D-aware models. The second group of methods explicitly leverages multi-view geometry by fusing information across camera views within the model architecture. We consider three architectures:

- **VGGT**: A ViT-L backbone pretrained with DINOv2, paired with an alternating attention aggregator that performs iterative frame-level and global-level cross-view attention to fuse multi-view features. VGGT is the largest model in our comparison.

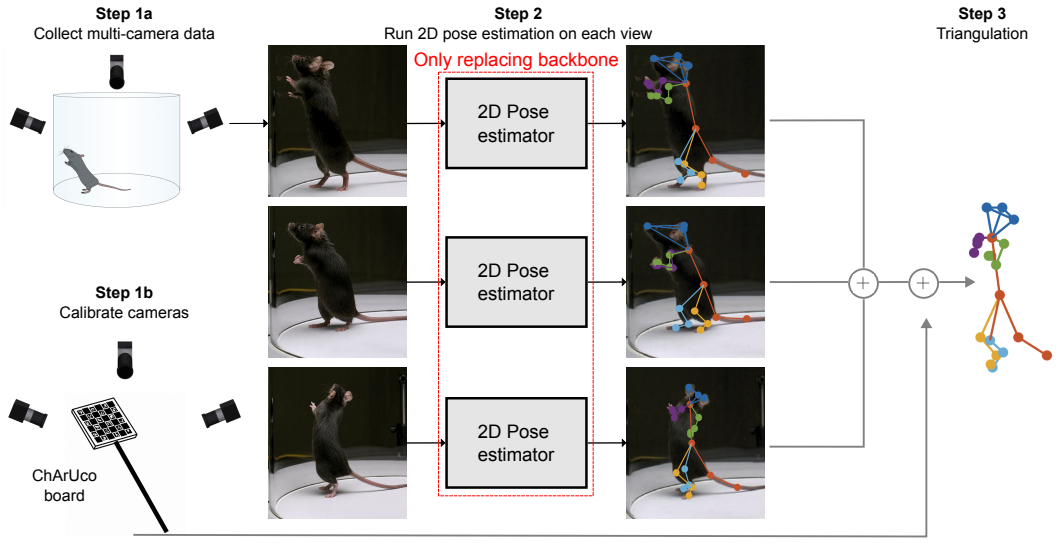

Figure 9: **Pose estimation pipeline for single-view heatmap models.** Step 1: Collect synchronized multi-camera data and calibrate cameras using a ChArUco board. Step 2: Run 2D pose estimation independently on each view, sweeping the backbone across single-view heatmap models for comparison. Step 3: Triangulate the per-view 2D predictions into 3D keypoints using the calibrated camera parameters. Some images modified from [9].

- **E-RayZer**: A 3D-aware transformer that uses a ViT-B backbone with VGGT-style cross-view attention layers to aggregate information across views. The backbone is initialized from dataset-specific E-RayZer pretrained weights.
- **BEAST3D**: Extends the BEAST pretraining paradigm to the 3D setting, using a ViT-B backbone with cross-view transformer layers. The backbone is initialized from dataset-specific BEAST3D pretrained weights.

E-RayZer and BEAST3D use a training batch size of 8 and require camera calibration parameters as additional input. For VGGT, we reduce the training batch size to 4 and employ gradient accumulation over 2 steps (effective batch size of 8) to accommodate the model's memory requirements on a single GPU.

G.2 Hyperparameter selection

For each (backbone, dataset) pair, we perform an independent learning rate sweep over five candidates spanning two orders of magnitude: $\{1e{-}5, 5e{-}5, 1e{-}4, 5e{-}4, 1e{-}3\}$. We train three models per learning rate with different random seeds controlling both the data split and heatmap head weight initialization, record the minimum validation heatmap MSE achieved during training for each seed, and average across seeds. The learning rate minimizing this averaged metric is used for all subsequent training and evaluation. Selecting per pair ensures that no method is penalized by a shared hyperparameter choice.

G.3 Additional results

Ablations. We report pose estimation results using the BEAST3D ablation without the per-view DINOv3 encoder, which produces substantially worse results (Fig. 10).

3D error. The pose estimation results in Fig. 4 report 2D pixel error, aggregated across all views; we additionally report 3D error in physical space measured in millimeters (Fig. 11).

Few-shot pose estimation. All previous pose estimation results fine-tune backbones on 100 labeled frames (Figs. 4, 10, 11). To further test the few-shot abilities of the different backbones, we also fine-tuned on 25 and 50 labeled frames, again using 3 seeds each (different train/val splits per seed), with the 25-frame set nested within the 50-frame set and the 50-frame set nested within the 100-frame set for a given seed. We repeated the error-versus-uncertainty quantification on the held-out test sets for each label budget (Fig. 12), and find that BEAST3D maintains its general superiority across all budgets.

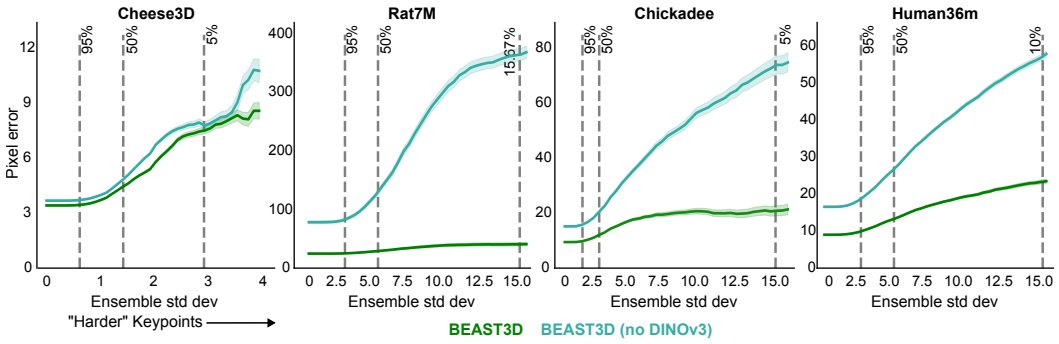

Figure 10: **Pose estimation results with DINOv3 ablation.** Figure conventions as in Fig. 4.

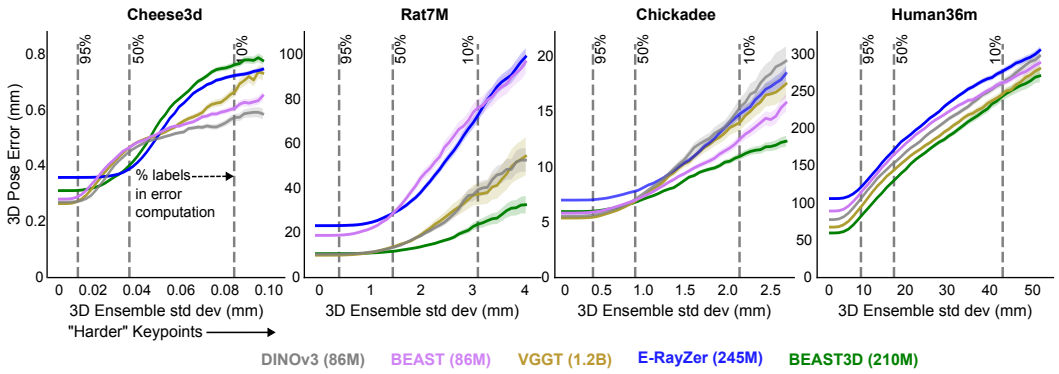

Figure 11: **3D pose estimation results.** Figure conventions as in Fig. 4.

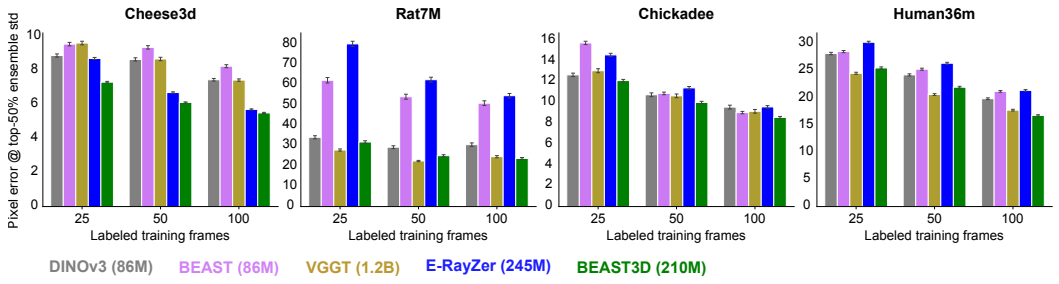

Figure 12: **Few-shot pose estimation across labeling budgets.** Pixel error at the 50% ensemble standard deviation threshold (lower is better) for backbones fine-tuned on 25, 50, or 100 labeled training frames, evaluated on held-out OOD test subjects. Bars show mean±s.e.m. across three seeds (distinct train/val splits). Backbone parameter counts are shown in the legend.

H Neural encoding

We assess whether the 3D representations recovered by BEAST3D carry information about simultaneously recorded neural activity by training a simple encoding model that maps a window of 3D structure to a window of per-neuron firing rates. The same model architecture and training recipe are used for both Cheese3D and Chickadee datasets; only the inputs differ. Table 7 summarizes the two electrophysiology datasets used in this section.

Table 7: Summary of the two electrophysiology datasets used for neural encoding. "Spike windows" counts the non-overlapping 2.0 s segments within the video used as encoding trials.

	Cheese3D	**Chickadee**
Subject / area	mouse facial motor nucleus [18]	chickadee hippocampus [3]
Views	6	6
Video FPS	50 Hz (downsampled)	60 Hz (native)
Video frames	61,697 (≈20.6 min)	54,000 (15 min)
Spike binning	50 Hz (matched to video)	60 Hz (matched to video)
Window length	2.0 s = 100 frames	2.0 s = 120 frames
Train / val / test windows	431 / 92 / 93 (total 616)	315 / 67 / 68 (total 450)
3D keypoints per frame	28	18
Point-cloud points per frame	8,192	1,024
Sorted units (raw)	8	132
Filter policy	none	1 Hz
Units used for encoding	8	52

Per-frame 3D point clouds. For each session, we run the pretrained BEAST3D model in a held-out per-frame inference pass. For every video frame and every reference view, the geometry transformer predicts one pixel-aligned 3D Gaussian per spatial token (Eq. 7); we only keep the centroid of each Gaussian and discard the rotation, opacity, scale, and spherical-harmonic parameters that are only used for rendering. We retain only Gaussians whose predicted opacity exceeds 0.05 *and* whose corresponding pixel falls inside the SAM3 foreground mask, which removes background floaters and emphasizes the subject. For Cheese3D we additionally clip points to a fixed axis-aligned bounding box around the head and keep at most the top 8,192 points per frame ranked by opacity; for Chickadee no bounding box is applied and the full foreground point set is kept. The resulting per-frame point clouds, together with their integer frame indices, are concatenated across the session and serve as the visual input to the encoding model.

Encoding model. The encoding model is intentionally lightweight and input-agnostic so that it does not itself learn 3D structure that could compensate for a weak input. Each frame's point cloud is sub-sampled to a fixed cardinality P (P=8192 for Cheese3D, P=1024 for Chickadee), optionally centered and unit-normalized, and encoded by a per-frame PointNet [61]: a shared MLP $(3 \to 64 \to 128 \to 256)$ with batch norm and ReLU is applied to every point, followed by global max pooling and a two-layer projection to a d=256 embedding. The resulting (B, T, d) sequence of per-frame embeddings is processed by a 2-layer Transformer encoder (4 heads, GELU, pre-norm, learned positional embeddings up to length 256), and a linear readout maps each timestep to N neuron-specific rate predictions. The output is passed through a softplus to enforce positivity since we train with a Poisson negative log-likelihood loss, $\mathcal{L} = \frac{1}{BTN} \sum(\hat{\lambda}_{btn} - y_{btn} \log \hat{\lambda}_{btn})$, where y_{btn} is the observed spike count.

Training. We optimize with AdamW (weight decay 0.05, gradient clip 1.0) for 300 epochs using a cosine schedule from the configured peak learning rate down to 10^{-6} (peak 10^{-5} for Cheese3D, 10^{-3} for Chickadee, set per dataset). The training batch size is 8 for the dense Cheese3D point clouds and 32 for the smaller Chickadee point clouds. We track validation bits-per-spike (BPS) every epoch and retain the checkpoint with the highest validation BPS for test-set evaluation. Because the train / val / test windows are non-overlapping in time and disjoint at the window level, a model that only memorized the training windows cannot trivially extrapolate to test windows.

Baseline. Holding the encoding architecture and training recipe fixed, we vary only the visual input so that any difference in test BPS reflects the input representation rather than the readout. Our model uses the foreground-filtered BEAST3D point cloud, which is produced fully self-supervised: BEAST3D is pretrained without any keypoint or pose annotations, and the per-frame point cloud is read out directly from the pretrained encoder without any task-specific labels. As a baseline, we replace this input with multi-view-triangulated 3D keypoints (outputs of the ensemble Kalman smoother (EKS) [54]; 28 keypoints for Cheese3D, 18 for Chickadee; see Appendix A), which are produced by a supervised pose-estimation model trained on manually labeled

keypoints in each view and then triangulated across views. This baseline therefore represents the prevailing label-dependent input modality for downstream behavioral analyses, whereas BEAST3D requires no manual labeling.

Appearance control. The point clouds described above carry geometry only. Because every retained Gaussian is pixel-aligned, each 3D point can additionally be assigned the foreground-masked RGB color of the image pixel from which it was predicted, which makes it possible to separate the contribution of geometry from that of appearance. Holding the point set, the encoding model, and the training procedure fixed, we compare three per-point input representations: the 3D coordinates alone (XYZ, $P\times3$), the color alone (RGB, $P\times3$), and their concatenation (XYZ + RGB, $P\times6$). The RGB-only model has no access to spatial structure and thus serves as an appearance-only control. Table 8 reports test BPS for both datasets. Color alone falls below the mean-rate null model on both datasets, and appending color to the coordinates leaves performance essentially unchanged. The encoding performance of BEAST3D is therefore attributable to the recovered 3D structure rather than to image appearance.

Table 8: Appearance control for neural encoding. Test bits-per-spike (BPS, averaged over units) of the encoding model with the 3D coordinates (XYZ), the per-point color (RGB), or both (XYZ + RGB) as input. P denotes the number of points per frame (P=8192 for Cheese3D, P=1024 for Chickadee). Learning rates are selected on the validation split.

Model input	**Input dim**	**Test BPS**
Cheese3D		
XYZ	$P\times3$	0.3903
RGB	$P\times3$	−0.0430
XYZ + RGB	$P\times6$	0.3903
Chickadee		
XYZ	$P\times3$	0.1141
RGB	$P\times3$	−0.0090
XYZ + RGB	$P\times6$	0.1188

I Broader impacts

The BEAST3D framework enables more efficient extraction of meaningful information from multi-view video data, potentially accelerating behavioral neuroscience research with several beneficial outcomes. By reducing the need for extensive human labeling while improving accuracy, BEAST3D can democratize advanced video analysis capabilities for laboratories with limited resources. This efficiency could accelerate basic science discoveries that underlie advances in biomedical applications, neurological disorder treatments, and improved understanding of brain function. While BEAST3D is developed primarily for behavioral neuroscience studies using animal subjects, the underlying technology could potentially be repurposed for human video analysis, raising several concerns:

- Surveillance capabilities: The improved ability to track behaviors could enhance surveillance technologies, potentially infringing on privacy rights if deployed without appropriate oversight.
- Bias and fairness: As with any AI system trained on specific datasets, BEAST3D-derived models may perform differently across demographic groups if applied to human subjects, potentially perpetuating biases in downstream applications.
- Resource inequality: While a pretrained BEAST3D model can improve the efficiency of downstream tasks, the computational requirements for pretraining itself may limit access to this technology for under-resourced institutions, potentially widening existing disparities in research capabilities.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We clearly declare and introduce our main contribution as a self-supervised 3D Gaussian splat technique for animal behavior videos.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Conclusion section for limitations of the work.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: This paper does not include theoretical results.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.

- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We include model architecture and training details in the main text and Appendix. We also provide hyperparameter ranges and selection strategy for all model training experiments.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: Code and data preparation instructions are available at https://github.com/paninski-lab/beast. URLs for public datasets are in the Appendix.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We provide all training and hyperparameter details in the Appendix.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: We report error bars in tables and plots and describe how they are calculated in the corresponding captions or accompanying text.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [No]

   Justification: We report an estimate of the required compute resources for initial pretraining in the Conclusion, but do not provide compute resource estimates for the downstream tasks.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.

- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We have carefully read and followed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: We discuss broader impacts in the Appendix.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: The paper does not include any data or models that have a high risk for misuse.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.

- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: For all existing assets we clearly cite the original paper and provide the name of the license.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [N/A]

    Justification: This paper does not release new assets.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: This paper does not involve research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: The core method development in this paper does not involve LLMs.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.