Supplementary Material: Probing the 3D Awareness of Visual Foundation Models

Mohamed El Banani[1] Amit Raj[2] Kevis-Kokitsi Maninis[2] Abhishek Kar[2] Yuanzhen Li[2]
Michael Rubinstein[2] Deqing Sun[2] Leonidas Guibas[2] Justin Johnson[1] Varun Jampani [2*]
[1] University of Michigan [2] Google Research

A. Additional Experimental Details

We provide a high-level summary of the experimental setup in the main body of the paper and omitted details to enhance readability. In this section, we provide a more extensive coverage of our experimental setup, as well as explain the rationale behind some of our design choices. We will also release the code at https://github.com/mbanani/probe3d to allow others to replicate and expand on our analysis.

A.1. Visual Foundation Models

Our experiments consider 26 checkpoints that span ten learning objectives that cover five different forms of supervision. The models were chosen with two criteria in mind: (1) coverage of the main approaches used for large-scale training and (2) comparable model and training scale to allow comparisons. We only use publicly-available checkpoints to understand the 3D awareness of the models that are commonly used. Although we tried to find comparable models, we note that these models were trained on different data sets using different recipes, which confounds our findings, as discussed in App. C.

We discuss all the models we used, explain their learning objective, and specify the checkpoints we used. While many of our comparisons focused on the models listed in Table 1, we used all checkpoints to calculate the correlations presented in Figure 8. We also discuss some of the additional trends that we observed in App. B.

MAE. He et al. [13] showed that training vision transformers to reconstruct images based on randomly masked inputs is an effective pretraining task. Such models are trained with a large masking ratio; *e.g.*, 75% of the input image patches are masked. In our experiments, we use the ViT-B/16 and ViT-L/16 models trained on ImageNet-1k. We use the checkpoint[1] available on the Transformers library [41].

FCMAE. Fully-convolutional masked autoencoders (FCMAE) are similarly trained to reconstruct images. However, unlike MAE, they use a ConvNeXt backbone instead of visual transformers. Woo et al. [42] extended the previous work to convolutional architectures and proposed several architectural changes to further improve performance. In our experiments, we use the ConvNeXtv2 base architecture, which has a model capability comparable to the base visual transformer architectures. Following the analysis of Goldblum et al. [12], we use the model pre-trained on ImageNet-22k to allow a more comparable training data to other models. We use the checkpoints available in the timm library [40]. [2]

DINO. Caron et al. [5] proposed a self-distillation approach for model pretraining. The proposed approach trains a student network to generate features similar to a teacher network, where the teacher is an exponential moving average of the student network. At its core, this approach relies on instance discrimination as the model is trained to learn to generate similar embeddings for different crops of the same image instance. In our work, we evaluate the ViT-B/16 architecture trained on ImageNet-1k. We use the checkpoint released by the authors.[3]

iBOT. Zhou et al. [50] combine ideas from DINO and MAE by training a model to reconstruct masked dense features based on a teacher network. iBOT uses both an image-level and a dense distillation objective. We analyze the ViT-B/16 and ViT-L/16 architectures trained on ImageNet-1k and ImageNet-22k. We evaluate the checkpoints released by the authors.[4]

DINOv2. Oquab et al. [26] scale up the hybrid approach proposed by Zhou et al. [50] while improving the training recipe and incorporating improved losses and regularizers. Furthermore, the training data and recipe are both scaled up in magnitude, resulting in much better performance. This includes the collection of a large curated private dataset called LVD-142M which is curated through the use of the clustered

* Current affiliation is Stability AI.

[1] https://huggingface.co/facebook/vit-mae-base

[2] https://huggingface.co/timm/convnextv2_base.fcmae_ft_in22k_in1k_384

[3] https://github.com/facebookresearch/dino

[4] https://github.com/bytedance/ibot

features a pre-trained self-supervised model using several downstream datasets, including NYUv2 [35]. Although DINOv2 was trained on ImageNet-22k, those weights are not publicly available. We discuss the impact of these curated datasets in App. C. We also consider DINOv2 + reg which incorporates register tokens [7], however, we find that it results in slightly worse performance than the classic DINOv2 model. We evaluate the ViT-B/14 and ViT-L/14 models for DINOv2 and the ViT-B/14 DINOv2 model with registers. We use the checkpoints released by the authors.[5]

DeiT III. Touvron et al. [38] propose an updated training recipe for supervised vision transformers that incorporates recent best practices of self-supervised learning. The result is a much stronger supervised transformer compared to previous training recipes. We evaluate the ViT-B/16 and ViT-L/16 architectures trained on ImageNet-22k. We use the checkpoint released by the authors.[6]

CLIP. Vision and language models are trained to generate aligned feature embeddings using a contrastive objective. The original CLIP family of models was proposed by Radford et al. [29] and included a wide variety of architectures in a private dataset of 400M image-text pairs called WIT. More recently, Ilharco et al. [14] trained several CLIP models using several architectures trained on publicly available datasets. We consider five CLIP checkpoints. First, we evaluated the ViT-B/16 and ViT-L/14 checkpoints released by OpenAI. Second, we evaluate the ViT-B/16 checkpoint released by OpenCLIP [14] that was trained on LAION [34]. [7] Finally, we consider two ConvNeXt-base [22] checkpoints trained with and without additional augmentations [36] that were also released by OpenCLIP. We use these models to try and shed some insight on why the CLIP trained model performs poorly relative to the other models.

SigLIP. SigLIP [46] also learns from images and captions, but instead replaces the contrastive objective with an instance-wise sigmoid loss. The sigmoid loss does not require the computation of all pairs across the batch, as it only relies on the image and text embedding. This simplifies the objective while enabling further scaling up of the batch size for training. The publicly available checkpoints were trained on WebLI; a private dataset. We use the checkpoint available through OpenCLIP [14].[7]

StableDiffusion. StableDiffusion [32] is trained using text-conditioned image generation using a denoising objective. This family of models has achieved remarkable generation performance. In our experiments, we use the text-conditioned checkpoint of StableDiffusion v2-1.[8] Following previous work [44, 48, 49], we extract features from the decoding blocks of the UNet. However, we deviate from previous work in two ways. First, some prior work [37, 47] computes features the image with different sampled noise and then averages them. While this form of ensembling is unique to diffusion-based models, it is possible to compute features based on image crops and similarly average them. To enable fair comparison, we simply computed features once for each image and instead experimented with different noise levels, and we observed that low noise levels appear to work best. In our experiments, we set the noise level to t=1. Second, previous work often computes features using both the image and some auxiliary information; *e.g.*, VPD [49] uses the image class to generate prompts for feature extraction. Such auxiliary information is not used by other models, nor is it available in many settings. Instead, we use an empty vector as a prompt similar to Zhan et al. [47].

MiDaS. MiDaS is a family of models trained on a collection of monocular depth datasets using a scale-invariant depth estimation objective [30]. In this work, we consider MiDaS 3.0 [31] DPT Large which trains a dense prediction transformer (DPT) head on top of the features extracted from a ViT-L/16. While newer iterations of MiDaS 3.1 and ZoeDepth [4] include base-size transformers, we cannot use them due to their reliance on relative positional biases. Specifically, most ViTs rely on absolute learned or heuristic positional encoding, which can be easily interpolated to handle variable image sizes with minimal performance deterioration. However, we find that interpolating relative positional biases severely deteriorates performance. As a result, we instead use MiDaS 3.0 which used absolute positional embeddings. We note that the use of a larger backbone likely exaggerates the performance of MiDaS in our analysis. We use the checkpoint released by the authors.[9]

SAM. Kirillov et al. [16] recently proposed interactive class-agnostic segmentation as a training objective to allow generalizable open-world segmentation. The model is trained on a novel 10M image dataset with 1 billion masks [16]. Although the SAM architecture uses a mask decoder and a prompt encoder, the features are computed by a visual transformer backbone. We use the ViT-B/16 and ViT-L/16 backbones from the SAM base and large models. We use the checkpoint released by the authors.[10]

A.2. Evaluation Datasets

NAVI. NAVI is a dataset of objects annotated with high-quality 3D information that was proposed by Jampani et al. [15]. The dataset depicts a set of 36 objects in a wide range of poses and environments. High-quality object meshes are aligned to each image, which provides accurate depth and pose annotation. We extend the dataset by generating

[5] https://github.com/facebookresearch/dinov2
[6] https://github.com/facebookresearch/deit
[7] https://github.com/mlfoundations/open_clip
[8] https://huggingface.co/stabilityai/stable-diffusion-2-1
[9] https://github.com/isl-org/MiDaS
[10] https://github.com/facebookresearch/segment-anything

surface normal annotations for each image. The dataset is organized into multiview image collections which include a larger number of multiview images of the object in the same pose and scene, as well as a wild set that depicts the object in different poses and environments. We use the full dataset and use multiview images for training and validation, and the wild set for testing. Furthermore, we exclude 2 objects from the dataset as they do not have multiview and wild-set images. For correspondence estimation, we only use the wild set images and for each image, we sample a pair that has a relative rotation between 0 and 120 degrees.

NYU v2. The NYU Depth v2 dataset is a dataset of indoor scenes proposed by Silberman et al. [35]. The dataset consists of RGB-D video collected using a Microsoft Kinect camera and includes dense annotation for both depth and semantic segmentation. Furthermore, Ladickỳ et al. [17] provided surface normal annotations for the labeled set of 1449 images. We use the original train/test split for surface normal estimation. For depth estimation, we also included unlabeled instances, providing us with a total of 24231 images for training.

ScanNet Pairs. ScanNet [6] is a large dataset of RGB-D videos depicting indoor scenes. Sarlin et al. [33] extracted a small subset of 1500 image pairs as a benchmark for correspondence estimation. Since our correspondence estimation experiments do not require training, we use all pairs as a test set.

SPair 71k. SPair-71k [24] is a dataset of image pairs that were extracted from the PASCAL datasets [10, 43]. The image pairs capture a set of 18 categories and depict different object instances of the same class. Furthermore, 8 of the categories depict non-rigid objects; *e.g.*, cats, cows, humans. All images are annotated with class-specific keypoints, and image pairs are further annotated with amount of viewpoint variation as judged by the human annotator. We follow the experimental setup of Zhang et al. [48] of sampling an equal number of image pairs for each class, but instead use a larger number of 200 image pairs per class. We extend this setup by separating the sampled pairs based on the annotated viewpoint variation, which is a subjective measure of how much the viewpoint changed between the two instances and is annotated with 0, 1, or 2.

A.3. Evaluation Tasks

We evaluate all models on four tasks: monocular depth estimation, surface normal estimation, semantic correspodnence estimation, and geometric correspondence estimation. We chose those tasks because they evaluate the model along the two dimensions of 3D awareness we are interested in: single-image 3D understanding and multiview consistency. While we train the models for depth and surface normal estimation, we directly evaluate the features for correspondence. We note that while we use the same setup for evaluating correspondence for NAVI and NYU, SPair follows a different setup due to the existence of keypoints, which we describe separately. We describe each of the tasks and their evaluation procedure below.

A.3.1 Monocular Depth

Task Definition: Given an image, estimate a depth value for each pixel in the image. This problem is ill-posed because it suffers from scale ambiguity; *i.e.*, a larger object that is further away will produce the same image as a smaller object that is closer to the camera. Although the regularity of our environment still allows objects to learn accurate metric depth for specific image collections, such models struggle to generalize to other image collections as different camera intrinsics or image augmentations can introduce effects similar to scale ambiguity [45]. An alternative approach is to predict depth up to scale and then scaling it appropriately.

We use metric depth estimation for NYU due to the regularity of the data and to enable direct comparison to previous work. However, we observe that scale-invariant is more appropriate for NAVI due to the larger variance in cameras as well as a relatively small depth variation in the object surface relative to how far the object is. As a result, we scale the depth for NAVI objects between 0 and 1 for a scale-invariant depth estimation task where 0 means the closest pixel to the camera and 1 means the furthest point on the object from the camera. This variation still enables models to learn accurate depth, as shown in the main paper, and allows us to use standard depth evaluation metrics.

We use the AdaBins [3] parameterization of depth estimation due to its relatively strong performance. Rather than regressing the depth values, Bhat et al. [3] proposes dividing the depth range into several bins and estimating the probability of each. The final depth value is the weighted sum of the bin probabilities and the bin center values. Similar to Oquab et al. [26], we use 256 uniformly distributed bins and only estimate the bin probabilities. We use a depth range of 0-10m for NYU and 0-1 for NAVI.

Losses: We use a combination of the scale-invariant sigmoid depth loss [8] and the gradient matching loss [19] similar to Oquab et al. [26].

Evaluation Metrics: We follow the evaluation setup of Eigen et al. [8] and compute the root mean square error and the prediction accuracy at different thresholds. The accuracy, δ_i, is computed as the number of pixels whose ratio of depth prediction to ground-truth is less than 1.25^i:

$$\delta_i(d^{pr}, d^{gt}) = \frac{1}{N} \sum_{j \in N} \max\left(\frac{d_j^{pr}}{d_j^{gt}}, \frac{d_j^{gt}}{d_j^{pr}}\right) < 1.25^i \quad (1)$$

where d^{pr} is predicted depth and d^{gt} is ground-truth depth.

Probe: We use a non-linear multiscale convolutional probe inspired by the DPT decoder [31]. The probe takes as input multiscale features that are extracted from several stages in the network. Prior work has shown that vision transformer features focus on different objects at different layers [1, 48] and that the granularity is not consistent between models [39, 47]. Instead of using a probe at a single layer, we train a multistage probe on features extracted from several layers. ConvNeXt architectures often group their layers into four stages. We follow this delineation and extract features after every stage. For ViTs, we split the layers into 4 equally sized blocks and extract features after each block; *e.g.*, for ViT-B, this is after layers 3, 6, 9, and 12. For StableDiffusion, the decoding portion of the UNet consists similarly of 4 blocks. We extract the features after each of these blocks. Since prior work has found that the earliest-stage features are often not useful, we only train the model on the latter three stages. This is flipped for StableDiffusion (earlier three stages) as we are sampling from the decoding part of the UNet.

Optimization: We train models for 10 epochs with a linear warm-up of the learning rate for 1.5 epochs and cosine decay to 0. We use the AdamW optimizer [21] with a linear rate of 0.001 and a weight decay of 0.01.

A.3.2 Surface Normals

Task Definition: Given an image, our goal is to estimate the direction of the surface at every pixel. The direction is predicted as a unit norm that is orthogonal to the surface at the point.

Training Loss: The cosine distance is a commonly used loss due to its relative simplicity. However, Bae et al. [2] observe that the model can be severely penalized by areas that are ambiguous. As a result, they propose predicting a fourth value that captures uncertainty and calibrating the loss using that value. The loss uses the estimated uncertainty of the weight loss at each pixel, while encouraging the model to minimize its uncertainty. We use the loss formulation proposed by Bae et al. [2] in our experiments.

Evaluation Metrics: For each pixel, we compute the error as the relative angle between the predicted and ground-truth surface normals in degrees. Similar to depth estimation, we compute the RMSE for each image as well as the accuracy at different thresholds. However, instead of using the ratio as done in depth, we simply compute the accuracy at different angular thresholds ($11.25°$, $22.5°$, $30°$) similar to previous work [2, 11, 27].

Probe: We use the same probe as depth estimation with the main difference of the final layer output dimensionality being 4 instead of 256. The four values correspond to the x-, y-, and z-components of the surface normal direction the uncertainty value used in the loss computation. We normalize the 3 directional components to a unit normal.

Optimization: The optimization procedure is identical to that used for depth estimation.

A.3.3 Geometric Correspondence

Task Definition: Given two images that depict the same object or scene from different viewpoints, the goal is to identify pairs of pixels in images that depict the same 3D point in space. We consider two settings: object-centric and scene-centric. For the scene-centric evaluation, we allow correspondences to be computed for all pixels across the images. However, for object-centric evaluation, we only consider pixels that lie on the object mask.

Inference Procedure: Given two images, we first extract a feature map for each image. We then estimate the correspondence using the nearest neighbors in the feature space. This provides us with the correspondence for each pixel, many of which will be inaccurate. We filter the correspondences using Lowe's ratio test [23] which aims to find unique matches by discounting points that have more than one strong correspondence. For each point p, we find its first and second nearest neighbors: q_0 and q_1. We then compute the ratio r as follows:

$$r = 1 - \frac{D(p, q_0)}{D(p, q_1)} \tag{2}$$

where $D(x, y)$ is the cosine distance between x and y. We rank the correspondences using the ratio test and keep the top 1000 correspondences.

Evaluation: Correspondences are evaluated based on either 2D projection error or 3D metric error. Given an estimated correspondence between pixel locations p in image 1 and q in image 2, the 2D projection distance is computed by first projecting the point p into the 3D space using known depth and intrinsics and then projecting it into image 2 using known intrinsics of the camera and the relative viewpoint between the two images. This allows us to find the actual location of the point p when projected in image 2: p'. The 2D correspondence error can be computed as the distance between p' and q in the image plane. This works very well for scenes, but can be problematic for objects where points that are not invisible can still be projected into the image plane. While it is possible to omit surface points that are not visible, the approach ignores a lot of points on thin structures; *e.g.*, points on a wire. Instead, we can simply compute the 3D correspondence error by focusing both points p and q in a shared 3D space and computing the distance between them. We use the 2D projection error for scenes and the 3D error for objects.

We evaluate performance using correspondene recall; *i.e.*, the percentage of correspondences whose error is below a specific threshold. Since we are interested in the consistency of representation, we split the image pairs based on the viewpoint change between them where θ_i^j means the error for image pairs whose relative viewpoint angle is between i and j degrees. One thing to note is that while two views with a relative angle of 180° depict the opposite side of the object with no mutually visible surfaces, a room viewed from the opposite corner has a relative viewpoint change of 180° with a large portion of the images being mutually visible. Hence, while increasing relative viewpoints imply increasing difficult, the numbers are not directly comparable, as one is viewing the scene from inside of it, but viewing the object from its outside.

A.3.4 Semantic Correspondence

Task Definition: Given two images and a set of semantic keypoints in image one, the goal is to find the location of the pixel belonging to those key points in the second image. Key points are often semantic parts; *e.g.*, a cat's left ear or the front right wheel of the car. Unlike the previous task where one has to find a set of points in both images that match each other, the set of points in the first image are already specified. Furthermore, while the previous task is matching points belonging to the same scene or the same object instance, semantic keypoints are defined at the class level so the images often depict two different instance; *e.g.*, two different cats or two different cars.

Inference Procedure: We follow previous work [37, 48] and simply use nearest neighbors. There is no need for filtering, as the goal is to just find the point in the second image that is most similar to the keypoint.

Evaluation Metrics: The evaluation is often based on the percentage of keypoints within a pixel threshold; *i.e.*, the percentage of predicted keypoints within N pixels of the ground-truth match. The evaluation is based on the assumption that each keypoint has a single valid match in the second image. This results in each evaluation only considering the predicted keypoint and its ground-truth location, and ignoring everything else. We evaluate all models using this procedure in App. B, but we also consider an alternative evaluation as discussed in the main paper.

An alternative way to evaluate the prediction is to compare to all keypoints in that image. Instead of asking how close the prediction is to the ground truth for the same keypoint type, we can ask which ground-truth keypoint is closest to the prediction. This allows us to understand which keypoints are getting confused with each other, rather than how many keypoints are correctly classified. This is important since the threshold is usually 10-20% of the size of the bounding box, which can include several different keypoints.

Previous work reports the average performance for all pairs. Instead, we separate the performance for image pairs of different viewpoint changes. Specifically, we use the annotation of viewpoint variation provided by SPair [24] and report the performance for different viewpoint difficulties.

A.4. Performance Correlation

One question tackled in our paper is how well performance is correlated across tasks. If several tasks are measuring the same capability, we would expect their performance to be well correlated. Although a high correlation could be caused by a variety of other factors. Hence, while a high correlation provides some evidence that tasks measure the same capability, a very low correlation would imply that tasks are not related.

We compute the correlation of model performance across different tasks and task domains. Specifically, we compute the Pearson correlation coefficient, which assumes a linear relationship between models. One possibility is that the relationships across tasks may not be linear and that a rank correlation might be well-suited. Empirically, we observe that both statistical measures often result in similar trends with some minor exceptions. When considering cases where the correlations deviate from each other, we find that they are caused by fluctuations in the rankings that arise for very small changes in performance for similarly performing models. As a result, we choose to report the Pearson correlation coefficient as a descriptive statistic of the relationships between model performance.

When considering the overall model performance, such as Figure 1, we aggregate performance across all tasks. Since the absolute performance values are not directly comparable, we instead rely on the model rankings. While a model's ranking can fluctuate due to minor differences, such fluctuations tend to get canceled out when averaging the ranking across multiple tasks. We convert the rankings to a normalize rating where a 1 means the best performing model and 0 means the worst performance model. The overall ratings are shown in Figure 1. We emphasize that such ratings represent relative, not absolute, model performance. Hence, a rating of 1 does not mean the representations are 3D aware, but rather that they are more 3D aware than the other models considered.

B. Additional Results

We chose to focus on general performance trends and salient comparisons in the main body of the paper. Here, we report the complete results for all the models and tasks considered in Tables 1 to 5. Additionally, we attempt to provide some targeted comaprisons that shed some light on some of the findings discussed in the main paper, as well as address some of the confounders introduced by using publicly available checkpoints.

What explains the low performance of vision and language models? One interesting finding is that vision and language models perform poorly across all tasks. Although previous work has shown that CLIP struggles with 2D spatial relationships [18, 20], the disparity in performance that we observe is quite surprising. This makes it difficult to determine what causes the performance disparity. Below, we consider several possibilities: training data, training objective, model architecture, and augmentations.

One possibility is that CLIP's WIT dataset does not capture such relationships. However, we note that the OpenCLIP model trained on LAION and the SigLIP model trained on WebLI both achieve very similar performance as seen in Tabs. 1 to 5. Furthermore, StableDiffusion achieves a very strong performance despite also being trained on LAION.

Another possibility is that this is caused by the training objective; *i.e.*, the contrastive objective discourages such relationships. Our experiments suggest that the training objective is likely the major culprit, but it remains unclear what about the objective is causing this issue. It is unclear whether semantics inhibits spatial understanding since both DeiT and StableDiffusion achieve a strong performance despite learning from some form of semantics. Furthermore, it is not the discriminative aspect of the learning objective, as DeiT and DINO are both trained with discriminative objects and also perform well. On the other hand, SigLIP learns from a non-contrastive objective and performs poorly.

We find that the greatest improvement comes from changing the backbone from ViT to ConvNeXt. This change results in a qualitative change in CLIP's performance; *e.g.*, from predicting flat surfaces for depth to generating something that looks like a depth map. We note that this is not because ConvNeXt is a strictly superior architecture; *e.g.*, we find that DeiT's ViT performs better than ConvNeXt for supervised training on ImageNet-22k.

One final difference is the use of augmenations during training. Most of the approaches considered rely on augmentation to create a learning signal such as self-supervised learning or to provide additional diversity in the training data. However, CLIP relies on minimal augmentation, and we are unaware of any transformer-based checkpoints that train with augmentations. Instead, we consider two CLIP ConvNeXt checkpoints that were trained with and without AugReg [36]. We find that augmentation boosts performance, but the gains are relatively small. It remains unclear if such augmentations would be more impactful when used with ViT backbones.

In conclusion, we find that none of the factors individually explain the low performance of vision language models, although choice of architecture appears to be a useful direction to further puruse. Our findings also suggest that the training signal or model architecture does not independently determine the 3D awareness of features. However, more controlled experiments are needed to confirm this claim.

What is the impact of model scale? While our experiments focused on ViT-B backbones, we also evaluated the large checkpoints and report their performance here. Although using a larger backbone typically increases performance, the improvement is fairly marginal. Furthermore, we find that the reported patterns and correlations do not change when we consider the larger checkpoints.

What is the impact of model architecture? We choose a subset of our models and compare their performance when trained with ViT or ConvNeXt backbones. While ConvNeXts result in a huge performance gain for CLIP models, the results are mixed for other models; *e.g.*, FCMAE and DeIT. As noted earlier, our experiments suggest that model architecture and training objective are not independent when it comes to 3D awareness. However, more research is needed to further explore this relationship.

What is the impact of training dataset? While dataset scale appears to impact performance, the results are often mixed or marginal. Our main data point here is to compare models trained on ImageNet-1k and ImageNet-22k. Beyond scale, dataset curation appears to have a significant impact on performance as reported by Oquab et al. [26]. Despite these confounders, we find that the variation in performance due to different datasets is smaller than the variation caused by training objectives. Nevertheless, we hope that future work can shed some light on the impact of such datasets, and more importantly, on what properties of the dataset encourage or inhibit 3D awareness.

Additional SPair Confusion Matrices. In the main body of the paper, we visualize the confusion matrices for the chair class under different viewpoint variations. The chair class is most representative of this distinction for two reasons: (1) its keypoints neatly segment into semantic groups that only differ based on their relative location in the chair's canonical frame of references; and (2) semantically similar keypoints can be visible in the same image in different relative orientations for each other. Many other classes do not fulfill those criteria, especially the second point. For example, several keypoints for humans and animals are unique; *e.g.*, mouth, nose, tail, forehead. In other cases, semantically related keypoints almost always appear in the same 2D configuration. For example, eyes and ears often appear in the same 2D configuration when they are both visible. As a result, their 2D relative locations are very strongly correlated with their identity, making it difficult to assess the 3D awareness of the model. This is confounded by common photographic biases in datasets; *e.g.*, most human and naimal faces are often pictured facing the camera. Finally, many symmetric keypoints are almost never covisible. It is very rare for both front left and front right wheels to be covisible in the same image. Most car pictures featuring more than 1 visible wheel are side views. Hence, for most pairs of car views, a combi-

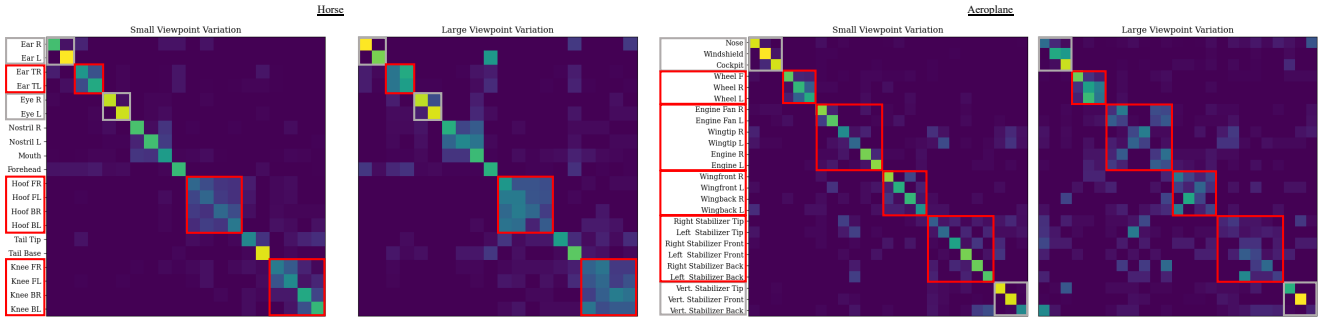

Figure 1. **Keypoint confusion matrices for horses and aeroplanes.** We find similar confusion patters in other SPair classes where large viewpoint changes result in high confusion between semantically related classes (highlighted in red). Furthermore, we find that keypoints that experience a lot of deformation (*e.g.*, knees and hooves) are confused for all image pairs as they appear in different 2D locatons relative to each other. However, we find that classes (highlighted in grey) that often appear in the same 2D configuration do not suffer from this effect.

nation of coarse semantic class (*i.e.*, car wheel) and relative location in the image can result in accurate correspondence.

We observe these findings in the data. We present the confusion matrices StableDiffusion features on two additional classes: horses and airplanes. The results are shown in Fig. 1. First, we find that unique classes, such as tail tip and tail base, have a similar accuracy across viewpoint variations. Furthermore, classes that appear in similar orientations, such as eyes or ears, are similarly unaffected. We note that the ear annotation refers to the front of the ear for horses. Meanwhile, the top of the ear, which is also visible when the horse is looking away, exhibits a different behavior where it is far more confused when for larger viewpoint changes.

We observe similar patterns for airplanes where unique keypoints such as the nose, windshield, and cockpit of the airplane are accurately predicted for large viewpoint changes. Meanwhile, the wheels, wings, and stabilizers are confused for larger viewpoint changes. To support that this is not simply due to some keypoints being more difficult, one can compare the performance for vertical stabilizers with the performance for right and left stabilizers. While the right and left stabilizers are confused, the vertical stabilizer, which looks the same but has a very different orientation, is still accurately predicted for larger viewpoint changes.

Finally, we observe interesting trends for knees and hooves that appear to be confused regardless of viewpoint change. Since animals can deform, hooves and legs consistently appear in different orientations, especially for horses, which are often pictured while moving. Nevertheless, the confusion is strongly restricted to the semantically equivalent classes. This strongly suggests that while the model understands semantics, it lacks 3D awareness.

C. Limitations

Our goal is to understand to what extent current large-scale models "understand" the 3D world that images depict, as well as what factors encourage or discourage such understanding from emerging. This is very challenging, as our collective understanding of what models learn or how they represent what they learn is still very limited. Furthermore, there is no consensus on how 3D geometry should be represented, nor what it even means for a model to have 3D understanding. Finally, there are concrete challenges in evaluation that make it difficult to conduct controlled experiments. The study presented in this work is a first step toward answering these questions. While our experiments and analysis provide some initial answers to this question, our study suffers from several limitations that limit the strength of the conclusions we can draw and point to several avenues for future exploration. We discuss the limitations below and outline some open questions that we hope future work will address.

Comparisons are limited to publicly available checkpoints. We focused on our analysis on publicly available checkpoints due to their availability and common use in the literature. However, as a result, our experiments often compare models trained with different recipes on different datasets. This is a significant confounding variable, as it is unclear if the trends we are observing are due to the main differentiators we observe or some minor implementation detail.

Ideally, we would train the same backbone architecture on the same dataset with the same training recipe, but with different objectives as in [9, 25]. However, the computational resources needed to train and tune all models are prohibitively large. This problem is also likely to be exacerbated with the current trend of moving towards ever larger scales. Beyond the resources required, different approaches often have different data requirements such as curation, labels, captions, or dense annotations. There are currently no large-scale datasets that meet all those requirements.

We tried to make comparisons more fair by choosing model checkpoints of comparable model capacity and pre-

training data scale. Although we expected that the dataset or pre-training scale might end up dominating all other effects, our experiments suggest that other factors can be more important. For example, while CLIP is trained on a much larger dataset than DINO, DINO consistently outperforms CLIP. Furthermore, model performance does not seem to be very sensitive to training data, with CLIP achieving similar performance whether it was trained on WIT or LAION. We hope that our analysis identified interesting patterns and that future work can conduct much more controlled experiments that focus on a specific model comparison or dataset comparison.

Our analysis focuses on two specific aspects of 3D awareness. Our ability to perceive and infer 3D properties is remarkable. While many tasks can showcase this ability, we restrict our analysis to single-image surface reconstruction and multiview consistency. While those two aspects are fundamental to 3D understanding, they are not comprehensive. The ability to reconstruct the full 3D shape, predict deformation, and estimate physical properties such as support and containment fall under the general umbrella of 3D understanding. However, it is unclear which of those properties should be readily perceived from the image as opposed to inferred with more complex processing. While such delineations can be more philosophical in nature, they can guide the experimental design, as it is important to understand what we are looking for before designing experiments to find it. We expect that more comprehensive benchmarks of 3D understanding would measure such capabilities as well, and we hope that this work provides an initial step towards the study of this problem.

Our experimental methodology focuses on probing methods. Our analysis has focused on linear probe and zero-shot analysis approaches. We have done this to analyze the features as they are without changing them to better adapt to 3D tasks. While we argue that frozen features provide a more accurate understanding of the 3D awareness of the features, it would definitely be useful to understand how much of those patterns transfer to fine-tuning setups. Furthermore, if we consider recent advances in natural language processing, we see a rise in in-context learning with similar adaptations in computer vision. While linear probes could still be applied, it is likely that prompting-based methods will be more suited to analyze the 3D awareness of such models.

Table 1. **Depth Estimation Results.** We present the depth estimation results for all models. Models are grouped based on the supervisory signal.

			NYU				NAVI			
Model	**Architecture**	**Dataset**	δ_1	δ_2	δ_3	RMSE	δ_1	δ_2	δ_3	RMSE
Self-Supervised Models										
MAE [13]	ViT-B16	IN-1k	67.89	91.43	97.92	0.6602	36.28	63.44	79.72	0.1568
MAE [13]	ViT-L16	IN-1k	70.22	92.50	98.00	0.6298	35.21	62.60	79.41	0.1588
FCMAE [42]	CNXTv2-B	IN-22k	82.73	97.63	99.67	0.4860	48.61	76.15	88.44	0.1205
DINO [5]	ViT-B16	IN-1k	80.44	96.52	99.26	0.5071	56.62	81.26	91.01	0.1043
iBOT [50]	ViT-B16	IN-1k	83.87	97.47	99.51	0.4635	57.11	82.00	91.49	0.1025
iBOT [50]	ViT-B16	IN-22k	85.04	97.82	99.55	0.4452	55.75	81.06	91.14	0.1044
iBOT [50]	ViT-L16	IN-1k	85.14	97.74	99.56	0.4451	62.56	85.27	93.19	0.0919
iBOT [50]	ViT-L16	IN-22k	91.08	98.98	99.79	0.3670	66.29	87.31	94.27	0.0833
DINOv2 [26]	ViT-B14	LVD	93.43	99.41	99.90	0.3307	68.84	89.13	95.31	0.0763
DINOv2 [26]	ViT-L14	LVD	94.84	99.57	99.93	0.3086	71.42	90.34	95.91	0.0721
DINOv2 + reg [7]	ViT-B14	LVD	92.93	99.37	99.91	0.3355	66.56	87.94	94.74	0.0806
Classification-Supervised Models										
DeiT III [38]	ViT-B16	IN-22k	86.89	98.20	99.73	0.4240	59.74	83.78	92.61	0.0960
DeiT III [38]	ViT-L16	IN-22k	89.55	98.75	99.78	0.3866	64.93	86.79	93.98	0.0855
ConvNeXt [22]	CNXT-B	IN-22k	80.15	96.88	99.46	0.5105	43.63	71.60	85.49	0.1342
Vision Language Models										
CLIP[28]	ViT-B16	WIT	52.11	81.70	93.72	0.9450	24.95	48.73	68.52	0.1993
CLIP[28]	ViT-L14	WIT	51.73	81.59	93.85	0.9445	23.97	46.71	66.34	0.2058
CLIP[14]	ViT-B16	LAION	52.46	82.15	94.05	0.9351	24.33	47.43	67.31	0.2031
CLIP[14]	CNXT-B	LAION	78.10	96.26	99.33	0.5285	45.24	73.35	86.68	0.1286
CLIP + AugReg[14]	CNXT-B	LAION	82.14	97.24	99.48	0.4784	48.22	75.85	88.09	0.1211
SigLIP [46]	ViT-B16	WebLI	63.81	89.72	97.33	0.7187	36.49	63.07	79.20	0.1571
SigLIP [46]	ViT-L16	WebLI	65.26	90.53	97.78	0.6931	35.94	62.43	78.78	0.1588
Text-Conditioned Image Generation Models										
StableDiffusion [32]	UNet	LAION	82.60	97.05	99.48	0.4801	51.86	78.68	89.77	0.1094
Densely-Supervised Models										
SAM [16]	ViT-B16	SA-1B	75.60	95.00	98.94	0.5665	49.28	76.36	88.40	0.1206
SAM [16]	ViT-L16	SA-1B	81.57	97.12	99.49	0.4905	52.45	79.17	90.25	0.1105
MiDaS [31]	ViT-L16	MIX 6	78.65	96.05	98.99	0.5300	58.53	82.58	91.60	0.1001

Table 2. **Surface Normal Estimation Results.** We present the surface normal estimation results for all models. Models are grouped based on the supervisory signal.

			NYU				NAVI			
Model	**Architecture**	**Dataset**	11.25°	22.5°	30°	RMSE	11.25°	22.5°	30°	RMSE
Self-Supervised Models										
MAE [13]	ViT-B16	IN-1k	44.60	66.08	74.37	30.34	30.27	57.70	69.97	32.12
MAE [13]	ViT-L16	IN-1k	45.60	67.50	75.67	29.57	30.36	58.20	70.54	31.59
FCMAE [42]	CNXTv2-B	IN-22k	43.87	68.73	78.05	27.24	28.91	58.06	70.76	31.54
DINO [5]	ViT-B16	IN-1k	49.11	69.32	77.02	28.35	39.56	65.85	76.23	28.75
iBOT [50]	ViT-B16	IN-1k	52.60	72.52	79.60	26.89	40.65	66.79	76.99	28.32
iBOT [50]	ViT-B16	IN-22k	54.54	73.79	80.52	26.10	22.75	50.60	64.10	35.03
iBOT [50]	ViT-L16	IN-1k	54.53	74.30	81.06	25.78	43.45	69.20	78.84	27.24
iBOT [50]	ViT-L16	IN-22k	58.51	76.68	82.90	24.46	45.19	70.51	79.84	26.56
DINOv2 [26]	ViT-B14	LVD	62.01	79.32	85.32	22.41	47.55	72.92	81.89	25.27
DINOv2 [26]	ViT-L14	LVD	64.05	80.78	86.48	21.55	50.15	74.70	83.12	24.52
DINOv2 + reg [7]	ViT-B14	LVD	61.37	79.12	85.22	22.54	45.81	72.00	81.28	25.66
Classification-Supervised Models										
DeiT III [38]	ViT-B16	IN-22k	39.58	64.22	74.07	29.44	26.91	55.66	68.71	32.47
DeiT III [38]	ViT-L16	IN-22k	55.73	75.34	82.00	25.05	32.91	61.54	73.49	29.92
ConvNeXt [22]	CNXT-B	IN-22k	25.79	47.42	59.15	35.40	21.72	50.54	64.75	34.34
Vision Language Models										
CLIP[28]	ViT-B16	WIT	28.85	51.93	63.07	35.34	17.01	42.06	56.10	39.58
CLIP[28]	ViT-L14	WIT	28.07	50.74	61.97	35.84	15.03	38.87	52.93	41.41
CLIP[14]	ViT-B16	LAION	30.11	53.65	64.60	34.68	16.23	40.95	55.07	39.90
CLIP[14]	CNXT-B	LAION	43.54	67.93	77.06	27.88	31.28	59.41	71.60	31.08
CLIP + AugReg[14]	CNXT-B	LAION	45.50	69.53	78.43	27.07	34.17	62.29	73.87	29.82
SigLIP [46]	ViT-B16	WebLI	30.68	52.73	63.60	34.96	21.47	47.60	60.93	36.69
SigLIP [46]	ViT-L16	WebLI	31.68	54.11	64.97	34.20	21.24	46.83	60.19	37.01
Text-Conditioned Image Generation Models										
StableDiffusion [32]	UNet	LAION	58.29	76.28	82.64	24.68	40.31	67.18	77.55	27.86
Densely-Supervised Models										
SAM [16]	ViT-B16	SA-1B	46.37	70.40	78.98	26.89	36.66	64.25	75.36	29.01
SAM [16]	ViT-L16	SA-1B	52.03	74.61	82.20	24.87	39.23	66.87	77.59	27.66
MiDaS [31]	ViT-L16	MIX 6	49.58	69.82	77.17	28.51	39.94	66.36	76.66	28.54

Table 3. **Correspondence Estimation Results for ScanNet.** We present the ScanNet correspondence estimation results for all models. The results are presented for features extracted at different layers with performance binned for different relative viewpoint changes between image pairs. The highest performing set of results for each model are highlighted and bolded.

			Block$_0$				Block$_1$				Block$_2$				Block$_3$			
Model	**Architecture**	**Dataset**	θ_0^{15}	θ_{15}^{30}	θ_{30}^{60}	θ_{60}^{180}	θ_0^{15}	θ_{15}^{30}	θ_{30}^{60}	θ_{60}^{180}	θ_0^{15}	θ_{15}^{30}	θ_{30}^{60}	θ_{60}^{180}	θ_0^{15}	θ_{15}^{30}	θ_{30}^{60}	θ_{60}^{180}
Self-Supervised Models																		
MAE [13]	ViT-B16	IN-1k	3.4	2.8	3.8	2.3	4.7	3.6	3.9	2.6	7.8	5.5	4.9	3.0	12.5	8.0	6.0	3.6
MAE [13]	ViT-L16	IN-1k	5.2	3.8	4.1	2.6	8.2	6.3	5.2	3.1	14.1	10.1	7.0	4.1	15.6	10.9	7.5	4.3
FCMAE [42]	CNXTv2-B	IN-22k	26.2	19.8	12.1	5.5	60.8	49.7	28.1	9.8	31.6	22.4	13.2	5.8	36.8	26.2	14.9	7.2
DINO [5]	ViT-B16	IN-1k	14.3	9.5	7.9	3.9	42.8	32.4	21.0	9.1	44.5	34.1	22.2	9.8	45.0	34.3	22.6	10.7
iBOT [50]	ViT-B16	IN-1k	19.3	13.0	9.5	4.1	37.2	26.6	17.6	7.8	39.0	28.7	18.9	8.8	37.8	27.4	18.4	9.1
iBOT [50]	ViT-B16	IN-22k	14.2	10.1	7.8	3.6	27.9	19.2	12.6	6.2	36.0	25.8	16.5	8.0	30.1	20.7	13.9	7.0
iBOT [50]	ViT-L16	IN-1k	30.3	20.7	14.0	5.8	43.8	32.8	21.0	9.1	44.7	33.8	22.0	10.1	46.0	34.6	22.8	10.8
iBOT [50]	ViT-L16	IN-22k	27.9	20.0	13.1	5.5	42.2	31.1	19.9	8.8	41.3	30.5	19.6	9.3	40.4	29.9	20.3	10.2
DINOv2 [26]	ViT-B14	LVD	25.4	19.5	12.7	4.9	47.1	36.4	22.4	8.4	37.6	26.8	16.8	7.5	37.0	27.5	19.7	11.2
DINOv2 [26]	ViT-L14	LVD	12.6	10.4	7.8	4.0	27.6	19.4	12.5	4.7	34.7	23.7	15.8	6.4	36.4	26.8	20.4	12.1
DINOv2 + reg [7]	ViT-B14	LVD	29.0	24.6	15.1	5.8	56.1	47.3	29.5	10.3	48.4	37.8	24.2	10.0	41.9	33.6	23.2	12.2
Classification-Supervised Models																		
DeiT III [38]	ViT-B16	IN-22k	17.6	12.1	8.8	3.4	38.3	29.7	17.9	7.2	28.5	21.3	14.0	6.6	20.7	13.7	9.2	5.0
DeiT III [38]	ViT-L16	IN-22k	34.3	26.8	16.6	5.4	36.4	27.8	16.7	7.0	26.5	19.4	12.7	6.3	28.3	20.6	14.0	7.5
ConvNeXt [22]	CNXT-B	IN-22k	22.8	17.1	9.9	4.7	42.9	32.9	18.1	6.9	10.2	6.7	3.9	2.6	15.0	10.6	5.7	3.6
Vision Language Models																		
CLIP[28]	ViT-B16	WIT	10.5	8.3	6.5	4.4	3.7	3.3	3.8	2.6	3.0	2.4	3.1	2.1	2.5	2.1	2.7	1.8
CLIP[28]	ViT-L14	WIT	8.4	7.3	5.8	3.9	4.8	4.1	4.3	2.9	3.9	3.3	3.6	2.6	3.4	2.9	3.0	2.5
CLIP[14]	ViT-B16	LAION	15.5	11.6	8.3	4.6	5.8	4.8	4.7	3.2	2.5	2.3	3.0	2.1	2.5	2.1	2.7	2.0
CLIP[14]	CNXT-B	LAION	26.8	20.8	13.3	6.6	30.3	21.7	12.1	5.3	47.1	38.1	22.3	8.9	37.1	28.5	16.6	8.0
CLIP + AugReg[14]	CNXT-B	LAION	24.2	18.9	12.4	6.3	37.7	27.3	14.5	6.0	32.3	22.9	13.0	5.4	31.6	22.8	13.1	7.1
SigLIP [46]	ViT-B16	WebLI	17.0	12.5	9.2	5.9	16.6	12.0	8.9	5.5	15.8	11.3	8.6	5.4	14.2	10.4	8.1	5.3
SigLIP [46]	ViT-L16	WebLI	15.4	11.1	8.1	4.9	14.4	10.2	7.3	4.6	13.4	9.3	6.7	4.2	11.7	8.5	6.4	4.2
Text-Conditioned Image Generation Models																		
StableDiffusion [32]	UNet	LAION	10.2	5.0	3.0	1.4	66.4	55.0	31.0	8.4	60.4	48.3	27.4	9.1	14.4	10.9	7.7	4.5
Densely-Supervised Models																		
SAM [16]	ViT-B16	SA-1B	8.3	6.0	5.6	2.9	35.0	26.2	17.4	5.3	52.9	44.7	29.3	8.9	55.3	47.0	30.4	9.4
SAM [16]	ViT-L16	SA-1B	14.5	9.9	7.5	3.5	37.2	29.7	19.7	6.2	47.6	40.4	27.3	8.7	52.6	43.9	28.7	9.6
MiDaS [31]	ViT-L16	MIX 6	50.3	39.0	24.4	11.2	56.4	47.4	31.6	13.9	55.5	46.0	30.8	14.3	52.4	42.1	27.6	13.1

Table 4. **Correspondence Estimation Results for NAVI.** We present the NAVI correspondence estimation results for all models. The results are presented for features extracted at different layers with performance binned for different relative viewpoint changes between image pairs. The highest performing set of results for each model are highlighted and bolded.

			Block$_0$				Block$_1$				Block$_2$				Block$_3$			
Model	**Architecture**	**Dataset**	θ_0^{30}	θ_{30}^{60}	θ_{60}^{90}	θ_{90}^{120}	θ_0^{30}	θ_{30}^{60}	θ_{60}^{90}	θ_{90}^{120}	θ_0^{30}	θ_{30}^{60}	θ_{60}^{90}	θ_{90}^{120}	θ_0^{30}	θ_{30}^{60}	θ_{60}^{90}	θ_{90}^{120}
Self-Supervised Models																		
MAE [13]	ViT-B16	IN-1k	73.1	40.9	19.2	10.7	75.8	42.3	19.8	10.8	76.9	43.1	20.3	11.1	76.3	43.0	20.2	11.1
MAE [13]	ViT-L16	IN-1k	68.5	38.7	19.3	11.2	73.2	40.7	19.8	11.0	74.0	40.9	19.8	10.9	73.3	40.3	19.4	10.7
FCMAE [42]	CNXTv2-B	IN-22k	37.9	24.7	16.3	11.0	66.2	37.5	19.5	11.6	74.0	48.4	28.6	18.1	82.8	59.5	39.6	26.7
DINO [5]	ViT-B16	IN-1k	85.7	50.2	22.7	11.9	89.3	58.1	30.1	18.3	87.2	57.4	31.7	20.6	86.0	56.0	31.3	20.3
iBOT [50]	ViT-B16	IN-1k	85.3	49.3	22.2	11.7	90.3	57.1	27.9	16.1	89.4	59.1	31.6	20.2	89.0	58.3	32.5	22.2
iBOT [50]	ViT-B16	IN-22k	84.3	47.6	21.5	11.3	90.3	55.4	25.7	15.1	89.9	59.6	31.8	20.0	88.7	57.7	31.2	20.8
iBOT [50]	ViT-L16	IN-1k	90.2	56.2	26.3	14.4	92.2	63.0	32.7	20.4	91.4	63.3	34.6	22.8	90.3	63.6	36.7	25.4
iBOT [50]	ViT-L16	IN-22k	89.3	54.1	24.1	12.6	93.1	65.5	34.0	20.9	92.2	66.2	36.7	23.9	89.5	64.5	39.0	27.0
DINOv2 [26]	ViT-B14	LVD	79.4	46.1	22.2	12.3	93.6	63.7	29.4	14.9	94.4	68.8	36.8	20.5	90.6	69.3	45.9	32.0
DINOv2 [26]	ViT-L14	LVD	66.2	37.2	19.6	11.5	92.1	57.9	25.6	12.8	95.3	70.0	35.4	18.5	92.2	72.3	48.9	35.0
DINOv2 + reg [7]	ViT-B14	LVD	73.0	41.9	21.0	12.0	92.6	62.0	28.7	14.6	94.4	70.0	37.9	21.3	89.0	67.3	44.8	31.1
Classification-Supervised Models																		
DeiT III [38]	ViT-B16	IN-22k	88.6	51.8	22.9	12.1	91.5	62.8	34.3	22.0	84.3	58.0	37.9	26.7	62.7	38.5	24.6	16.7
DeiT III [38]	ViT-L16	IN-22k	88.7	54.2	24.5	13.3	92.3	65.5	36.9	24.1	86.1	60.6	39.2	27.4	76.8	50.8	32.9	22.5
ConvNeXt [22]	CNXT-B	IN-22k	39.1	24.3	15.6	10.2	49.2	25.8	14.8	9.4	75.8	46.1	26.6	16.6	80.3	52.2	31.5	20.3
Vision Language Models																		
CLIP[28]	ViT-B16	WIT	42.3	26.2	16.2	10.7	34.8	22.8	14.6	10.0	25.7	17.9	12.9	8.9	22.1	15.6	11.4	7.9
CLIP[28]	ViT-L14	WIT	36.3	22.6	14.8	10.1	28.1	18.9	13.6	9.4	21.3	15.7	12.1	8.6	17.9	13.6	10.9	7.7
CLIP[14]	ViT-B16	LAION	41.8	25.2	15.8	10.5	36.5	22.6	14.5	9.7	26.0	17.1	12.2	8.6	21.6	15.3	11.0	7.9
CLIP[14]	CNXT-B	LAION	34.1	22.9	15.5	10.9	45.0	27.6	17.1	11.1	84.7	56.9	32.2	19.6	76.7	47.1	30.0	20.5
CLIP + AugReg[14]	CNXT-B	LAION	34.4	23.0	15.1	10.4	48.1	28.0	16.7	10.6	84.5	54.4	30.7	18.1	78.9	50.1	31.6	21.4
SigLIP [46]	ViT-B16	WebLI	40.6	26.7	18.6	12.7	41.7	27.5	18.6	12.6	40.5	26.5	18.3	12.3	35.3	23.2	16.7	11.4
SigLIP [46]	ViT-L16	WebLI	39.0	25.8	17.5	11.8	39.7	26.1	17.6	11.9	36.6	24.2	16.6	11.4	29.9	19.8	14.9	10.2
Text-Conditioned Image Generation Models																		
StableDiffusion [32]	UNet	LAION	77.4	36.4	14.8	7.4	91.2	58.9	25.7	11.1	70.8	41.1	20.3	11.5	33.9	21.9	14.7	9.7
Densely-Supervised Models																		
SAM [16]	ViT-B16	SA-1B	77.8	42.7	19.9	11.4	83.0	48.4	22.0	11.9	88.6	56.2	25.0	12.9	88.2	56.5	25.3	12.7
SAM [16]	ViT-L16	SA-1B	78.0	43.3	20.4	11.4	86.4	52.0	23.8	12.5	91.2	60.1	28.2	14.2	88.5	57.6	26.9	13.5
MiDaS [31]	ViT-L16	MIX 6	79.0	49.1	25.0	14.5	83.2	56.0	32.1	21.6	82.2	56.3	33.1	22.9	79.6	53.0	31.4	21.6

Table 5. **Correspondence Estimation Results for SPair-71k.** We present the SPair-71k correspondence estimation results for all models. The results are presented for features extracted at different layers with performance binned based on the viewpoint variation for the image pair. The highest performing set of results for each model are highlighted and bolded.

			Block_0				Block_1				Block_2				Block_3			
Model	**Architecture**	**Dataset**	d=0	d=1	d=2	all	d=0	d=1	d=2	all	d=0	d=1	d=2	all	d=0	d=1	d=2	all
Self-Supervised Models																		
MAE [13]	ViT-B16	IN-1k	8.3	4.7	3.8	6.8	9.9	6.3	5.1	7.9	10.5	6.6	5.5	8.5	9.8	6.2	4.9	7.9
MAE [13]	ViT-L16	IN-1k	8.3	5.5	4.6	6.9	9.1	6.5	5.1	7.6	9.3	6.4	5.2	7.7	8.3	5.8	4.5	7.2
FCMAE [42]	CNXTv2-B	IN-22k	5.3	4.7	4.9	5.0	8.3	6.7	6.0	7.3	26.4	24.3	25.4	24.8	28.9	27.1	28.2	27.5
DINO [5]	ViT-B16	IN-1k	15.3	8.2	6.4	11.6	26.7	17.9	18.6	21.9	32.1	25.1	25.9	28.3	30.4	24.0	24.3	26.8
iBOT [50]	ViT-B16	IN-1k	14.3	7.3	5.5	10.4	25.3	14.6	14.0	20.0	35.5	25.3	25.5	30.6	39.9	30.3	32.3	35.7
iBOT [50]	ViT-B16	IN-22k	12.7	6.6	4.5	9.6	22.0	12.5	10.6	17.0	36.0	25.0	24.6	30.6	41.9	29.6	31.1	36.1
iBOT [50]	ViT-L16	IN-1k	22.4	12.0	10.0	17.1	39.3	25.6	26.1	32.8	44.9	32.8	33.7	39.3	48.9	36.2	38.5	43.2
iBOT [50]	ViT-L16	IN-22k	19.4	9.9	7.8	14.7	39.6	23.2	23.5	31.6	46.8	32.2	33.6	40.0	51.2	40.9	43.1	45.9
DINOv2 [26]	ViT-B14	LVD	13.0	8.4	7.1	10.3	35.9	20.9	16.3	28.0	55.8	37.0	34.4	46.7	62.4	51.9	53.3	56.8
DINOv2 [26]	ViT-L14	LVD	8.5	6.2	5.3	7.5	25.0	14.0	10.8	19.3	53.9	34.6	31.6	44.5	62.8	53.3	54.2	57.2
DINOv2 + reg [7]	ViT-B14	LVD	12.0	8.3	7.3	10.1	33.2	19.6	15.4	26.1	57.4	39.4	38.1	48.8	58.3	51.4	53.4	53.7
Classification-Supervised Models																		
DeiT III [38]	ViT-B16	IN-22k	21.8	12.4	9.4	16.9	41.8	31.7	34.6	36.8	37.7	33.1	35.4	35.0	17.1	15.6	16.0	16.1
DeiT III [38]	ViT-L16	IN-22k	23.6	15.0	12.3	18.5	47.0	35.7	39.2	41.4	40.6	35.3	38.1	37.5	27.7	24.8	25.1	25.4
ConvNeXt [22]	CNXT-B	IN-22k	4.6	3.8	4.1	4.2	7.9	6.6	6.6	7.5	20.7	16.9	18.6	18.8	14.5	12.5	13.3	13.5
Vision Language Models																		
CLIP[28]	ViT-B16	WIT	6.1	5.4	5.7	5.7	5.8	4.3	3.7	4.9	4.9	3.6	3.5	4.3	6.0	3.5	2.8	4.7
CLIP[28]	ViT-L14	WIT	4.6	4.4	4.4	4.6	4.0	3.4	3.7	3.9	3.5	2.6	3.0	3.2	3.2	2.6	2.9	3.0
CLIP[14]	ViT-B16	LAION	5.6	5.0	4.2	5.2	5.5	4.1	4.1	4.8	6.4	3.1	2.8	5.0	7.2	3.2	2.6	5.3
CLIP[14]	CNXT-B	LAION	6.0	5.3	5.4	5.5	5.3	4.1	4.1	4.8	24.8	19.9	20.4	21.9	21.5	18.6	21.1	19.1
CLIP + AugReg[14]	CNXT-B	LAION	5.4	4.8	4.2	5.1	6.8	5.6	5.7	6.1	27.4	22.9	22.4	24.4	27.9	24.3	27.8	25.7
SigLIP [46]	ViT-B16	WebLI	8.4	7.4	8.1	8.1	8.7	7.2	7.6	8.2	7.7	6.7	6.4	7.1	5.9	5.2	4.9	5.6
SigLIP [46]	ViT-L16	WebLI	7.1	5.7	5.8	6.5	7.3	6.0	6.2	6.7	6.7	5.3	5.5	6.3	4.4	3.9	4.0	4.3
Text-Conditioned Image Generation Models																		
StableDiffusion [32]	UNet	LAION	14.2	5.7	3.8	10.1	58.0	34.6	28.5	46.4	21.9	15.5	12.2	17.8	4.6	4.1	4.2	4.2
Densely-Supervised Models																		
SAM [16]	ViT-B16	SA-1B	9.2	5.5	4.6	7.4	16.4	10.7	8.7	13.0	29.0	18.6	14.0	23.2	30.9	19.2	14.4	24.7
SAM [16]	ViT-L16	SA-1B	9.9	6.1	5.4	8.0	22.6	15.8	12.5	18.3	34.8	23.1	17.0	28.2	30.2	18.1	13.0	24.1
MiDaS [31]	ViT-L16	MIX 6	15.6	10.2	8.7	13.0	27.3	22.8	23.2	24.5	28.2	23.4	25.1	25.5	25.8	21.3	23.6	23.4

References

[1] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *arXiv preprint arXiv:2112.05814*, 2021. 4

[2] Gwangbin Bae, Ignas Budvytis, and Roberto Cipolla. Estimating and exploiting the aleatoric uncertainty in surface normal estimation. In *ICCV*, 2021. 4

[3] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, 2021. 3

[4] Shariq Farooq Bhat, Reiner Birkl, Diana Wofk, Peter Wonka, and Matthias Müller. Zoedepth: Zero-shot transfer by combining relative and metric depth. *arXiv preprint arXiv:2302.12288*, 2023. 2

[5] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2021. 1, 9, 10, 11, 12, 13

[6] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 3

[7] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023. 2, 9, 10, 11, 12, 13

[8] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeruIPS*, 2014. 3

[9] Mohamed El Banani, Karan Desai, and Justin Johnson. Learning Visual Representations via Language-Guided Sampling. In *CVPR*, 2023. 7

[10] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338, 2010. 3

[11] David F. Fouhey, Wajahat Hussain, Abhinav Gupta, and Martial Hebert. Single image 3D without a single 3D image. In *ICCV*, 2015. 4

[12] Micah Goldblum, Hossein Souri, Renkun Ni, Manli Shu, Viraj Prabhu, Gowthami Somepalli, Prithvijit Chattopadhyay, Mark Ibrahim, Adrien Bardes, Judy Hoffman, Rama Chellappa, Andrew Gordon Wilson, and Tom Goldstein. Battle of the backbones: A large-scale comparison of pretrained models across computer vision tasks. In *NeurIPS Datasets and Benchmarks Track*, 2023. 1

[13] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1, 9, 10, 11, 12, 13

[14] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. OpenCLIP, 2021. 2, 9, 10, 11, 12, 13

[15] Varun Jampani, Kevis-Kokitsi Maninis, Andreas Engelhardt, Arjun Karpur, Karen Truong, Kyle Sargent, Stefan Popov, André Araujo, Ricardo Martin-Brualla, Kaushal Patel, Daniel Vlasic, Vittorio Ferrari, Ameesh Makadia, Ce Liu, Yuanzhen Li, and Howard Zhou. Navi: Category-agnostic image collections with high-quality 3d shape and pose annotations. In *NeurIPS Datasets and Benchmarks Track*, 2023. 2

[16] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. *arXiv preprint arXiv:2304.02643*, 2023. 2, 9, 10, 11, 12, 13

[17] L'ubor Ladickỳ, Bernhard Zeisl, and Marc Pollefeys. Discriminatively trained dense surface normal estimation. In *ECCV*, 2014. 3

[18] Martha Lewis, Qinan Yu, Jack Merullo, and Ellie Pavlick. Does clip bind concepts? probing compositionality in large image models. *arXiv preprint arXiv:2212.10537*, 2022. 6

[19] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2041–2050, 2018. 3

[20] Zhuowan Li, Cihang Xie, Benjamin Van Durme, and Alan Yuille. Localization vs. semantics: Visual representations in unimodal and multimodal models. In *EACL*, 2024. 6

[21] Liyuan Liu, Haoming Jiang, Pengcheng He, Weizhu Chen, Xiaodong Liu, Jianfeng Gao, and Jiawei Han. On the variance of the adaptive learning rate and beyond. In *ICLR*, 2020. 4

[22] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, 2022. 2, 9, 10, 11, 12, 13

[23] David G Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 4

[24] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence. *arXiv prepreint arXiv:1908.10543*, 2019. 3, 5

[25] Norman Mu, Alexander Kirillov, David Wagner, and Saining Xie. Slip: Self-supervision meets language-image pre-training. In *Eur. Conf. Comput. Vis.*, 2022. 7

[26] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, Mahmoud Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Hervé Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning Robust Visual Features without Supervision, 2023. 1, 3, 6, 9, 10, 11, 12, 13

[27] Luigi Piccinelli, Christos Sakaridis, and Fisher Yu. idisc: Internal discretization for monocular depth estimation. In *CVPR*, 2023. 4

[28] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 9, 10, 11, 12, 13

[29] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *Int. Conf. Machine Learning*, 2021. 2

[30] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *TPAMI*, 2020. 2

[31] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 2, 4, 9, 10, 11, 12, 13

[32] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, pages 10684–10695, 2022. 2, 9, 10, 11, 12, 13

[33] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich. Superglue: Learning feature matching with graph neural networks. In *CVPR*, 2020. 3

[34] Christoph Schuhmann, Richard Vencu, Romain Beaumont, Robert Kaczmarczyk, Clayton Mullis, Aarush Katta, Theo Coombes, Jenia Jitsev, and Aran Komatsuzaki. Laion-400m: Open dataset of clip-filtered 400 million image-text pairs. *arXiv preprint arXiv:2111.02114*, 2021. 2

[35] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2, 3

[36] Andreas Steiner, Alexander Kolesnikov, Xiaohua Zhai, Ross Wightman, Jakob Uszkoreit, and Lucas Beyer. How to train your vit? data, augmentation, and regularization in vision transformers. *arXiv preprint arXiv:2106.10270*, 2021. 2, 6

[37] Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. Emergent correspondence from image diffusion. *arXiv preprint arXiv:2306.03881*, 2023. 2, 5

[38] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit III: Revenge of the ViT. In *ECCV*, 2022. 2, 9, 10, 11, 12, 13

[39] Matthew Walmer, Saksham Suri, Kamal Gupta, and Abhinav Shrivastava. Teaching matters: Investigating the role of supervision in vision transformers. In *IEEE Conf. Comput. Vis. Pattern Recog.*, 2023. 4

[40] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 1

[41] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, 2020. Association for Computational Linguistics. 1

[42] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. In *CVPR*, 2023. 1, 9, 10, 11, 12, 13

[43] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2014. 3

[44] Jiarui Xu, Sifei Liu, Arash Vahdat, Wonmin Byeon, Xiaolong Wang, and Shalini De Mello. ODISE: Open-Vocabulary Panoptic Segmentation with Text-to-Image Diffusion Models. In *CVPR*, 2023. 2

[45] Wei Yin, Chi Zhang, Hao Chen, Zhipeng Cai, Gang Yu, Kaixuan Wang, Xiaozhi Chen, and Chunhua Shen. Metric3d: Towards zero-shot metric 3d prediction from a single image. In *ICCV*, 2023. 3

[46] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. *ICCV*, 2023. 2, 9, 10, 11, 12, 13

[47] Guanqi Zhan, Chuanxia Zheng, Weidi Xie, and Andrew Zisserman. What does stable diffusion know about the 3d scene?, 2023. 2, 4

[48] Junyi Zhang, Charles Herrmann, Junhwa Hur, Luisa Polania Cabrera, Varun Jampani, Deqing Sun, and Ming-Hsuan Yang. A tale of two features: Stable diffusion complements dino for zero-shot semantic correspondence. In *NeurIPS*, 2023. 2, 3, 4, 5

[49] Wenliang Zhao, Yongming Rao, Zuyan Liu, Benlin Liu, Jie Zhou, and Jiwen Lu. Unleashing text-to-image diffusion models for visual perception. *arXiv preprint arXiv:2303.02153*, 2023. 2

[50] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *International Conference on Learning Representations (ICLR)*, 2022. 1, 9, 10, 11, 12, 13