DepthCues: Evaluating Monocular Depth Perception in Large Vision Models

Supplementary Material

A. Additional Results on *DepthCues*

A.1. Full Benchmark Results

In the main paper, we reported the performance of 20 pre-trained vision models on *DepthCues* and NYUv2 [28] using bar charts, and focused on salient observations. Here we provide detailed numerical results on these datasets and additionally the linear probing results on ImageNet-1k [6]. Moreover, we also provide an analysis of the impact of the pre-training dataset, objective, and architectural choices of the vision models on their performance. Our analysis is based solely on publicly available models; therefore, as outlined in the limitations section of the main paper, we are unable to control all variables in these analyses.

The probing results of all models are summarized in Tab. A1, and the average performance of models on *DepthCues* is plotted against their NYUv2 depth estimation (Fig. A1) as well as DIW depth ordering (Fig. A2) results. It is noted that there are slight differences between the linear probing results on NYUv2 and ImageNet-1k compared to previous works [8, 22]. These may stem from differences in the implementation details including the features probed, learning rates, and training iterations, *etc*. For example, the probing results on NYUv2 depth estimation in [8] are generally higher than our results. Possible reasons for this include the following factors. Firstly, while we train a simple linear layer to probe the patch tokens from the last layer of the vision models, a more complex non-linear probe similar to the DPT decoder [25] is used to probe features from multiple layers of the models in [8]. Secondly, the class tokens are utilized for several models (*e.g.,* DINO, DINOv2) in [8], while we only use patch tokens for consistent evaluation across all models as some of the models do not include a class token. Thirdly, because of the difference in the probes, their optimization hyper-parameters can differ. Similar reasons can be applied to the differences in our ImageNet-1k probing results from previous works. In particular, for CLIP, the class token seems to be crucial for its ImageNet classification ability. We made these design choices because our primary focus is to evaluate the awareness of human-like monocular depth cues in vision models in a fair and comparable manner, rather than optimizing their probing performance on external benchmarks.

Impact of Pre-Training Dataset Size. Fig. A3 shows the average performance of models on *DepthCues* against the size of their pre-training datasets. Excluding two vision-language models (CLIP and SigLIP) and the generative model (StableDiffusion, which also trained for text-to-image generation using language supervision), there is a

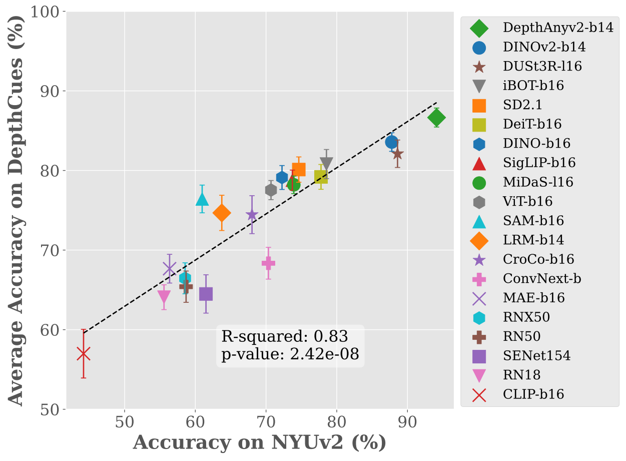

Figure A1. **Performance of vision models on *DepthCues* vs. NYUv2 depth estimation.** A strong correlation is observed between depth cue understanding and depth estimation.

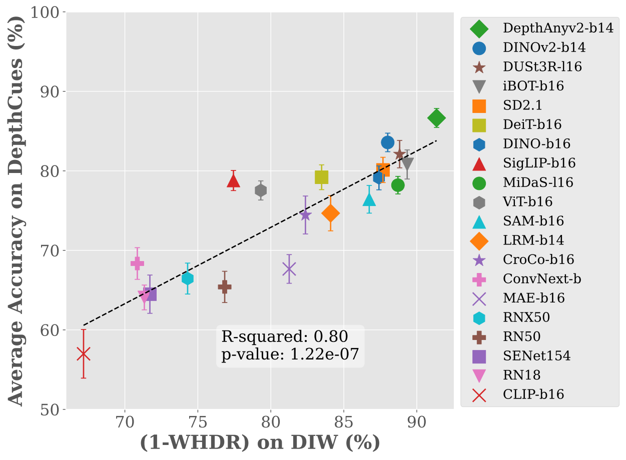

Figure A2. **Performance of vision models on *DepthCues* vs. DIW depth ordering.** A strong correlation is observed between depth cue understanding and depth estimation.

moderate positive correlation between these two variables (Pearson's $r = 0.69, p = 0.002$). Apart from the overall trend, we observe that certain models demonstrate greater data efficiency in developing depth cue awareness. For instance, among models pre-trained on approximately 10^6 data samples, DINO achieves the best performance. For those pre-trained on around 10^7 samples, DUSt3R emerges as the best. Finally, when comparing models pre-trained on roughly one order of magnitude more data, DepthAnythingv2 outperforms DINOv2. However, there are other

	DepthCues						NYUv2	ImageNet-1k
	elevation (%)	light-shadow (%)	occlusion (%)	perspective (%)	size (%)	texture-grad (%)	depth (%)	classification (%)
CLIP-b16	27.56 (2.76)	62.49 (0.19)	58.41 (0.20)	58.40 (3.06)	68.37 (0.21)	66.68 (0.77)	44.20	0.59
RN18	57.17 (0.83)	61.86 (0.84)	72.79 (0.20)	53.87 (3.97)	70.57 (0.11)	68.22 (0.35)	55.57	68.67
SENet154	55.19 (2.97)	60.95 (1.04)	74.07 (0.09)	47.47 (2.71)	76.94 (0.63)	72.30 (0.32)	61.51	81.58
RN50	58.38 (1.61)	61.26 (0.48)	75.96 (0.18)	52.53 (3.05)	76.01 (1.35)	68.26 (1.00)	58.70	73.62
RNX50	59.72 (1.73)	62.49 (0.82)	76.36 (0.20)	52.93 (5.07)	75.93 (0.77)	71.28 (0.07)	58.56	76.40
MAE-b16	51.03 (3.33)	75.72 (0.99)	68.05 (0.22)	66.93 (3.23)	74.27 (1.40)	69.96 (0.66)	56.36	25.99
ConvNext-b	62.52 (1.21)	61.86 (0.64)	75.60 (0.22)	55.07 (6.52)	79.04 (0.47)	75.98 (0.50)	70.34	80.37
CroCo-b16	56.64 (1.57)	78.28 (0.66)	69.21 (0.08)	74.53 (4.57)	75.64 (0.72)	92.40 (0.14)	68.02	14.52
LRM-b14	56.58 (1.75)	76.56 (0.68)	69.14 (0.02)	83.07 (5.79)	75.51 (0.97)	87.18 (0.51)	63.75	17.68
SAM-b16	62.33 (1.56)	74.93 (0.70)	78.29 (0.10)	89.07 (1.16)	76.74 (0.49)	77.18 (0.43)	60.97	19.58
ViT-b16	71.68 (1.57)	74.42 (0.50)	73.94 (0.06)	87.60 (2.09)	76.29 (1.42)	81.28 (0.23)	70.70	68.67
MiDaS-l16	71.04 (1.38)	77.70 (0.66)	74.02 (0.16)	84.93 (6.29)	77.70 (2.27)	83.80 (0.50)	73.93	42.79
SigLIP-b16	71.44 (4.12)	76.98 (0.53)	77.76 (0.08)	89.60 (3.64)	81.40 (1.10)	75.52 (0.13)	73.73	37.96
DINO-b16	73.80 (3.15)	75.19 (0.29)	72.63 (0.18)	91.20 (0.78)	76.88 (0.40)	85.00 (0.54)	72.26	41.29
DeiT-b16	67.44 (1.26)	79.81 (0.41)	78.00 (0.12)	91.47 (1.66)	78.17 (0.92)	80.28 (0.37)	77.77	84.53
SD2.1	69.45 (3.56)	78.53 (0.26)	78.20 (0.14)	93.07 (1.00)	77.92 (1.85)	83.50 (0.74)	74.63	32.67
iBOT-b16	72.08 (1.32)	76.05 (0.50)	75.19 (0.09)	96.67 (1.58)	79.61 (0.89)	85.24 (0.34)	78.54	38.92
DUSt3R-l16	74.65 (2.44)	76.75 (0.45)	76.02 (0.30)	95.47 (0.98)	80.31 (0.92)	89.42 (0.35)	88.59	25.94
DINOv2-b14	79.13 (1.11)	83.95 (0.25)	79.93 (0.17)	94.53 (1.86)	83.57 (0.93)	80.32 (0.49)	87.78	77.95
DepthAnyv2-b14	83.74 (0.57)	84.74 (0.68)	81.01 (0.08)	96.93 (0.90)	83.51 (1.59)	89.98 (0.34)	94.12	68.67

Table A1. **Detailed evaluation results on *DepthCues*, depth estimation (NYUv2), and image classification (ImageNet-1k).** The mean and standard deviation of the accuracy of each vision model on the six tasks in our benchmark are summarized. The last two columns show the accuracy of these models on NYUv2 depth estimation and ImageNet-1k classification.

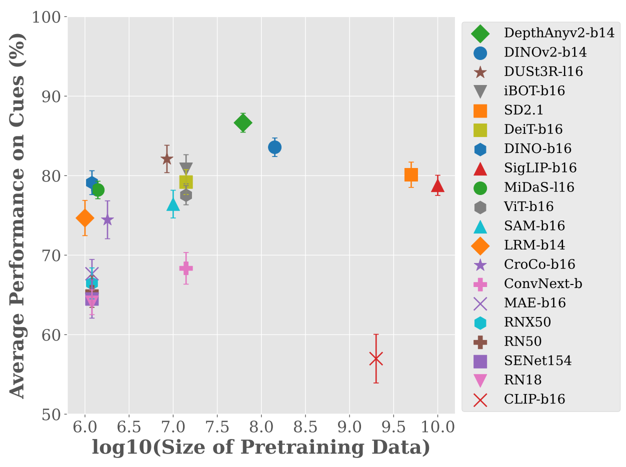

Figure A3. **Impact of pre-training dataset size on the performance of vision models on *DepthCues*.** Omitting the three models that involve language supervision (CLIP, SigLIP, and SD), we observe a moderate positive correlation between depth cue understanding and pre-training data size.

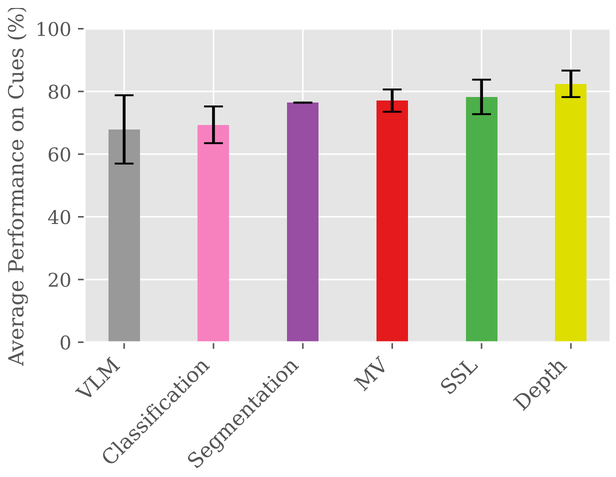

Figure A4. **Average performance of vision models on *DepthCues* by pre-training objective.** The error bar denotes the standard deviation over models.

factors such as the pre-training objective, model architecture, and the dataset distribution which can confound these analyses. It is desirable to perform future studies on the pre-training dataset size where the aforementioned factors are better controlled.

Impact of Pre-Training Objective. We report the average performance achieved on *DepthCues* by models pre-trained with different types of supervision in Fig. A4. On average, models pre-trained for depth estimation obtain the highest performance, followed by (single-view) self-supervised (SSL) and multi-view models. Considering the differences in the pre-training data of these groups, we also collate results for different groups when the pre-training dataset is the same. The only models that support such analysis are either classification or SSL-based (more details in Appendix D), and their results are shown in Fig. A5. We can see from Fig. A5 (a) and (b) that SSL methods have developed a better understanding of studied depth cues on average. However, in these two plots, the architecture of classification models is mixed (convolutional and transformer-

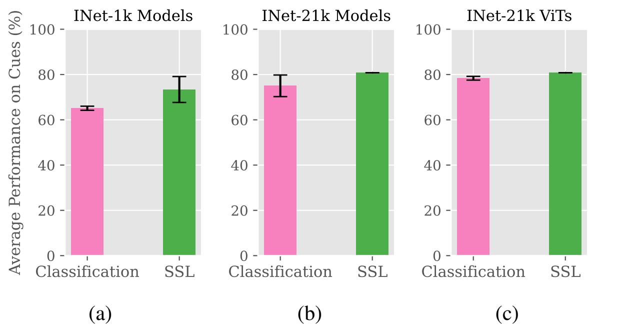

Figure A5. **Average performance of vision models on *DepthCues* by pre-training objective, with fixed pre-training data.** We show results for ImageNet-1k (a) and ImageNet-21k models (b), and ImageNet-21k ViT-based models (c). The error bar denotes the standard deviation over models.

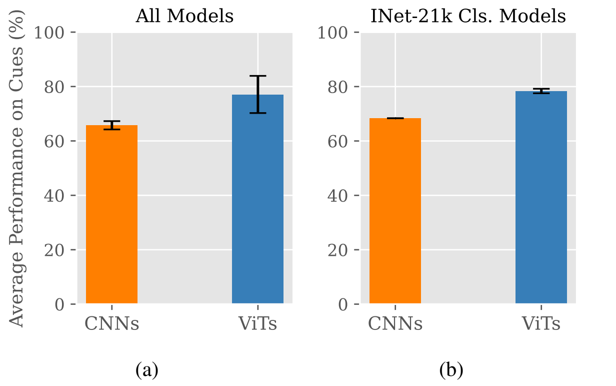

Figure A6. **Average performance of vision models on *DepthCues* by model architecture.** We show results for all models (a) and ImageNet-21k classification models (b). The error bar denotes the standard deviation over models.

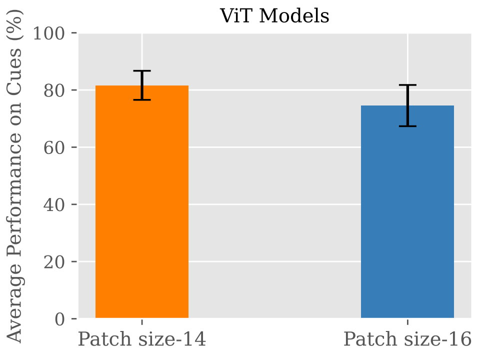

Figure A7. **Average performance of ViT-based models on *DepthCues* by patch size.** The error bar denotes the standard deviation over models.

based models), whereas SSL models only contain ViT backbones. Therefore, we further reduce the effect of architecture and show a comparison for ViT-only models ('Base' models; patch size 16) in Fig. A5 (c), where we still see an

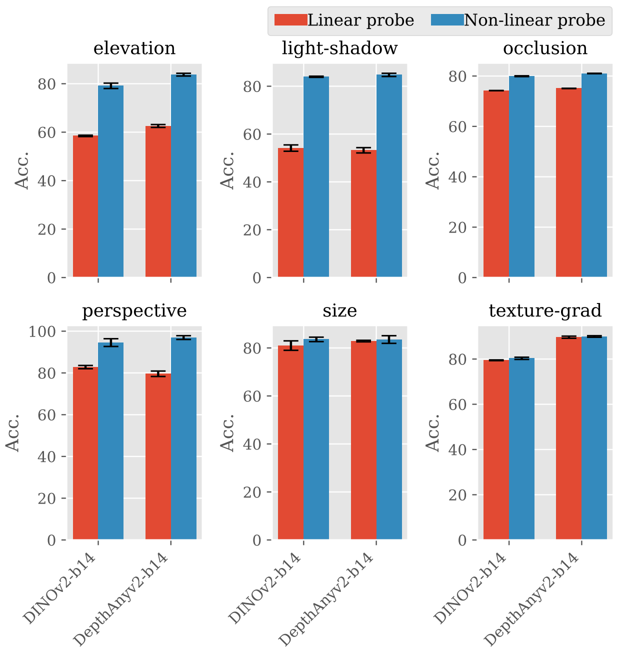

Figure A8. **Linear vs. non-linear probes.** We see consistently better performance from non-linear probes across the tasks in *DepthCues*.

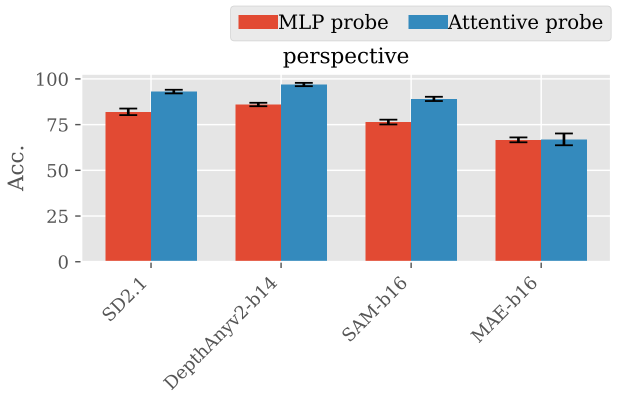

Figure A9. **MLP vs. Attentive probe on the perspective task.** We observe that for a task that requires global information, the attentive probe yields better results than an MLP.

advantage from the SSL model (iBOT) over the classification ones (ViT and DeiT).

Impact of Model Architecture. Here we compare the average depth cue awareness between vision transformer (ViT)-based models and models with convolutional architectures (CNNs). Fig. A6 (a) shows the results for all evaluated models, regardless of pre-training objective and dataset, where we see a clear advantage of the ViT models. To reduce confounding factors, in Fig. A6 (b) we also compare these architectures when the pre-training setting is fixed to be classification on ImageNet-21k. It is observed that under such a setting, the two ViT models (ViT and DeiT) demonstrate enhanced knowledge of depth cues compared to ConvNext.

Impact of Patch Size on ViT Models. We also perform a comparison between ViT models regarding their patch size. As shown in Fig. A7, models with a smaller patch size (14 *vs*. 16) achieved better average performance on *DepthCues*. However, it is noted that the "patch size-14" models only include DINOv2, DepthAnythingv2, and LRM, and there may be other factors that led to their superior performance.

A.2. Comparison Between Probes

As discussed in Sec. 4.2 in the main paper, we adopt non-linear probes to evaluate models on *DepthCues*. Specifically, an MLP probe is used for light-shadow, occlusion, size, and texture-grad, and an attentive probe [1, 9] is used for elevation and perspective. Our motivation for using these instead of linear probes is that it is not clear that the solutions to our tasks should be a linear function of the model features. In addition, non-linear probes have been adopted in previous work on probing vision models for depth estimation [8, 10]. To justify our choice empirically, we compare our non-linear probes and the linear probe. Following the same protocol described in the main paper, we obtained additional linear probing results of two models (DINOv2 and DepthAnythingv2) on all six depth cue tasks. The results are summarized in Fig. A8, showing that our non-linear probes consistently outperform linear probes, based on the average of five runs, although the gap is small for size and texture-grad. Moreover, for the light-shadow task, the performance of the linear probe is close to random, indicating that a linear classifier is not sufficient to solve the problem using these models' features.

For elevation and perspective, our choice of the attentive probe, instead of the MLP probe used for the other four tasks, is motivated by the extra step in the former for aggregating global information, which we consider important for these two tasks. This is also supported by our results in Fig. A9, where it can be seen that the attentive probe results in significant performance gains for most models on the perspective task.

A.3. Hyper-Parameter Search Results

It has been shown in previous works [21, 42] that different layers of pre-trained vision models have varied performance when probed for different tasks. Therefore, for all the 20 vision models, we perform a hyper-parameter search on their layers. We restrict the search to four layers for each model, which are selected by equally dividing the networks into four blocks (similar to [8]), where applicable. We train the probes on features from each block, and evaluate their validation performance. The layer with the best validation result is then selected for subsequent analysis. The layer search results for all models are summarized in Tab. A2. Consistent with previous findings, we observe that different layers of a model exhibit varying strengths. For instance, in DINOv2, the 9th layer achieved the best performance on light-shadow and occlusion, the 12th layer provides superior features for elevation, perspective, and size, while texture-grad is handled best by the 6th layer.

A.4. Example Model Failure Cases

We examine where vision models consistently fail. Focusing on the top-five models, namely DepthAnythingv2, DINOv2, DUSt3R, iBOT, and StableDiffusion, we identify test instances where they all fail to predict the correct label over five runs, and visualize these cases in Fig. A10.

Elevation. In the first (top row) instance of Fig. A10 (a), the horizon line is largely occluded by the architectures, making it a challenging case. In the second case (bottom row), although the horizon line is less occluded by objects, its visibility is still low due to the fog, which can be a possible explanation for the failure of the models.

Light-shadow. In both of the examples in Fig. A10 (b), the target objects (overlaid with red masks) and query shadows (overlaid with green masks) overlap with each other, possibly affecting the models' predictions. This is especially the case for the second image, where the true shadow of the target object also overlaps with the false query shadow.

Occlusion. The first example (top row) of Fig. A10 (c) is challenging because the person highlighted with a red mask undergoes only very minor occlusion, limited to his feet. In the second example, the stones are treated as one object and labeled occluded in the source data [44], which can be confused with their individual boundaries or occlusions.

Perspective. Both of the examples in Fig. A10 (d) contain distractors that weaken the notion of the dominant vanishing point. In the first example (top), while the ground-truth label indicates the converging point of the two sides of the staircase, some models are influenced by the parallel lines formed by the bricks on the walls. In the second example (bottom), some models might be affected by the intersection line of the mountain and sky.

Size. In the first example of Fig. A10 (e), all models incorrectly predicted that the car (highlighted in red) has a larger 3D size than the van (highlighted in green). In the second example, while the trailer (green) has a smaller size than the car, all models predicted the opposite case. These indicate limitations in the models' understanding of the size cue.

Texture-grad. All models failed to predict the correct depth order in both cases in Fig. A10 (f). In these challenging images, the texture gradient cue is relatively weak, and the locations of the two regions are not drastically different, which can make these cases challenging.

Model	Layers Probed	Elevation (%)				Light-shadow (%)				Occlusion (%)			
		Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4
CLIP-b16	{3, 6, 9, 12}	**0.189**	0.243	0.372	0.355	**66.357**	64.965	65.893	65.429	**58.283**	57.531	55.912	56.206
RN18	{1, 2, 3, 4}	0.145	0.127	0.122	**0.105**	64.965	63.805	**65.197**	65.029	58.360	67.302	**73.175**	66.744
SENet154	{1, 2, 3, 4}	0.120	0.108	**0.105**	0.112	65.893	66.357	66.473	**67.749**	60.987	**74.066**	73.532	68.681
RN50	{1, 2, 3, 4}	0.138	0.118	**0.096**	0.109	65.197	65.429	65.313	**69.026**	59.391	65.737	**75.701**	70.735
RNX50	{1, 2, 3, 4}	0.136	0.113	**0.097**	0.113	65.777	65.545	66.357	**68.329**	59.329	69.913	**76.554**	67.480
MAE-b16	{3, 6, 9, 12}	0.151	0.130	**0.116**	0.122	81.323	81.787	**82.251**	80.510	61.623	63.381	**67.550**	66.744
ConvNext-b	{1, 2, 3, 4}	0.129	0.121	0.108	**0.096**	65.197	66.125	**69.722**	67.169	59.623	65.838	**75.136**	71.347
CroCo-b16	{3, 6, 9, 12}	0.125	0.111	**0.107**	0.108	83.411	**83.991**	83.759	83.411	62.870	65.946	**68.712**	68.271
LRM-b14	{3, 6, 9, 12}	0.135	0.108	**0.108**	0.114	68.677	81.903	**84.919**	83.411	62.142	67.651	**69.650**	67.256
SAM-b16	{3, 6, 9, 12}	0.133	0.111	0.085	**0.084**	80.046	**80.858**	80.046	80.278	63.513	72.571	78.080	**78.700**
ViT-b16	{3, 6, 9, 12}	0.101	**0.081**	0.087	0.098	79.466	**81.671**	76.798	70.186	64.637	72.393	**73.291**	67.883
MiDaS-l16	{6, 12, 18, 24}	0.108	**0.078**	0.091	0.083	**83.759**	80.974	79.814	78.886	69.882	**73.524**	73.408	73.222
SigLIP-b16	{3, 6, 9, 12}	0.090	0.072	**0.072**	0.083	80.858	**86.543**	81.903	71.694	68.852	**76.809**	74.508	70.277
DINO-b16	{3, 6, 9, 12}	0.099	0.073	**0.073**	0.074	81.206	82.483	**83.179**	81.903	65.791	72.641	73.470	**73.617**
DeiT-b16	{3, 6, 9, 12}	0.091	**0.084**	0.086	0.118	83.527	**87.007**	73.434	66.357	68.681	**77.158**	73.679	67.488
SD2.1	{1, 2, 3, 4}	0.100	**0.094**	0.096	0.139	81.090	**88.399**	79.698	66.125	69.371	**77.096**	72.129	59.468
iBOT-b16	{3, 6, 9, 12}	0.114	0.093	0.081	**0.078**	78.190	82.947	83.527	**83.527**	62.994	72.772	**75.027**	73.570
DUSt3R-l16	{18, 24, 33, 36}	**0.066**	0.076	0.083	0.093	83.759	**86.427**	85.383	82.251	73.764	69.975	74.330	**75.833**
DINOv2-b14	{3, 6, 9, 12}	0.117	0.073	0.068	**0.065**	67.053	79.234	**89.211**	87.935	63.412	76.399	**79.544**	75.205
DepthAnyv2-b14	{3, 6, 9, 12}	0.114	0.070	0.064	**0.058**	66.473	79.350	**90.603**	90.023	63.436	77.111	**80.962**	77.057

Model	Layers Probed	Perspective (%)				Size (%)				Texture-grad (%)			
		Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4	Block 1	Block 2	Block 3	Block 4
CLIP-b16	{3, 6, 9, 12}	0.259	**0.230**	0.253	0.264	**73.636**	73.635	72.727	70.909	**67.000**	64.900	63.600	61.200
RN18	{1, 2, 3, 4}	0.290	0.265	**0.235**	0.259	68.485	**78.788**	77.273	77.273	**67.400**	65.300	64.500	63.700
SENet154	{1, 2, 3, 4}	0.270	**0.250**	0.262	0.294	74.242	77.879	**83.333**	81.515	69.200	**74.800**	66.300	64.400
RN50	{1, 2, 3, 4}	0.296	0.290	**0.230**	0.264	73.030	75.455	**78.788**	78.485	68.500	69.000	**70.200**	67.300
RNX50	{1, 2, 3, 4}	0.294	0.270	**0.240**	0.277	70.606	78.182	**80.606**	76.061	68.900	**72.600**	69.000	65.200
MAE-b16	{3, 6, 9, 12}	0.264	0.240	0.218	**0.217**	76.364	**79.697**	79.696	78.485	66.300	69.500	68.900	**69.800**
ConvNext-b	{1, 2, 3, 4}	0.291	0.296	**0.244**	0.248	71.818	74.242	80.303	**81.818**	70.800	**73.600**	61.600	63.600
CroCo-b16	{3, 6, 9, 12}	0.234	0.177	**0.148**	0.153	76.364	77.273	**79.091**	78.788	74.900	89.800	91.600	**92.700**
LRM-b14	{3, 6, 9, 12}	0.253	0.200	0.146	**0.137**	75.152	78.485	**80.000**	79.697	66.900	72.900	81.000	**87.800**
SAM-b16	{3, 6, 9, 12}	0.224	0.178	**0.116**	0.145	76.667	80.909	80.606	**80.909**	76.800	**78.200**	74.600	71.800
ViT-b16	{3, 6, 9, 12}	0.149	0.109	**0.108**	0.213	77.879	79.394	**80.000**	78.788	75.100	**82.600**	76.000	70.100
MiDaS-l16	{6, 12, 18, 24}	0.227	**0.135**	0.137	0.158	76.970	**78.788**	77.879	78.485	84.100	**85.200**	84.000	83.300
SigLIP-b16	{3, 6, 9, 12}	0.181	**0.111**	0.124	0.161	79.091	81.515	79.697	**82.121**	76.700	**78.300**	69.700	61.700
DINO-b16	{3, 6, 9, 12}	0.152	**0.078**	0.092	0.115	77.576	**80.303**	77.576	79.091	80.000	84.700	**84.900**	84.000
DeiT-b16	{3, 6, 9, 12}	0.179	**0.107**	0.194	0.251	77.576	**80.909**	80.303	80.606	79.900	**81.100**	72.100	62.500
SD2.1	{1, 2, 3, 4}	**0.102**	0.110	0.105	0.297	79.091	**81.212**	79.091	73.636	81.800	**83.200**	79.600	70.300
iBOT-b16	{3, 6, 9, 12}	0.182	0.086	**0.069**	0.078	79.091	80.909	79.697	**81.818**	80.400	83.600	**85.100**	84.200
DUSt3R-l16	{18, 24, 33, 36}	**0.076**	0.097	0.098	0.106	83.636	**85.455**	82.121	77.576	87.500	87.600	91.700	**91.900**
DINOv2-b14	{3, 6, 9, 12}	0.218	0.142	0.089	**0.088**	79.091	79.697	82.727	**85.455**	77.200	**82.500**	80.800	80.600
DepthAnyv2-b14	{3, 6, 9, 12}	0.189	0.091	**0.085**	0.092	77.576	83.030	82.424	**86.667**	76.700	83.000	86.100	**90.000**

Table A2. **Layer search for all models on all tasks in *DepthCues*.** We report the validation performance of different layers, and **bold** the best score for each model. Note that "Block i" corresponds to the ith layer indicated in the second column. Horizon detection error and Euclidean distance are used to assess validation performance for elevation and perspective respectively, while accuracy is used for the other four tasks.

	DepthCues (%)					
	elevation	light-shadow	occlusion	perspective	size	texture-grad
DINOv2	77.46	83.26	75.35	96.00	83.57	78.90
DINOv2+*DC*	79.84 (+2.38)	82.67 (-0.59)	76.11(+0.76)	94.00 (-2.00)	86.10 (+2.53)	88.30 (+9.40)

Table A3. **Probing results of the original and fine-tuned DINOv2 on *DepthCues*.** Here '+*DC*' indicates the fine-tuned model. We observe an overall increase in performance, especially on texture-grad.

Model	NYUv2 Acc. (%) ↑	DIW WHDR (%) ↓
DINOv2	87.81 (0.09)	11.95 (0.05)
`concat`(DINOv2, DINOv2+*DC*)	88.43 (0.09)	11.66 (0.11)
CLIP	43.82 (0.04)	35.33 (0.09)
`concat`(CLIP, CLIP+*DC*)	44.40 (0.18)	32.69 (0.98)

Table A4. **Linear probing on downstream depth estimation with different random seeds.** Fine-tuning on *DepthCues* benchmark is also repeated.

A.5. Learning Depth Cues

In Sec. 5.2 of the main paper, we presented linear probing results on NYUv2 and DIW, showing that fine-tuning on *DepthCues* improves models' performance on depth estimation. Here we further investigate whether the fine-tuned model has improved understanding of the depth cues by comparing the probing results of the fine-tuned and pre-trained DINOv2 on *DepthCues*. Note we focus on probing

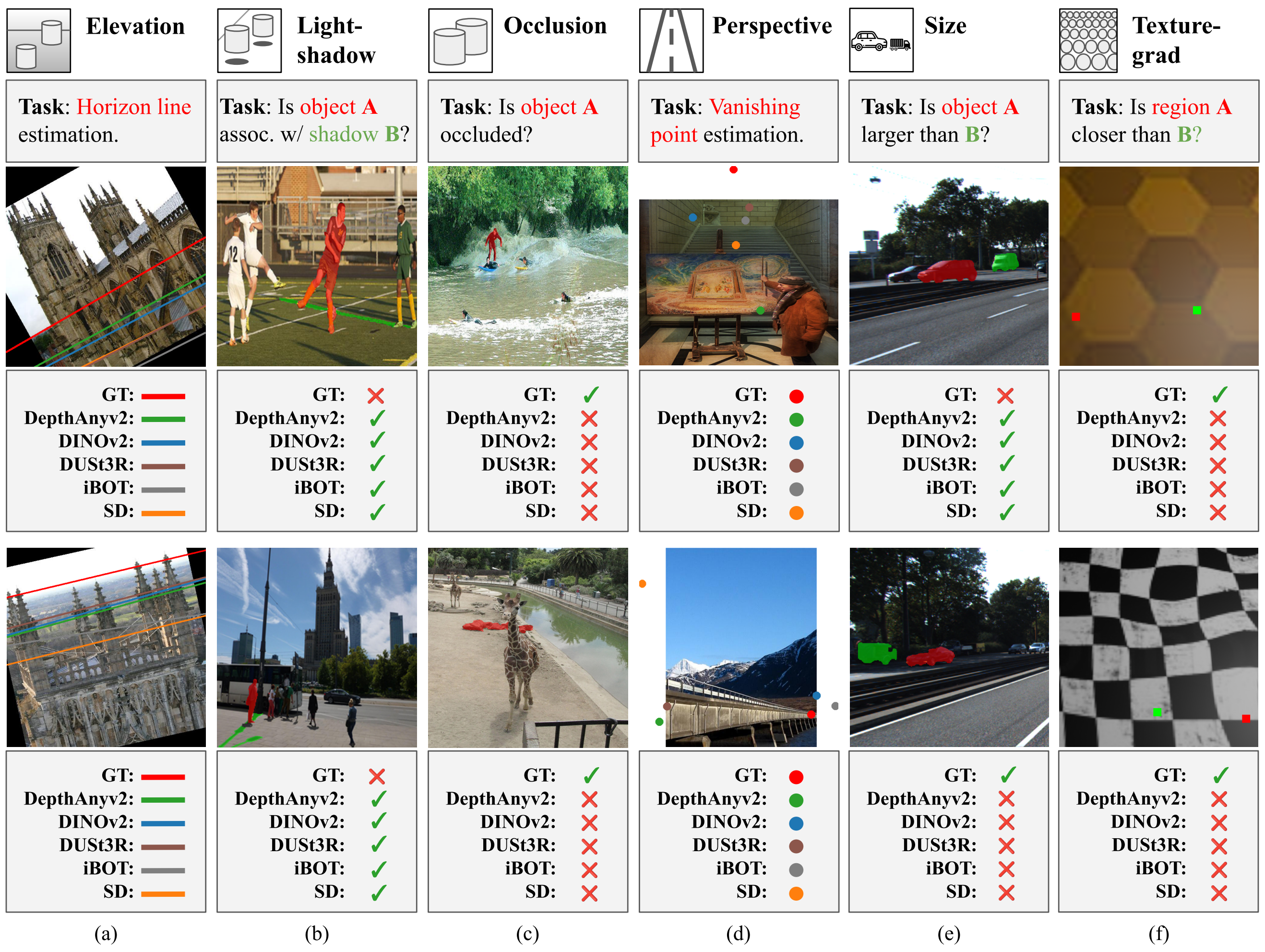
Elevation
Light-shadow
Occlusion
Perspective
Size
Texture-grad
Task: Horizon line estimation.
Task: Is object A assoc. w/ shadow B?
Task: Is object A occluded?
Task: Vanishing point estimation.
Task: Is object A larger than B?
Task: Is region A closer than B?
GT:
DepthAnyv2:
DINOv2:
DUSt3R:
iBOT:
SD:
(a)
(b)
(c)
(d)
(e)
(f)

Figure A10. **Failure cases of the top-five vision models on *DepthCues*.** Each column shows two examples for a depth cue.

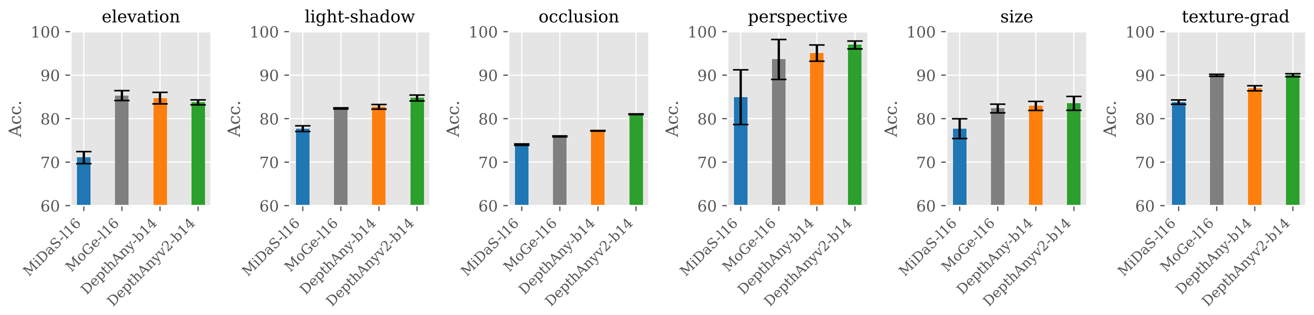

Figure A11. **Performance of monocular depth estimation models on *DepthCues*.** Here we evaluate recent depth estimation models, and observe that these models show competitive performance, indicating their strong understanding of the monocular depth cues.

the last layer since LoRA was only applied on that layer during fine-tuning. The results from Tab. A3 shows that the fine-tuned version outperforms the original DINOv2 on four cues, with a slight drop in performance on light-shadow and perspective. Notably, we see a 9.4% increase in accuracy on texture-grad, on which the original DINOv2 shows relatively weaker understanding (see Fig. 3 in the main paper). In addition, we repeated the fine-tuning (on *DepthCues*) and evaluation of DINOv2 and CLIP on NYUv2 and DIW with three different random seeds, and show the results in Tab. A4, where significant improvements are observed.

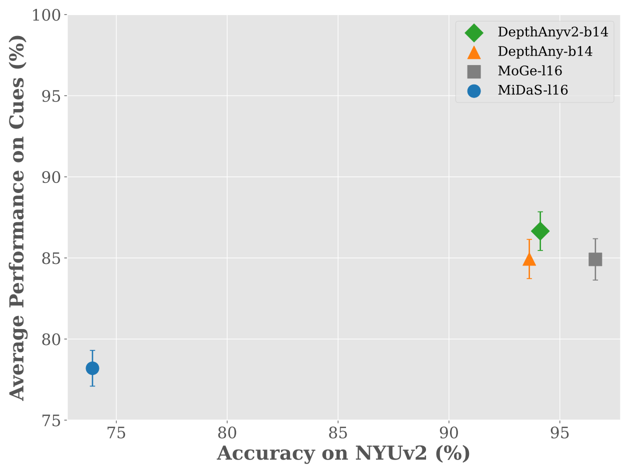

Figure A12. **Performance on monocular depth estimation models *DepthCues* vs. NYUv2 depth estimation**.

DAv2 ver.	Ele.	Lit-shd.	Occ.	Prsp.	Size	Txt.-grd.
relative	83.73	84.65	81.16	95.33	82.16	89.90
abs. (outdoor)	74.96	82.91	78.60	92.00	83.85	88.70
abs. (indoor)	79.96	82.44	79.67	94.67	83.43	89.90

Table A5. **Probing results of relative *vs*. metric versions of DepthAnyv2 on *DepthCues***.

A.6. Monocular Depth Estimation Models

In addition to the 20 vision models evaluated in the main paper, here we present additional results for some recent monocular depth estimation models: DepthAnything [39] and MoGe [31]. The average performance of these models on *DepthCues* and depth estimation is shown in Fig. A11 and Fig. A12, along with the two depth models already included in our study. In general, we observe a similar trend to our previous findings, that the models' depth estimation performance highly correlates with their performance on *DepthCues*.

Relative vs. Metric Depth Models. The DepthAnythingv2 model checkpoint we evaluated so far was pre-trained for relative depth estimation. Here we additionally evaluate the two publicly available metric versions of DepthAnythingv2 on *DepthCues*, which are fine-tuned from the relative version on indoor and outdoor datasets respectively. It is observed from Tab. A5 that the metric versions achieved lower performance than the relative one. As metric models are fine-tuned from the relative one on smaller datasets, the difference in performance could also be impacted by the training data. We leave further investigation to future work.

B. Dataset Construction Details

Here we provide additional details on the construction of the *DepthCues* benchmark and show other examples from the datasets.

Instance Selection for Size. To create an instance for the size dataset, we sample two objects from an image, and obtain the label by comparing the volumes of their 3D bounding boxes (provided by the source datasets, KITTI [11] and SUN-RGBD [29]). It is mentioned in the main paper that a threshold is applied to filter out cases where the difference in the sizes of the two objects is very small. This is motivated by our observation that the 3D bounding boxes from the source datasets can contain minor errors. Therefore, to reduce mislabeling in the size dataset, we apply a minimum threshold of 2.5 m^3 for the size difference between two objects in an image from KITTI, and a threshold of 0.4 m^3 for SUN-RGBD. We found empirically that these thresholds provided a good balance between label accuracy and dataset size.

Generating Masks with SAM. Four of the tasks in *DepthCues*, namely light-shadow, occlusion, size, and texture-grad, require object masks for the probing evaluation. The object masks for light-shadow and occlusion are directly obtained from their source datasets [33, 44], and the masks for texture-grad are manually defined during dataset synthesis (see Sec. 3.6 in the main paper). However, the object masks for the size task are not available in the source datasets which are originally designed for 3D object detection. Therefore, we made use of an off-the-shelf segmentation model, SAM [19], to create these masks. Specifically, for each object, we obtain its 2D bounding box from the source dataset, and discard the object if its 2D bounding box has a height/width less than eight pixels, to filter out potentially incorrect annotations. Next, we predict the mask for an object by feeding its 2D bounding box and the image to SAM, and only keep the part of the mask that falls into the 2D bounding box. Finally, to make sure the desired object is segmented, we check whether the mask takes up a too small portion of the bounding box ($< 20\%$), and discard the object if that is the case. These filtering steps are applied to both objects in a candidate image, and the candidate is included in the dataset only if both object masks pass these checks (and satisfy the size difference threshold specified above).

More Examples from the Datasets. Additional examples from the *DepthCues* datasets are shown in Fig. A13.

C. Additional Implementation Details

C.1. Probing Experiments

Probing Method. To evaluate vision models on *DepthCues*, we adopt a probing approach. While the task-specific feature extraction procedure and the probe models have been discussed in the paper, we illustrate the processes in Fig. A14.

Training Settings. The MLP probes for light-shadow, occlusion, size, and texture-grad are trained with a binary

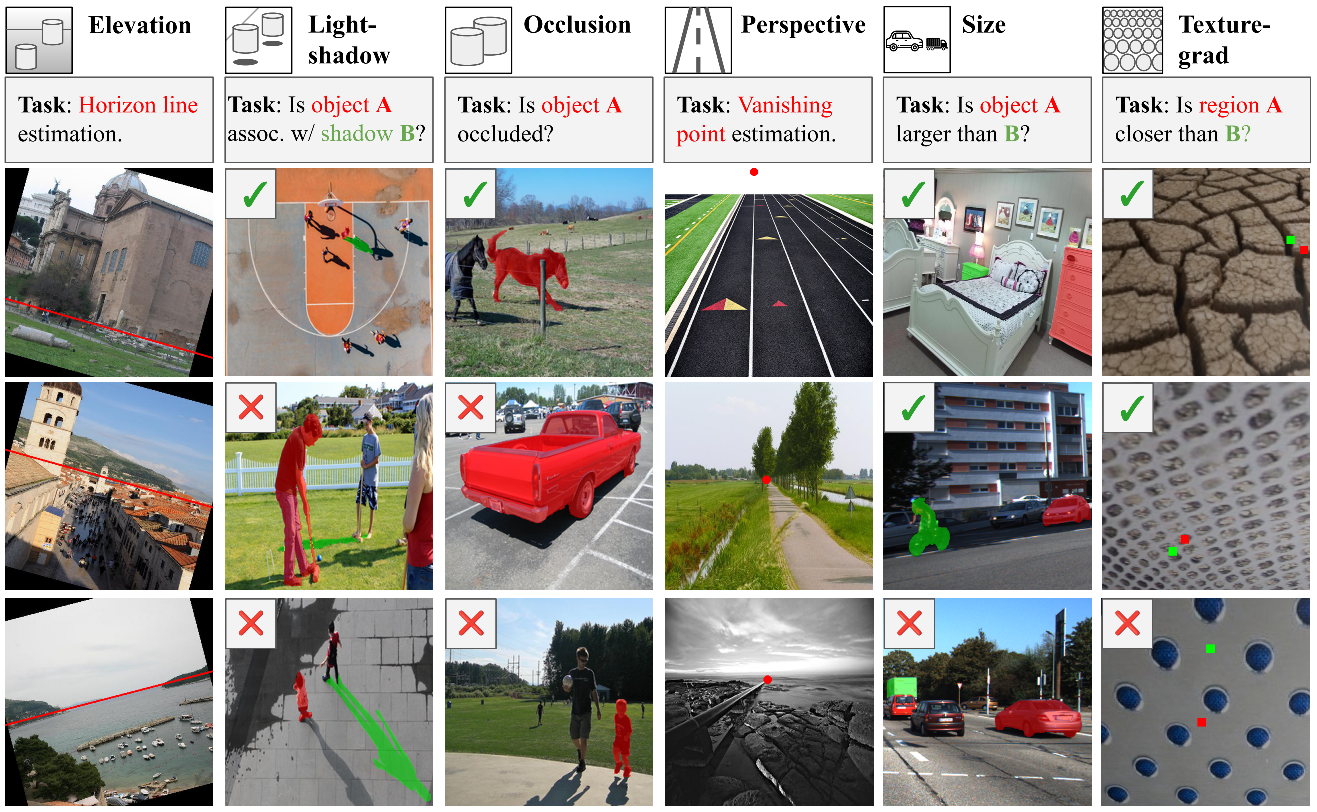
Elevation
Light-shadow
Occlusion
Perspective
Size
Texture-grad
Task: Horizon line estimation.
Task: Is object A assoc. w/ shadow B?
Task: Is object A occluded?
Task: Vanishing point estimation.
Task: Is object A larger than B?
Task: Is region A closer than B?

Figure A13. **Examples data instances from *DepthCues*.** The horizon lines for elevation and the vanishing points for perspective are indicated by red lines and dots respectively. For the other four cues where we define binary classification tasks, the labels are indicated by the ticks and crosses.

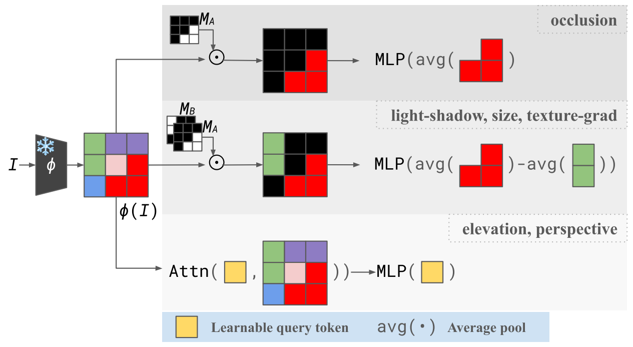

Figure A14. **Method for probing vision models on *DepthCues*.** We extract task-specific features from the image I using the vision model $\phi(\odot)$ and object masks where applicable, then train the MLP or attentive probe to solve the tasks in *DepthCues*. For illustrative purposes, here we only show 3×3 image features, but in practice, the spatial resolution is much higher.

cross-entropy loss since the associated tasks are binary classification ones. The attentive probes for elevation and perspective are trained with mean squared error loss due to their regression nature. All the MLP probes (for all tasks and all models) are trained for approximately 30k iterations with a batch size of eight, and the attentive probes are trained for 3,750 iterations with a batch size of 64. We used the AdamW [18] optimizer with cosine learning rate decay. For the hyperparameter search over model layers, we follow [8] and partition the networks into four equal-sized chunks, and evaluate the features from each chunk. For example, for DINOv2-b14, we search over layers 3, 6, 9, and 12.

Computational Cost. We provide an estimation of the computational cost for evaluating a model based on the ViT-Base architecture on *DepthCues*, using the standard implementation[1] of DINO [3]. Based on the statistics of our experiment runs, on a single NVIDIA RTX A5000 GPU (24GB VRAM), benchmarking a ViT-Base on *DepthCues* under our protocol and settings takes approximately 241 hours. This includes 92.7 hours for searching for the best layer for each cue (24 training + validation runs: 6 cues $\times$ 4 layers), and 148.3 hours for repeating the probing of the best layer (30 training + test runs: 6 cues $\times$ 5 repeats for statistical robustness). Note, these timings assume that the selected layer is the last one (*i.e.,* layer 12 for a ViT-Base model), and the total time can be lower than the estimate if this is not the case and the model is truncated up to the

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

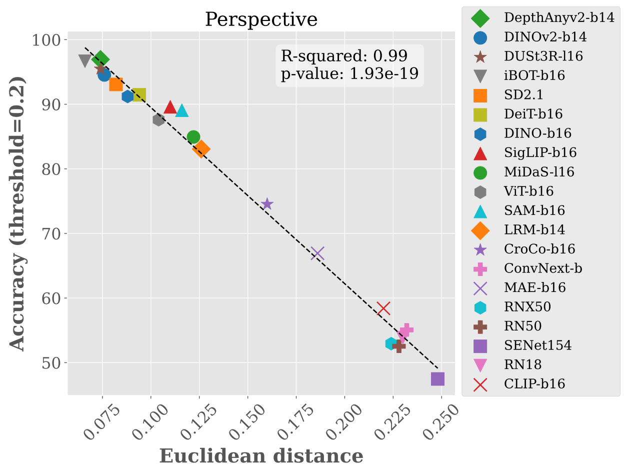

Figure A15. **Validating the threshold for evaluation of perspective.** We used a threshold of 0.2 to convert the Euclidean distance between ground-truth and predicted vanishing points to accuracy.

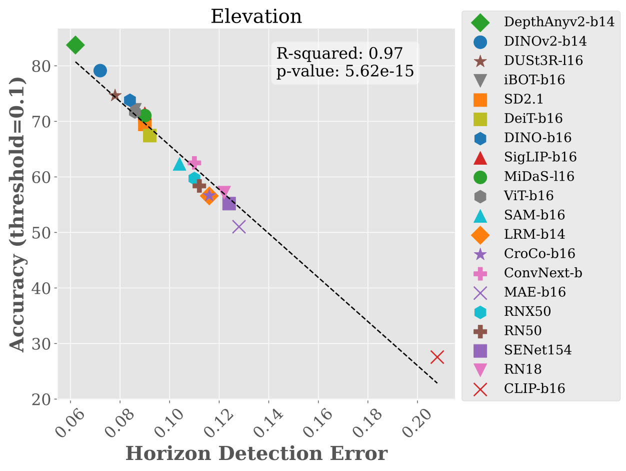

Figure A16. **Validating the threshold for evaluation of elevation.** We used a threshold of 0.1 to convert the horizon detection error to accuracy.

desired layer.

C.2. Evaluation Metrics

While we use accuracy to evaluate performance on the binary classification tasks, as discussed in the main paper we apply thresholding to the Euclidean distance (for perspective) and horizon detection error[2] [37] (for elevation) to convert these metrics into accuracy. Specifically, we used a threshold of 0.2 for perspective and 0.1 for elevation. We validate these choices of thresholds by evaluating the correlation between the original metrics and converted accuracies in Figs. A15 and A16.

[2]This measures the maximum distance between the predicted and the ground-truth horizon lines, normalized by the image height.

For evaluation of depth estimation on NYUv2, as in [2, 8], we use accuracy, which is calculated per image using the ground-truth and predicted depth maps, $d, \hat{d}$, as:

$$\text{Acc} = \frac{1}{N}\sum_{i=1}^{N}\max(\frac{\hat{d}_i}{d_i}, \frac{d_i}{\hat{d}_i}) < 1.25, \tag{1}$$

where N denotes the number of pixels in the image. For evaluation on DIW [4], we report the Weighted Human Disagreement Rate (WHDR), which measures the proportion of depth order predictions that disagree with the manual labels provided in DIW.

C.3. Fine-Tuning on *DepthCues*

To fine-tune the vision models on *DepthCues* we use Low Rank Adaptation (LoRA) [16]. Here, we specify our implementation for the fine-tuned models for DINOv2. We add LoRA to the query and value projection matrices of the self-attention block in the last layer. That is, the new query $\mathbf{q}$ and value $\mathbf{v}$ for a token $\mathbf{x}$ are obtained as

$$\mathbf{q} = W^Q\mathbf{x} + \Delta W^Q\mathbf{x}, \mathbf{v} = W^V\mathbf{x} + \Delta W^V\mathbf{x}, \tag{2}$$

$$\Delta W^Q = B^Q A^Q \qquad \Delta W^V = B^V A^V, \tag{3}$$

where $B^Q, B^V \in \mathbb{R}^{D\times R}$ and $A^Q, A^V \in \mathbb{R}^{R\times D}$. To make $\Delta W^Q, \Delta W^V$ the low-rank approximations of the original query and value projection matrices, we set R to 4, which is much smaller compared to the original feature dimension D (768 for DINOv2) of the model.

To train the models on *DepthCues* tasks, we attach a two-layer MLP with an intermediate GELU activation [14] and output neurons corresponding to the tasks. The models (LoRA weights and the MLP) are trained for around 15,000 iterations using AdamW [18] optimizer with cosine learning rate decay and a batch size of 32. The learning rate is set to 10^{-5} at the start and warms up to 10^{-3}.

D. Evaluated Models

Here we describe each model we evaluated in *DepthCues*, grouped by the supervision type. We summarized the architecture, supervision, and training dataset, along with the median rank (m-Rank) of each model in Tab. A6. The median rank is calculated based on the model's rank for each cue. For all models, we used publicly available checkpoints provided by the official codebases or from the timm [35] or transformers [36] Python libraries.

Categorization. The following models are trained for image classification problems with category labels using the ImageNet dataset [6]. ResNet18 (RN18) and ResNet50 (RN50) [12] are convolutional neural networks with residual connections. ResNext50 (RNX50) [38] and SENet (SENet154) [17] are extensions of residual convolutional networks, offering improved efficiency and scalability.

m-Rank	Model	Architecture	Supervision	Dataset (Size)
17.5	ResNet18 [12]	ResNet	Category	ImageNet (1.2M)
16	ResNet50 [12]	ResNet	Category	ImageNet (1.2M)
15	ResNext50 [38]	ResNet	Category	ImageNet (1.2M)
16.5	SENet [17]	ResNet	Category	ImageNet (1.2M)
11.5	ViT [7]	ViT-B/16	Category	ImageNet (14M)
6.5	DeiT III [30]	ViT-B/16	Category	ImageNet (14M)
12.5	ConvNext [20]	CNXT-B/16	Category	ImageNet (14M)
17.5	MAE [13]	ViT-B/16	Self-Supervised	ImageNet (1.2M)
5	iBOT [43]	ViT-B/16	Self-Supervised	ImageNet (14M)
9	DINO [3]	ViT-B/16	Self-Supervised	ImageNet (1.2M)
2	DINOv2 [22]	ViT-B/14	Self-Supervised	LVD (142M)
6.5	StableDiffusion [26]	UNet	Language	LAION (5B)
8	MiDaS [24]	ViT-L/16	Depth	MIX-6 (1.9M)
1	DepthAnythingv2 [40]	ViT-B/14	Depth	MIX-13 (0.5M+62M)
14.5	LRM [15]	ViT-B/14	Multi-View & 3D	Objaverse (10M), MVImgNet (6.5M)
14	CroCo [34]	ViT-B/16	Multi-View & Self-Supervised	Habitat (1.8M)
3.5	DUSt3R [32]	ViT-L/16	Multi-View & 3D	MIX-8 (8.5M)
11.5	SAM [19]	ViT-B/16	Segmentation	SA (1B)
20	CLIP [23]	ViT-B/16	Language	LAION (2B)
7.5	SigLIP [41]	ViT-B/16	Language	WebLI (18B)

Table A6. **Evaluated vision models.** We consider a range of publicly available large vision models that span several forms of supervision. In most cases, we select checkpoints of comparable model and training size. We also report the median rank (m-Rank) of each model, which is calculated based on the model's rank for each cue in *DepthCues*.

ConvNext (ConvNext-b) [20] is also designed with convolutional blocks, but its design choices are reconsidered based on the success of recent transformer-based [7] models. ViT (ViT-b16) [7] and DeiT III (DeiT-b16) [30] are transformer-based image models built on multi-head attention. We use the base configuration with 16 patch sizes for both models.

Depth. MiDaS (MiDaS-l16) [24] and DepthAnythingv2 (DepthAnyv2-b14) [40] are models trained with dense depth supervision. Both models use transformer architectures and are trained on a mix of datasets containing dense depth supervision for pixel values. The encoder network of DepthAnythingv2 is initialized with DINOv2 [22], while MiDaS is trained from scratch.

Segmentation. SAM (SAM-b16) [19] is trained on a large-scale segmentation dataset that provides dense pixel-level category information.

Language. CLIP (CLIP-b16) [23] and SigLIP (SigLIP-b16) [41] are trained to align the representations of images with their textual descriptions with a contrastive objective. StableDiffusion (SD2.1) [26] is a diffusion-based generative network that produces images conditioned on text descriptions. It is trained on a large-scale dataset[27] containing image-text pairs.

Multi-View. CroCo (CroCo-b16) [34] is trained with a cross-view completion objective using multi-view images, where the task involves predicting a patch of an image from another view. DUSt3R (DUSt3R-l16) [32] addresses the 3D reconstruction task for the generalized stereo case using neural networks through direct regression. It takes multi-view images as input and predicts dense 2D-3D point mappings for each view. LRM (LRM-b14) [15] is a large-scale 3D reconstruction network that takes images as input and outputs 3D representations. The network is trained with direct supervision using a large-scale 3D object repository [5].

Self-Supervised. MAE (MAE-b16) [13] and DINO (DINO-b16) [3] are trained with masked and contrastive self-supervised objective terms, respectively, without using any human-provided labels. iBOT (iBOT-b16) [43] and DINOv2 (DINOv2-b14) [22] combine masked and contrastive objectives to train networks. All of these methods are based on the transformer [7] architecture with the base configuration.

References

[1] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *TMLR*, 2024. Featured Certification. 4

[2] Shariq Farooq Bhat, Ibraheem Alhashim, and Peter Wonka. Adabins: Depth estimation using adaptive bins. In *CVPR*, 2021. 9

[3] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *ICCV*, 2021. 8, 10

[4] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. *NeurIPS*, 2016. 9

[5] Matt Deitke, Ruoshi Liu, Matthew Wallingford, Huong Ngo, Oscar Michel, Aditya Kusupati, Alan Fan, Christian Laforte, Vikram Voleti, Samir Yitzhafk Gadre, et al. Objaverse-xl: A universe of 10m+ 3d objects. *NeurIPS*, 2024. 10

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1, 9

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*, 2021. 10

[8] Mohamed El Banani, Amit Raj, Kevis-Kokitsi Maninis, Abhishek Kar, Yuanzhen Li, Michael Rubinstein, Deqing Sun, Leonidas Guibas, Justin Johnson, and Varun Jampani. Probing the 3d awareness of visual foundation models. In *CVPR*, 2024. 1, 4, 8, 9

[9] Alaaeldin El-Nouby, Michal Klein, Shuangfei Zhai, Miguel Angel Bautista, Alexander Toshev, Vaishaal Shankar, Joshua M Susskind, and Armand Joulin. Scalable pre-training of large autoregressive image models. In *ICML*, 2024. 4

[10] Yongtao Ge, Guangkai Xu, Zhiyue Zhao, Libo Sun, Zheng Huang, Yanlong Sun, Hao Chen, and Chunhua Shen. Geobench: Benchmarking and analyzing monocular geometry estimation models. *arXiv:2406.12671*, 2024. 4

[11] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 7

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 9, 10

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

[14] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv:1606.08415*, 2016. 9

[15] Yicong Hong, Kai Zhang, Jiuxiang Gu, Sai Bi, Yang Zhou, Difan Liu, Feng Liu, Kalyan Sunkavalli, Trung Bui, and Hao Tan. Lrm: Large reconstruction model for single image to 3d. *ICLR*, 2024. 10

[16] Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *ICLR*, 2022. 9

[17] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *CVPR*, 2018. 9, 10

[18] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 8, 9

[19] 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. In *ICCV*, 2023. 7, 10

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

[21] Wufei Ma, Guofeng Zhang, Qihao Liu, Guanning Zeng, Adam Kortylewski, Yaoyao Liu, and Alan Yuille. Imagenet3d: Towards general-purpose object-level 3d understanding. In *NeurIPS*, 2024. 4

[22] Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy V. Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel HAZIZA, Francisco Massa, Alaaeldin El-Nouby, Mido Assran, Nicolas Ballas, Wojciech Galuba, Russell Howes, Po-Yao Huang, Shang-Wen Li, Ishan Misra, Michael Rabbat, Vasu Sharma, Gabriel Synnaeve, Hu Xu, Herve Jegou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. 1, 10

[23] 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 *ICML*, 2021. 10

[24] 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. 10

[25] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 1

[26] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 10

[27] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *NeurIPS*, 2022. 10

[28] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 1

[29] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 7

[30] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *ECCV*, 2022. 10

[31] Ruicheng Wang, Sicheng Xu, Cassie Dai, Jianfeng Xiang, Yu Deng, Xin Tong, and Jiaolong Yang. Moge: Unlocking

accurate monocular geometry estimation for open-domain images with optimal training supervision. *arXiv preprint arXiv:2410.19115*, 2024. 7

[32] Shuzhe Wang, Vincent Leroy, Yohann Cabon, Boris Chidlovskii, and Jerome Revaud. Dust3r: Geometric 3d vision made easy. In *CVPR*, 2024. 10

[33] Tianyu Wang, Xiaowei Hu, Qiong Wang, Pheng-Ann Heng, and Chi-Wing Fu. Instance shadow detection. In *CVPR*, 2020. 7

[34] Philippe Weinzaepfel, Vincent Leroy, Thomas Lucas, Romain Brégier, Yohann Cabon, Vaibhav Arora, Leonid Antsfeld, Boris Chidlovskii, Gabriela Csurka, and Jérôme Revaud. Croco: Self-supervised pre-training for 3d vision tasks by cross-view completion. *NeurIPS*, 2022. 10

[35] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 9

[36] 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 *EMNLP System Demonstrations*, 2020. 9

[37] Scott Workman, Menghua Zhai, and Nathan Jacobs. Horizon lines in the wild. In *BMVC*, 2016. 9

[38] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017. 9, 10

[39] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 7

[40] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. *NeurIPS*, 2024. 10

[41] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 10

[42] Guanqi Zhan, Chuanxia Zheng, Weidi Xie, and Andrew Zisserman. A general protocol to probe large vision models for 3d physical understanding. In *NeurIPS*, 2024. 4

[43] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *ICLR*, 2022. 10

[44] Yan Zhu, Yuandong Tian, Dimitris Metaxas, and Piotr Dollár. Semantic amodal segmentation. In *CVPR*, 2017. 4, 7