Perception Encoder: The best visual embeddings are not at the output of the network

Daniel Bolya[1,*], **Po-Yao Huang**[1,*], **Peize Sun**[1,*], **Jang Hyun Cho**[1,2,*,†], **Andrea Madotto**[1,*], **Chen Wei**[1], **Tengyu Ma**[1], **Jiale Zhi**[1], **Jathushan Rajasegaran**[1], **Hanoona Rasheed**[3,†], **Junke Wang**[4,†], **Marco Monteiro**[1], **Hu Xu**[1], **Shiyu Dong**[5], **Nikhila Ravi**[1], **Daniel Li**[1], **Piotr Dollár**[1], **Christoph Feichtenhofer**[1]

[1]Meta FAIR, [2]UT Austin, [3]MBZUAI, [4]Fudan University, [5]Meta Reality Labs
[*]Joint first author, [†]Work done during internships at Meta

We introduce Perception Encoder (PE), a state-of-the-art encoder for image and video understanding trained via simple vision-language learning. Traditionally, vision encoders have relied on a variety of pretraining objectives, each tailored to specific downstream tasks such as classification, captioning, or localization. Surprisingly, after scaling our carefully tuned image pretraining recipe and refining with our robust video data engine, we find that contrastive vision-language training *alone* can produce strong, general embeddings for all of these downstream tasks. There is only one caveat: *these embeddings are hidden within the intermediate layers of the network.* To draw them out, we introduce two alignment methods, language alignment for multimodal language modeling, and spatial alignment for dense prediction. Together with the core contrastive checkpoint, our PE family of models achieves state-of-the-art performance on a wide variety of tasks, including zero-shot image and video classification and retrieval; document, image, and video Q&A; and spatial tasks such as detection, depth estimation, and tracking. To foster further research, we are releasing our models, code, and a novel dataset of synthetically and human-annotated videos.

Code: https://github.com/facebookresearch/perception_models
Dataset: https://ai.meta.com/datasets/pe-video/

Meta

1 Introduction

For the last decade in computer vision, pretrained vision encoders have been the core building block for most applications requiring *perception*. From million-scale ImageNet [24] pretrained convolutional networks [40, 59, 79, 120, 127] to billion-scale web-pretrained transformers [17, 22, 27, 31, 52, 99, 126, 147, 153], the dominant strategy in vision has consistently been to adapt large-scale pretrained encoders to downstream tasks.

There are many pretraining objectives today, each with distinct characteristics and each yielding representations better suited for specific tasks: vision-language contrastive losses [103, 155] learn a global vision and language embedding well-suited for zero-shot classification and retrieval as well as provide vision-language alignment for open-world [67, 91] and generative tasks [105, 111]; captioning losses [35, 133] learn to predict image descriptions using a language decoder, which transfers well to downstream multimodal language model (MLLM) tasks; and spatially self-supervised losses [42, 95] learn dense spatial correspondences without language supervision, making them useful for tasks requiring precise localization like object detection.

Many works are now attempting to combine two or more of these techniques in different ways [17, 32, 33, 35, 43, 107, 153]. While many have been successful, the complexity of these strategies grows exponentially with number of use cases, which can make scaling difficult. There has not yet been shown a *single, simple, and easily scalable* pretraining technique that can learn state-of-the-art features for all downstream tasks.

In this work we discover that *global vision-language contrastive learning alone* can be one such approach. After building a state-of-the-art contrastive model for image and video, we found a surprising result: *inside the model were specific features aligned to OCR, VQA, grounding, detection, depth estimation, and tracking.* Compared to the state-of-the-art models with captioning [35] and spatially self-supervised [95] pretraining, our contrastive encoder has specific layers that, when used as a frozen features, matches or exceeds the performance

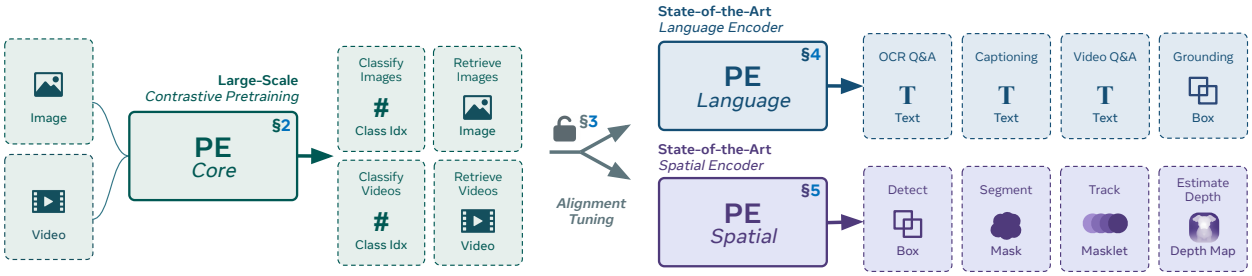


Figure 1 Perception Encoder (PE) is a family of large-scale vision encoder models with state-of-the-art performance on a large variety of vision tasks. By using a robust contrastive pretraining recipe and finetuning on synthetically aligned videos, PE not only outperforms all existing models on classification and retrieval (§2), but it also internally produces strong, general features that *scale* for downstream tasks (§3). PE unlocks the ability for large-scale contrastive pretraining to transfer to downstream tasks with alignment tuning to capitalize on those general features (§4, §5).

of the other two pretraining techniques *on tasks they should be the best at*. The only problem is—these features exist at *different layers* for each task. By exploiting this phenomenon with *alignment tuning*, we show it is possible to align these features to the end of the network in order to create state-of-the-art encoders for downstream MLLM and spatial tasks—all following the same easily scalable contrastive pretraining.

We begin by building PE_{core} (Fig. 1, left), a large-scale contrastively pretrained model with state-of-the-art zero-shot performance on *both* images and video (§2). To accomplish this, we first focus on developing a strong *image-only* contrastive pretraining recipe to extract general knowledge from billion-scale image-text data. Keeping the data and training FLOPs fixed, this recipe significantly improves upon vanilla CLIP in both absolute performance and robustness (§2.1). We then use the resulting model as a frame-based encoder to develop a *video* data engine for generating well-aligned video captions. Finetuning on this synthetic video-text data substantially improves performance on *both image and video* classification and retrieval tasks (§2.2). Motivated by this success, we release a large portion of the data used to train the engine: PE Video Dataset (PVD), consisting of 1M diverse videos with 120K human-refined annotations (§2.3). Finally, we scale our robust image pretraining and well-aligned video finetuning strategy to 2B parameters to produce $\text{PE}_{\text{core}}\text{G}$ (§2.4), a single unified encoder that outperforms SigLIP2 [134] on zero-shot image tasks and InternVideo2 [141] on most zero-shot video tasks. We further transfer this power to smaller model scales through distillation.

With the strongest image and video recognition model in hand, we shift our focus to downstream tasks. Remarkably, despite being pretrained with CLIP loss, we find that the *intermediate layers* of $\text{PE}_{\text{core}}\text{G}$ can rival AIMv2-3B [35] on language tasks and DINOv2-g [95] on spatial tasks, both of which among the strongest pretrained models in their respective domains. Upon investigation, we attribute this capability to our robust image pretraining strategy, which appears to have unlocked the potential of contrastive pretraining to scale effectively for downstream tasks (§3). However, a challenge remains: the model does not naturally output these features, keeping them hidden internally. To address this, we introduce two *alignment tuning* methods (Fig. 1, right) to extract these strong, general features.

First, in §4, we investigate the most effective technique to align features to the end of the network by adapting to a large language model. This *language alignment* enables us to construct $\text{PE}_{\text{lang}}\text{G}$, which individually outperforms all other popular vision encoders for MLLM tasks. Moreover, when paired with our Perception Language Model (PLM) [19], the combination rivals the latest state-of-the-art MLLMs, like InternVL3 [162].

Second, in §5, we identify a dichotomy in the layers optimal for spatial tasks. By visualizing the features and pinpointing the explicit reason for this dichotomy, we develop a straightforward *spatial alignment* approach: distilling *from the model's own frozen features* to achieve most of the alignment, complemented by a novel use of SAM 2 [108] for *spatial correspondence* distillation to refine the process. The resulting $\text{PE}_{\text{spatial}}\text{G}$ not only outperforms other popular models in depth estimation, tracking, and semantic segmentation, but also matches the absolute state-of-the-art performance on COCO [74] detection with a much simpler decoder.

With this family of checkpoints, Perception Encoder unlocks the potential to scale one simple pretraining method to solve many downstream vision tasks. We are releasing our models, code, and PE Video Dataset.

2 Perception Encoder: Core

To build Perception Encoder (PE), we start by training a large-scale, robust, and highly performant vision-language contrastive model for image *and video*. We have two objectives: first, to enhance the scalability and data efficiency of contrastive training; and second, to create a unified model effective on both image and video.

These goals are somewhat conflicting: image-text data is plentiful and training on images is efficient, but video-text data is scarce and video training is expensive. Thus, we decouple image and video training into two stages. We first develop a strong *image* pretraining recipe (§2.1) with several regularization techniques to create a robust starting point. Then we use the resulting image model as a frame encoder to develop a *video data engine* (§2.2) supported by our novel human-refined video-text dataset (§2.3) to generate aligned captions for video clips. Finally, we finetune the image encoder on the resulting aligned video data (§2.4). Using our data engine design, this short finetuning step substantially improves *both* image and video performance.

2.1 Robust Image Pretraining

In the first stage of pretraining, we want to learn as much visual information as possible from large set of image-text data. Notably, a unique quirk of contrastive training is the loss for a given sample depends on the other samples in the batch. Because each batch is different, there is potential to learn new information every time an example is sampled, even if that sample has been seen before. Thus, we find contrastive learning to benefit from a long training schedule. To exploit this, we design our pretraining recipe with high regularization, stability, and training efficiency in mind.

Setup. (Fig. 2.1) We track our changes on a vanilla CLIP model using an OpenCLIP [49] ViT-L/14 model at 224 resolution as a baseline. We keep the training budget fixed to around 1T GFLOPs (*i.e.*, a ZFLOP), and train on a fixed 2.3B image-text dataset curated using the MetaCLIP [147] text-only curation pipeline. For the baseline, we use a global batch size of 32K, class token, AdamW [81], and train for 12B samples seen. To assess the *generality* of the information learned during pretraining, we report not only zero-shot ImageNet val [24] results but also the average performance across a range of robustness metrics, including ImageNet val [24], ImageNet v2 [109], ObjectNet [4], ImageNet Adversarial [45], ImageNet Rendition [44], and ImageNet Sketch [138]. Like observed with other pure CLIP models [31, 103, 147], the average robustness metric performance of this vanilla recipe is much lower than ImageNet val alone.

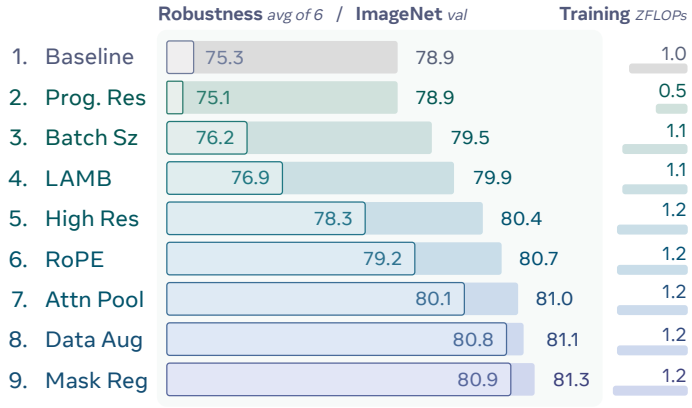


Figure 2 Robust Image Pretraining. We tune our pretraining recipe (§2.1) to maximize performance on a fixed set of data, starting with an OpenCLIP [49] ViT-L/14 model. We report cumulative zero-shot classification results for each modification. The inner bars show robustness evaluation, calculated as the average of 6 robustness benchmarks [4, 24, 44, 45, 109, 138], and the outer bars show ImageNet val [24] alone. Several changes significantly improve robustness, indicating that ImageNet val scales more with data, while robustness can scale with refined training techniques.

Progressive Resolution. (Fig. 2.2) To enable longer training, we first improve training efficiency. As shown in many works [68, 69, 77, 127, 132], vision encoders work well with a *progressively increasing* resolution schedule. Thus, we *halve* the training FLOPs while maintaining performance by evenly splitting the baseline 12B-sample run into 98, 154, and 224 resolution stages, with 4B samples per stage.

Increasing Batch Size. (Fig. 2.3) We use the extra budget to double the batch size from 32K to 64K, increasing the total samples seen from 12B to 24B. Larger batch size means a higher likelihood for there to be a non-trivially novel pair of samples, *i.e.*, hard negatives. This is akin to increasing the "task difficulty" of CLIP and improves ImageNet val by +0.6% and robustness by double of that, +1.1%.

LAMB Optimizer. (Fig. 2.4) We switch from AdamW to LAMB [151], which is known to stabilize large batch training. More importantly, LAMB allows us to train stably with a higher learning rate of 2×10^{-3} compared

to the original 5×10^{-4}. We observe that starting with a high learning rate is important to allow the model to adapt to different resolutions. These factors combine for +0.4% on ImageNet val and +0.7% on robustness.

Increasing Final Resolution. (Fig. 2.5) A classic finding is that parameters and resolution should be scaled together [34, 127]. Thus, we add a fourth 336 resolution stage at the end of training. To keep the training FLOPs the same, we adjust the training schedule to 10B samples at 98 resolution, 8B at 154, 4B at 224, and 2B at 336. While ImageNet val only increases by +0.5%, robustness improves threefold, rising by +1.4%.

RoPE. (Fig. 2.6) We add 2D RoPE [123] to each attention layer to improve extrapolation, keeping the original position embedding. 2D RoPE only improves ImageNet val by +0.3% but enhances robustness by +0.9%.

Attention Pooling. (Fig. 2.7) We follow [155] in constructing the CLIP embedding using an attention probing transformer block. Surprisingly, we found keeping the class token as an input to this block is important for small model performance. Together, this improves ImageNet val by +0.3% and robustness by +0.9%.

Tuned Data Augmentation. (Fig. 2.8) Despite training on billions of samples, we find data augmentation still important—especially for transfer to unlikely scenarios like in ObjectNet [4]. We add heavy random cropping, brightness/saturation jitter, and horizontal flip. Random cropping encourages using the entire caption, as not everything is in frame. Jitter helps low-light settings and documents. Horizontal flip improves natural images and does not hurt OCR (see §2.5). These improve robustness by +0.7%, notably, ObjectNet by +2.4%.

Mask Regularization. (Fig. 2.9) As regularization, we want the model to produce the same features if some patches are not visible. However, passing the CLIP gradients through masked images may negatively alter behavior on unmasked images. Thus, we convert MaskFeat [142] into a regularization loss by duplicating and masking 1/16th of the batch. At the output, the masked tokens are aligned to their unmasked counterparts by maximizing cosine similarity. Care is taken to ensure that the CLIP and masked gradients are disjoint.

Scaling Behavior. (Figs. 3 and 4) In Fig. 3, we show the performance of our recipe (Fig. 2.9) *vs.* the original CLIP recipe (Fig. 2.1) across S/14, B/14, and L/14 models. For each benchmark, our recipe scales around the same rate or better than the original CLIP recipe. On some difficult datasets like ObjectNet [4] and ImageNet Adversarial [45], our recipe shows distinctly better scaling. This indicates that the improvements in performance were not at the cost of scalability, meaning we can further benefit from scaling the model size.

Figure 3 Scaling Behavior (Model Size). Results before and after our recipe changes (Fig. 2) for S/14, B/14, and L/14 models. Our recipe improves scaling for difficult metrics like ObjectNet [4] and ImageNet Adeversarial [45].

In Fig. 4, we additionally show the performance of our recipe *vs.* the original CLIP recipe across L/14 models trained with 120K steps (one-third schedule), 240K steps (two-thirds schedule), and 360K steps (full ablation schedule). All models are their own training runs with full learning rate annealing and the progressive resolution schedule adjusted proportionally. We see nearly linear trends for our recipe on most datasets. This suggests we can train longer for more performance, even at L scale and with 24B samples seen already.

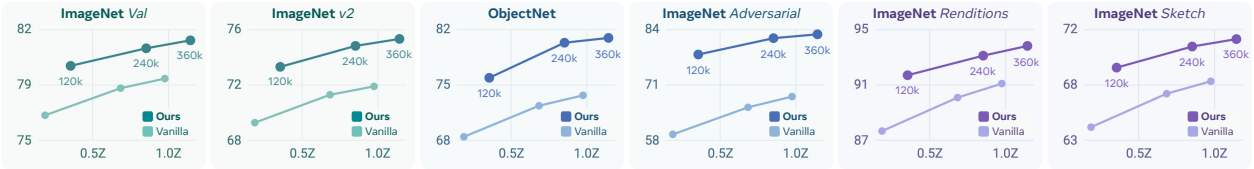


Figure 4 Scaling Behavior (Training Steps). Results before and after our recipe changes for an L/14 model trained with 120K, 240K, and 360K steps, adjusting the learning rate and progressive resolution schedules accordingly. Despite our recipe being much stronger than the original, there is still room for further improvement by training longer.

2.2 Bootstrapping a Video Data Engine with Perception Encoder

With a robust image pretraining recipe settled and its scaling behavior confirmed, our next step is to extend the image-only encoder to accommodate video and build a unified image-video model. Unlike web-scale image-text data, which comes in many cases with human-generated descriptive alt-text information, videos with aligned language annotation are inherently scarce. High-quality human-annotated captions for videos are even rarer. This scarcity presents a unique and significant challenge in training encoders capable of effectively processing video inputs. Inspired by the recent success of image data engines [56, 62, 93, 108, 146], we extend this concept to develop a robust video data engine that generates well-aligned synthetic captions for a diverse set of videos, facilitating the training of a video encoder. This innovative approach represents the first large-scale exploration of its kind. In the following sections, we introduce the process of building our video data engine.

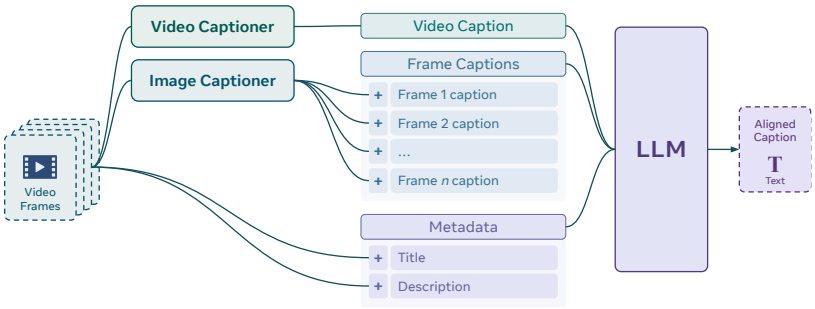


Figure 5 Video Data Engine. To create aligned video-text data for contrastive training, we use a PE-based video captioner [19] to generate a holistic video caption and an image-level captioner [80] on sampled frames. We then provide those captions as well as the original video metadata to text-only LLM [80] to synthesize a single short, aligned caption optimal for contrastive training.

To bootstrap our contrastive video finetuning, we focus on synthesizing video captions. We build our data engine in three stages: (1) we create a strong baseline video captioner, which we call the Perception Language Model (PLM), described in [19]; (2) we add additional high quality video data with human-refined captions to further enhance the captioner's quality; (3) we refine and summarize the generated video captions with an LLM to construct a large video dataset to use for the contrastive video finetuning of our Perception Encoder.

Phase 1: Base Video Captioner (PLM). We build our data engine on an early version of PLM [19], a multimodal large language model with PE as the vision encoder and Llama [80] as the language decoder. We train PLM on a large-scale collection of open-access image and video datasets [19]. In total, the training dataset consists of 64.7M images and videos covering natural images, charts, documents, exocentric and egocentric videos.

Phase 2: PLM + Refined Data. To further boost captioning performance, we collect a set of 265K videos (105K from PVD which we release, see §2.3), caption them with our base PLM model, and ask human raters to refine the captions[1]. We then finetune our base PLM model with this data, significantly improving captioning quality (see Tab. 1).

	AuroraCap [12]		VCG Diverse [85]		VCG Bench [84]
Captioner	*Score*	*Acc*	*Score*	*Acc*	*Score*
PLM	2.2	51.9	3.1	65.1	34.3
PLM + Human-Refined Data	**3.4**	**71.1**	**3.6**	**79.4**	**35.2**

Table 1 Video Captioning. We use an early version of PLM-8B [19], consisting of our image-only PE encoder and a Llama decoder, for captioning. Adding human-refined data significantly boosts captioning performance (higher is better).

Phase 3: LLM Summarization. We synthesize the final aligned video captions by incorporating the PLM video captions, Llama 3.2 [80] image-only frame captions, and the existing video metadata of video titles and descriptions (Fig. 5). Similar to image alt-text, video metadata contains knowledge often not covered by the image and video captioning models. Thus, combining the two leads to more comprehensive captions. We summarize video captions, frame captions, and video metadata together using the Llama 3.3 70B model to provide the final captions. The prompt used to generate the summary can be found in Appendix A.1.

Using the Engine. Finally, we use the resulting data engine bootstrapped with an image-only checkpoint of PE to generate well-aligned, information-dense captions for a diverse set of 22M videos for contrastive finetuning.

Training with Recaptioned Videos. Our goal is to develop a unified image *and* video encoder. To encode videos

[1]The annotators are instructed to remove, correct, and add information from the captions.

using our existing image encoder, we uniformly sample $N = 8$ frames from video clips and extract frame-level embeddings with the image encoder. We then apply average pooling over these frame embeddings to obtain video embeddings, which are used for contrastive learning with encoded video captions by the text encoder. Despite being extremely simple, we find this technique surprisingly effective in producing a strong joint image-video encoder. We share this finding with previous studies [17, 82], which note that simple average pooling outperforms more complex pooling strategies like attention-based compression for video.

Ablations. In Tab. 2, we conduct an ablation study on the components of the video data engine by finetuning an intermediate image-only checkpoint on 17M of the 22M videos recaptioned by our video data engine. The results show that the video data engine significantly enhances zero-shot classification and retrieval performance for both image and video benchmarks, compared to the image-only baseline encoder (first row). Notably, using the video data engine's video-level and frame-level captions provides significant improvements over relying solely on metadata such as video title and description (second row), highlighting the importance of building a robust video data engine to compensate noise in web videos. Our analysis reveals that the most critical components are the video metadata and PLM's video caption; however, all components are necessary to achieve peak performance in our video data engine.

				Image Zero-Shot						*Video Zero-Shot*				
Title	Description	Video Caption	Frame Caption	**Average Image**	ImageNet *val* [24]	ImageNet *v2* [109]	ObjectNet *IN Classes* [4]	MS-COCO *txt→img* [74]	MS-COCO *img→txt* [74]	**Average Video**	Kinetics *400* [53]	Kinetics *600* [53]	MSR-VTT *txt→vid* [148]	MSR-VTT *vid→txt* [148]
				72.6	83.3	77.8	85.8	49.4	66.8	50.9	69.7	68.4	38.0	27.3
✓	✓			75.4	83.2	78.2	87.1	47.3	66.0	56.0	74.1	73.5	39.0	37.3
✓	✓	✓		78.2	83.5	78.4	86.8	56.0	74.3	60.9	73.8	73.4	47.6	48.8
✓	✓	✓*		78.1	83.7	79.0	87.7	54.1	73.0	60.9	75.4	75.1	46.7	46.5
✓	✓	✓	✓	78.2	83.7	79.0	87.5	54.6	73.2	61.6	75.8	75.5	47.4	48.1

Table 2 Video Data Engine Ablation. We ablate our video data engine in Fig. 5 by finetuning on an in-development image-only version of PE by averaging the frame embeddings to create a single video CLIP embedding. Video captions are generated by PLM trained with or without* human-refined data (see §2.3). Frame captions are generated by the Llama 3.2 vision model. Each component helps on different metrics, overall culminating in a huge boost to *both* image and video zero-shot performance.

In Fig. 6, we investigate the impact of scaling recaptioned video data on a later checkpoint of the same image-only model as in Fig. 2. Notably, scaling synthetic video data demonstrates consistent improvement in both image and video benchmarks. Full results of this scaling experiment can be found in the Appendix 19.

In the top row, scaling synthetic video data consistently improves performance on image benchmarks, with monotonic improvements of +1.1% in ObjectNet and +1.6% in ImageNet Adversarial. ImageNet val and ImageNet v2 have smaller gains, with accuracy increases of 0.3% to 0.5%, plateauing at ∼7M samples. We also observe a significant boost to zero-shot retrieval (here, COCO [74]) of +3.8% to +4.1% top-1 recall.

The video tasks listed in the bottom row demonstrate a consistent story. We observe a significant jump in performance between none and 3M videos across all video classification tasks, indicating that there is a domain gap for image-only models that hinders their ability to perform well on video out of the box. Further scaling synthetic video data leads to substantial performance gains in both video classification and retrieval. Video classification accuracy improves consistently by +5.6% to +11.7% without plateauing, while video

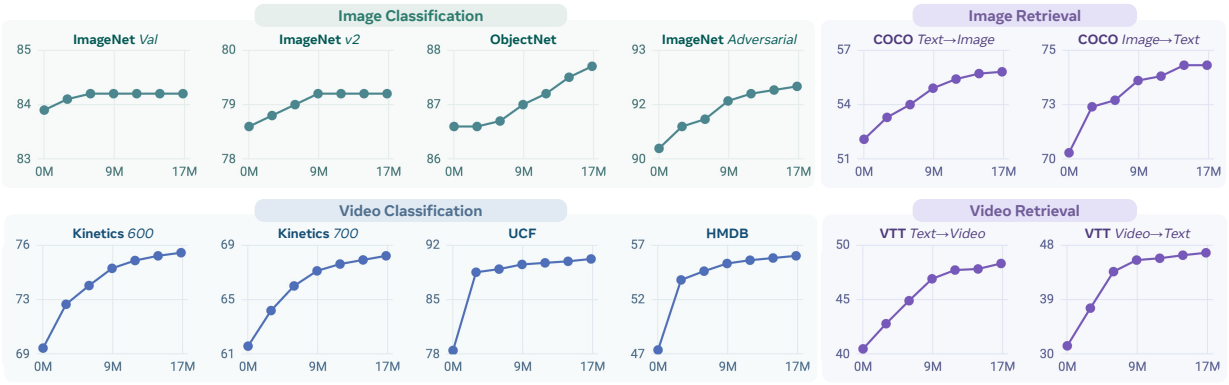
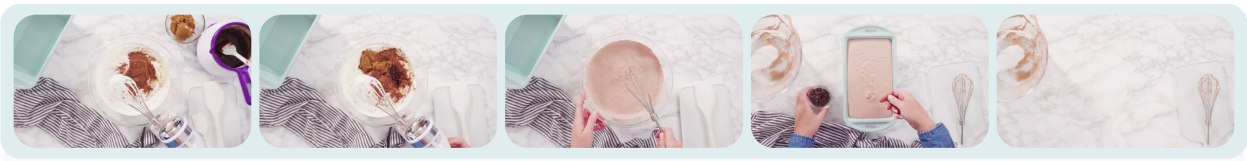


Figure 6 Video Data Scaling. Finetuning on videos recaptioned by the PE video data engine from 0M (baseline image-only model) to 17M samples consistently improves both image and video performance, both classification and retrieval.



Category: *Food Preparation*

Model caption: "The video shows a person making a cake. The person is seen mixing ingredients in a bowl, pouring the mixture into a pan, and adding chocolate chips. The video also shows the person using a whisk and a spatula to mix the ingredients and pour the mixture into the pan. The person is wearing a blue shirt and is seen holding a glass bowl and a spoon. The background of the video is a white marble countertop, and there is a striped towel on the counter."

Human refinement: "The video shows a person making a chocolate cake. The person mixes ingredients in a glass bowl and pours the mixture into a baking pan. Then, the person adds chocolate chips on top. The video also shows the person mixing the ingredients with an electric whisk and scraping the mixture into the pan with a spatula. The person is wearing a blue shirt and is seen holding a glass bowl. The background of the video is a white marble countertop, and there is a striped towel on the counter."

Figure 7 PE Video Dataset Example. A sample from PVD, our released video-text dataset. Initial captions are generated by our video captioning model and then refined by human annotators. Annotators are instructed to add details and remove model hallucination. In this example, the model hallucination "a spoon" is removed; and more details such as "glass bowl" and the action "scraping" are added. See Appendix Fig. 18 for more.

retrieval shows significant improvements of +7.7 to +15.3 top-1 recall.

These experiments highlight the quality of our video data engine and its ability to significantly improve encoder performance, even with only a relatively modest 17M videos compared to the billions of images seen during pretraining. Our video data engine is a vital component in build a strong, unified image-video encoder.

2.3 PE Video Dataset (PVD)

For the benefit of the community, we release a new video dataset: PE Video Dataset (PVD). PVD comprises of 1M high-quality and diverse videos with accompanying tags and descriptions. The videos are motion-centered, covering both first-person and third-person views with a wide coverage of scenes.

We additionally select 120K of these videos with the highest degree of motion to annotate with detailed captions by generating synthetic captions using our video captioner (§2.2) and employing 200 annotators to verify and refine them. We ask the human annotators to improve the synthetic captions by removing any hallucinations, correcting words that describe the video inaccurately, eliminate repetitive or redundant words to make the caption more concise, and add any missing actions being performed in the video.

We release two versions of annotations for the 120K PVD subset: (1) Human verified captions: extended summaries with an average length of 57.1 words that provide a high-level description of each video. These captions are suitable for CLIP-style training. (2) Long automated captions: detailed and fine-grained descriptions with an average length of 111.7 words that capture spatial and temporal events. These captions are ideal for fine-grained video understanding.

Videos	998,862
Human Captions	118,862
Total Duration	4625 hrs
Duration (s)	16.7±9.8
Human Caption Length	57.1±25.4
Model Caption Length	111.7±43.2

Table 3 PVD Statistics.

In Fig. 7, we visualize a video example together with their model and human captions from PE Video Dataset (See Fig. 18 for more). The dataset statistics are summarized in Tab. 3. Finally, We use 105K of these refined samples to improve the data engine (§2.2 phase 2) and 15K as a high-quality video retrieval benchmark.

PVD Benchmark. We use 15K of the human-refined video-caption pairs as a held-out test set, which we introduce as a new video retrieval benchmark, PVD Benchmark, to evaluate finegrained video-caption alignment. We follow the format of MSR-VTT [148] to construct the benchmark. We select videos from 10 different categories, including hand actions, object interactions, food preparation, work activities, outdoor scenes, animals, water scenes, object handling, close-up shots, and nature scenes, with an overall average caption length of 51.7 words (see Appendix A.2.3 for statistics). We use PVD Benchmark evaluate SigLIP [155], SigLIP2 [134], InternVL [17], and PE models, and the results can be found in Tab. 7.

2.4 A Unified Encoder for Image and Video

Using a robust, scalable image pretraining recipe and video-pretraining data recaptioned by the proposed video data engine, in this section we present **PE$_{\text{core}}$**, a unified image-and-video encoder.

Model Architecture. To capitalize on the promising scaling behavior observed in §2.1, we scale the largest PE$_{\text{core}}$ model to 2B parameters[2] (G scale). Tab. 4 shows the detailed model configuration of the vision and text transformers and the dimension of the output clip embedding space.

Scale	Tower	Params	Width	Depth	MLP	Heads	CLIP Dim
B	Vision	0.09B	768	12	3072	12	1024
	Text	0.31B	1024	24	4096	16	
L	Vision	0.32B	1024	24	4096	16	1024
	Text	0.31B	1024	24	4096	16	
G	Vision	1.88B	1536	50	8960	16	1280
	Text	0.47B	1280	24	5120	20	

Table 4 PE Model Configurations.

Smaller Model Distillation. To maximize the performance of smaller models (B and L scales in Tab. 4), we employ a distillation finetuning approach [47] using PE$_{\text{core}}$G as the teacher. This process involves a short finetuning schedule where both the student and teacher models encode image and text inputs separately to compute image-to-text and text-to-image similarity distributions, similar to CLIP training [103]. The student's distributions are then optimized to match those of the teacher by minimizing KL-divergence, distilling multimodal relational knowledge from the teacher into the student.

Notably, we find that using a smaller softmax temperature for the teacher's distributions, specifically $0.5\times$ the temperature used for the student's distribution, significantly enhances the effectiveness of knowledge distillation. By leveraging the strong embeddings provided by PE$_{\text{core}}$G, our short distillation finetuning schedule significantly boosts the performance of both B and L scale models of PE$_{\text{core}}$ (see Appendix C.3).

Model Training. The training process of PE$_{\text{core}}$ involves three stages:

1. *Image pretraining.* We scale up image pretraining to 5.4B publicly available image alt-text pairs curated with MetaCLIP [147] and a total of 86B samples seen to ensure convergence (58B for B and L). We use a global batch size of 131K, with progressive resolution from 98 to up to 448 depending on the model.
2. *Image and video finetuning.* Following the initial pretraining, we subsequently finetune the model at max resolution with a short schedule for 50M samples on the image pretraining data (as cooldown) followed by 22M samples on the recaptioned videos with a smaller learning rate and batch size. The video captions are produced using the proposed video data engine (§2.2). For each video clip, we uniformly sample 8 frames, encode them, take their average to produce a single video embedding, and align them with the corresponding video captions using the same contrastive objective in image training.
3. *Smaller model distillation.* We distill the 2B model (G scale) into smaller contrastive pretrained models at B and L scales under their final resolutions, using a short schedule that covers approximately 4B samples seen (∼8% of the pretraining schedule) with a lower learning rate and no weight decay.

The detailed training configuration and setups are listed in Appendix B.1.1.

2.5 Core Results

Zero-Shot Image Results. In Tab. 5, we present PE$_{\text{core}}$'s performance on zero-shot image benchmarks for classification and retrieval *vs.* the strongest existing models, including SigLIP2 [134] and proprietary models using JFT-3B [27], which is likely tuned for ImageNet. PE$_{\text{core}}$ outperforms all other contrastive models across the board on all zero-shot tasks, including the highly competitive average of zero-shot ImageNet robustness metrics [4, 24, 44, 45, 109, 138]. This marks a significant achievement, as we are the first accomplish this in over 3 years without access to Google's internal JFT-3B [27] or WebLI [15] datasets. And *at the same time*, PE$_{\text{core}}$ also exceeds the existing state-of-the-art on image-text retrieval and significantly improves on fine-grained classification—the first to simultaneously hold state-of-the-art on all common zero-shot categories.

By harnessing the power of our video data engine, training with a relatively small dataset of 22M videos and their corresponding synthetic captions leads to substantial *gains in image benchmarks*, with average general image classification improving by +0.6% with emphasis on more difficult benchmarks (notably +1.2%

[2]We employ the setup described in §2.1 except for the additional class token (only used for L and B). Interestingly, we find *using the same high learning rate* (2×10^{-3}) to perform well for G. We also did not find scaling the text encoder to be beneficial.

Model	Encoder Params	Resolution	Data	Zero-Shot Classification								Zero-Shot Fine-Grained Classification									Zero-Shot Retrieval				
				Avg Class.	ImageNet *val* [24]	ImageNet *v2* [109]	ObjectNet *IN Classes* [4]	ImageNet *Adversarial* [45]	ImageNet *Renditions* [44]	ImageNet *Sketch* [138]	**Avg Fine.**	Food *101* [8]	Flowers *Oxford* [94]	Pets *Oxford* [97]	Cars *Stanford* [57]	Aircrafts *FGVC* [86]	Countries *211* [129]	Scenes *SUN397* [145]	Satellite *RESISC* [18]	**Avg Retrieval**	MS-COCO *txt→img* [74]	MS-COCO *img→txt* [74]	Flickr-30k *txt→img* [152]	Flickr-30k *img→txt* [152]
Proprietary																									
BASIC [99]	2.4B	224	6.6B	84.3	85.7	80.6	82.3	85.6	95.7	76.1	-	95.1	91.2	97.9	-	-	-	76.2	72.7	-	-	-	-	-
CoCa [153]	1.0B	576	4.8B	85.7	86.3	80.7	82.7	90.2	96.5	77.6	-	-	-	-	-	-	-	-	-	72.6	51.2	66.3	80.4	92.5
LiT-22B [22]	21.7B	224	15B	-	85.9	80.9	87.6	90.1	96.0	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-
B Scale																									
SigLIP-B/16† [155]	0.1B	224	10B	69.9	76.2	69.5	70.7	45.1	90.2	67.9	69.5	91.6	85.2	94.2	90.8	44.0	15.9	70.0	64.6	69.8	47.2	64.5	77.9	89.6
SigLIP2-B/16† [134]	0.1B	224	10B	73.1	78.2	71.4	**73.6**	55.0	**91.7**	**68.9**	73.1	**92.8**	85.7	**95.4**	**93.4**	54.8	19.2	72.7	71.1	73.7	**52.1**	68.9	80.7	93.0
PE$_{\text{core}}$B	0.1B	224	5.4B	**73.2**	**78.4**	**71.7**	71.9	**62.4**	88.7	66.1	**75.0**	92.5	**86.5**	94.6	92.1	**57.0**	**30.5**	**74.0**	**72.7**	**74.3**	50.9	**71.0**	**80.8**	**94.4**
L Scale																									
SigLIP-L/16† [155]	0.3B	384	10B	80.7	82.1	75.9	80.9	76.5	95.0	73.6	74.4	95.6	89.4	**96.8**	94.8	53.2	24.7	72.5	67.9	74.7	52.8	70.5	82.6	92.9
SigLIP2-L/16† [134]	0.3B	384	10B	83.3	83.1	77.4	84.4	84.3	**95.7**	**75.5**	78.4	96.1	**90.0**	96.4	**95.8**	67.0	31.6	74.8	75.5	76.7	55.3	71.4	85.0	95.2
PE$_{\text{core}}$L	0.3B	336	5.4B	**83.9**	**83.5**	**77.9**	**84.7**	**89.0**	95.2	73.4	**80.0**	**96.2**	87.2	96.4	93.7	**67.8**	**45.6**	**77.4**	**75.7**	**78.8**	**57.1**	**75.9**	**85.5**	**96.6**
Unbounded Scale																									
DFN-H+† [31]	0.6B	378	5B	81.6	84.3	78.3	79.6	79.6	93.6	73.3	80.5	96.2	**91.6**	96.8	**96.0**	72.5	37.9	77.4	75.9	75.8	55.6	71.8	82.1	93.6
InternVL-C [17]	5.5B	224	5B	82.5	83.2	77.3	80.6	83.8	95.7	74.3	76.4	95.3	85.8	96.3	94.4	53.3	35.1	76.3	74.4	78.6	**58.6**	74.9	85.0	95.7
EVA 18B [126]	17.5B	224	2B	83.6	83.8	77.9	82.2	87.3	95.7	74.7	78.8	95.8	86.0	96.1	94.9	59.7	43.1	77.7	**76.9**	77.5	56.2	73.6	83.3	**96.7**
EVA 18B+ [126]	17.5B	336	2B	84.1	83.9	78.2	83.6	88.9	95.6	74.3	-	-	-	-	-	-	-	-	-	-	-	-	-	-
SigLIP2-g-opt† [134]	1.1B	384	10B	86.2	85.0	79.8	88.0	90.5	**96.6**	**77.4**	81.0	**97.0**	91.5	**97.8**	95.9	73.6	40.1	76.3	75.9	78.0	56.1	72.8	**86.0**	95.4
PE$_{\text{core}}$G *(image only)*	1.9B	448	5.4B	86.0	85.2	**80.2**	87.1	91.2	96.1	76.1	82.7	96.6	91.0	96.4	94.6	76.7	57.3	77.5	71.8	74.9	53.1	70.9	81.6	93.9
PE$_{\text{core}}$G	1.9B	448	5.4B	**86.6**	**85.4**	**80.2**	**88.2**	**92.6**	96.5	76.5	**83.7**	96.9	91.4	96.9	94.7	**78.2**	**57.6**	**78.5**	75.8	**78.9**	58.1	**75.4**	85.7	96.2

Table 5 Zero-Shot Image Results. Image zero-shot performance of PE$_{\text{core}}$ compared to the state-of-the-art for *both* proprietary and open models. PE$_{\text{core}}$G is the first vision encoder to outperform the best models trained on the proprietary JFT-3B [27] and WebLI [15] on general classification. Moreover at all model sizes, PE$_{\text{core}}$ obtains state-of-the-art results across general classification, retrieval, and finegrained classification. †Re-evaluated: DFN by [126]; SigLIP and SigLIP2 by us with the same benchmark settings if not reported in [134] (see Appendix B.1.2).

ObjectNet, +1.4% ImageNet Adversarial) and fine-grained classification by +1.0% on average. Furthermore, due to the high level of detail and alignment of our synthetic captions, zero-shot retrieval is significantly boosted by +3.6% on average. These results emphasize that training with well-aligned video text data does not just improve video performance—it creates a strictly better model for both videos *and* images.

Zero-Shot Video Results. We assess the performance of PE$_{\text{core}}$ on zero-shot video benchmarks by employing the same model as a frame-based video encoder, utilizing 8 uniformly sampled frames, as described in §2.2.

We present the corresponding video results in Tab. 6. Our base image encoder already outperforms all other image-only encoders on both zero-shot classification and retrieval, including SigLIP2-g-opt. With video finetuning, PE$_{\text{core}}$G significantly outperforms even native video models that use full temporal attention on video classification, and nearly matches the state-of-the-art on video retrieval using a simple frame-level encoder. This result underscores the importance of our video data engine, resulting in +3.9% on average zero-shot video classification, and a massive +11.1% on retrieval. Moreover, PE$_{\text{core}}$ does this with much less video data compared to other video-based approaches like InternVideo2 [141] and VideoPrism [159], highlighting the benefits of a joint image-video encoder.

Model	Encoder Params	Resolution	# Frames	Video Data	Zero-Shot Classification						Zero-Shot Retrieval						
					Avg Class.	Kinetics *400* [53]	Kinetics *600* [53]	Kinetics *700* [53]	UCF *101* [122]	HMDB *51* [60]	**Avg Retrieval**	MSR-VTT *txt→video* [74]	MSR-VTT *video→txt* [74]	MSVD *txt→video* [152]	MSVD *video→txt* [152]	ActivityNet *txt→video* [152]	ActivityNet *video→txt* [152]
B Scale																	
CLIP [103]	0.1B	224	8	n/a	54.3	58.4	55.1	46.1	68.9	43.2	29.2	30.4	24.2	40.5	57.2	9.1	13.2
CLIP4CLIP [82]	0.1B	224	12	n/a	-	-	-	-	-	-	-	32.0	-	38.5	-	-	-
SigLIP2-B/16† [134]	0.1B	224	8	n/a	57.3	58.7	55.0	48.4	82.0	42.3	39.9	38.5	30.1	49.0	67.2	28.6	25.8
PE$_{\text{core}}$B	0.1B	224	8	22M	**63.9**	**65.6**	**65.1**	**55.8**	**84.6**	**48.2**	**49.9**	**47.6**	**47.3**	**50.4**	**76.7**	**39.0**	**38.4**
L Scale																	
UMT-L [65]	0.3B	224	8	25M	-	-	-	-	-	-	47.1	40.7	37.1	49.0	74.5	41.9	39.4
SigLIP2-L/16† [134]	0.3B	384	8	n/a	64.1	65.3	62.5	56.8	86.7	49.3	44.7	41.5	31.4	53.7	74.2	35.9	31.5
PE$_{\text{core}}$L	0.3B	336	8	22M	**71.4**	**73.4**	**72.7**	**65.3**	**87.1**	**58.5**	**54.8**	**50.3**	**50.1**	**57.2**	**82.4**	**46.4**	**42.1**
Unbounded Scale																	
InternVL [17]	5.5B	224	8	n/a	-	69.1	68.9	60.6	-	-	-	44.7	40.2	-	-	-	-
InternVideo2 [141]	1.0B	224	8	102M	70.7	73.1	72.8	64.9	88.8	53.9	**59.9**	**51.9**	50.9	58.1	83.3	**60.4**	**54.8**
VideoPrism-g* [159]	1.1B	288	16	619M	-	76.4	-	-	-	-	-	39.7	**71.0**	-	-	52.7	50.3
SigLIP2-g-opt† [134]	1.1B	384	8	n/a	68.2	69.8	67.0	61.8	90.7	51.8	46.6	43.1	34.2	55.8	74.6	38.3	33.4
PE$_{\text{core}}$G *(image only)*	1.9B	448	8	n/a	70.9	73.1	72.2	64.3	89.5	55.5	47.6	44.3	35.2	54.3	73.9	41.4	36.3
PE$_{\text{core}}$G	1.9B	448	8	22M	**74.8**	**76.9**	**76.1**	**69.1**	**90.7**	**61.1**	58.7	51.2	49.9	**59.7**	**85.4**	54.7	51.2

Table 6 Zero-Shot Video Results. Video performance of PE$_{\text{core}}$ compared to recent video and image encoders. PE$_{\text{core}}$ obtains state-of-the-art in video classification and comparable performance on retrieval benchmarks while using only 22M videos. *Proprietary models. †SigLIP2 are evaluated by us with the same zero-shot prompts frame embedding averaging strategy (as in [17, 82, 103]). See Appendix B.1.2.

Model	Encoder Params	Resolution	Data	Zero-Shot Classification				Zero-Shot Retrieval			
				ObjectNet [4] *IN Overlap (113)*	ObjectNet [4] *All Classes (313)*	iNaturalist *2017* [136]	Dollar St *58* [37, 110]	TextCaps *img→txt* [118]	TextCaps Flip *img→txt* [118]	PVD Bench *text→vid*	PVD Bench *vid→txt*
SigLIP2-B/16 [134]	0.1B	224	10B	**73.6**	**59.1**	16.9	**55.9**	72.0	69.8	53.9	60.1
PE$_{\text{core}}$B	0.1B	224	5.4B	71.9	58.3	**25.9**	52.1	**72.3**	**71.9**	**59.8**	**61.1**
SigLIP2-L/16 [134]	0.3B	384	10B	84.4	73.2	26.7	57.6	78.0	76.2	61.9	**67.1**
PE$_{\text{core}}$L	0.3B	336	5.4B	**84.7**	**74.3**	**35.3**	**59.6**	**78.5**	**78.3**	**64.7**	65.2
InternVL-C [17]	5.5B	224	5B	80.6	67.2	19.4	58.2	72.3	67.8	63.4	65.1
SigLIP2-g-opt [134]	1.1B	384	10B	88.0	78.1	31.5	59.3	**78.8**	76.9	62.5	67.1
PE$_{\text{core}}$G	1.9B	448	5.4B	**88.2**	**79.0**	**41.1**	**62.3**	**78.8**	**78.7**	**77.0**	**76.6**

Table 7 Additional Zero-Shot Results. We present several additional zero-shot benchmarks from existing datasets and our own PVD (§2.3) to address evaluation gaps left by standard benchmarks.

Model	Encoder Params	Resolution	Data	Encoder Probing		
				ImageNet [24] *KNN*	ImageNet [24] *Linear*	ImageNet [24] *Attention*
DINOv2-g [95]	1.1B	224	145M	83.5	86.5	87.2†
RADIOv2.5-g [43]	1.1B	518	-	85.3	-	-
AIMv2 3B [35]	2.7B	448	7.2B	-	-	89.5
InternVL-C [17]	5.5B	224	5B	-	88.2	-
EVA 18B [126]	17.5B	224	2B	-	88.9	-
PE$_{\text{core}}$G	1.9B	448	5.4B	**86.8**	**89.5**	**89.8**

Table 8 Encoder Probing Results. We evaluate PE$_{\text{core}}$G's frozen features using the typical probing methods to compare to models without zero-shot support. †from [35].

Additional Zero-Shot Benchmarks. We further evaluate PE$_{\text{core}}$ on an additional set of zero-shot classification and retrieval benchmarks we construct in Tab. 7 to address key gaps in common benchmarks. For comparison, we also evaluate SigLIP2 [134] and InternVL-C [17] on these benchmarks.

First, we note that the version of ObjectNet [4] that is standard to benchmark robustness (e.g., in Tab. 5) is *not* the full set. ObjectNet consist of 313 classes of objects in challenging and uncommon orientations, locations, and viewpoints. However, the standard version used for benchmarking is a 113 class subset of classes that overlap with ImageNet-1k [24]. Naturally, benchmarking in this way rewards performing well on ImageNet classes over generality. To remove this bias, we construct the full ObjectNet set with all classes and compare to the reduced ObjectNet set in Tab. 7. Surprisingly, we find that while PE$_{\text{core}}$G performs +7.6% over InternVL-C and only +0.2% over SigLIP2-g-opt on the reduced ObjectNet set, it performs +11.8% over InternVL-C and +0.9% over SigLIP2-g-opt on the full set of classes, highlighting PE's generality.

Next, we include iNaturalist [136] as a *zero-shot* benchmark because of its level of specificity with 2,101 fine-grained long-tail classes. PE$_{\text{core}}$G outperforms the next best SigLIP2-g-opt model by *+9.6%*, emphasizing PE's long tail knowledge. We then evaluate PE's cultural diversity on Dollar Street [110][3], which consists of images of under-represented populations. Here too we find PE$_{\text{core}}$G to outperform existing methods, with +3.0% over SigLIP2-g-opt. Further, we test OCR performance by setting up TextCaps [118] as a retrieval dataset. Notably, PE$_{\text{core}}$ performs on par or better than SigLIP, which is known for good OCR performance. This is potentially surprising, as the horizontal flip augmentation we used during robust pretraining (§2.1) is typically thought to hurt OCR performance. However, instead it seems to have given PE$_{\text{core}}$ the ability to read backwards: we test the same TextCaps retrieval but with all images horizontally flipped. Other models suffer from this, but PE$_{\text{core}}$G's performance only drops by 0.1%. Finally, we evaluate PE$_{\text{core}}$G on the PVD benchmark (§2.3), a challenging video retrieval task on 15K diverse and human-refined videos. Here, PE$_{\text{core}}$G significantly outperforms InternVL [17] by +13.6% on text→video and +9.5% to SigLIP2 [134] on video→text.

Frozen Encoder Probing Results. To compare against models that are not capable of zero-shot classification, we additionally evaluate PE$_{\text{core}}$ using k nearest neighbors (following [95]), linear probing (following [17]), and attention probing (following [35]) on top of the ImageNet-1k [24] train set. We present these results in Tab. 8 and compare to other encoders using their reported numbers. In every case, PE$_{\text{core}}$G outperforms all existing open encoders, including those with significantly more parameters.

Summary. PE$_{\text{core}}$, a unified image-video encoder, achieves state-of-the-art performance across zero-shot classification and retrieval on both images and videos on a wide variety of benchmarks. This synergy is made possible by our robust image pretraining recipe (§2.1) and powerful video data engine (§2.2), which together enable the model to effectively leverage the strengths of both image and video data at scale.

[3]We use the version provided by [37] and re-evaluate all models to ensure a fair comparison.

3 General Features in a Contrastive Disguise

PE_{core} puts up strong results on the tasks contrastive encoders are known for, like zero-shot classification and retrieval. But while those tasks are useful, they are only a small part of the vision ecosystem. What *really matters* is whether or not the features learned with our pretraining recipe are useful to downstream tasks.

Today's common wisdom in the vision community cites that different pretraining methods result in features useful for different tasks: e.g., contrastive for classification, captioning for language modeling, and self-supervised learning for spatial tasks. To see how PE_{core} stacks up against against models with different pretraining techniques, we compare its *frozen features* to the state-of-the-art large-scale models for captioning (AIMv2-3B [35]) and self-supervised learning (DINOv2-g [95]) on a variety of downstream tasks.

Layerwise Feature Analysis. We summarize the results of our frozen feature analysis in Fig. 8 for several downstream benchmarks in 3 categories: classification, language modeling, and spatial tasks. For classification, we probe each model using a randomly initialized cross attention transformer block. For language alignment, we use the Perception Language Model (PLM) [19] frozen encoder evaluation setup, learning a projector and finetuning a decoder-only LLM (see §4), and for spatial tasks we train with several different decoders (ViTDet [70] Mask-RCNN [41] with Absolute Win [7] for detection, DPT [106] for depth, and zero-shot feature correspondance for tracking [50]). For each experiment, we sweep over the layers of the model as the optimal features are not necessarily the last [16]. In each case, we use an equivalent image size (window size for detection) of 32×32 tokens. In each plot, we normalize performance by the maximum and minimum performance across models on that task.

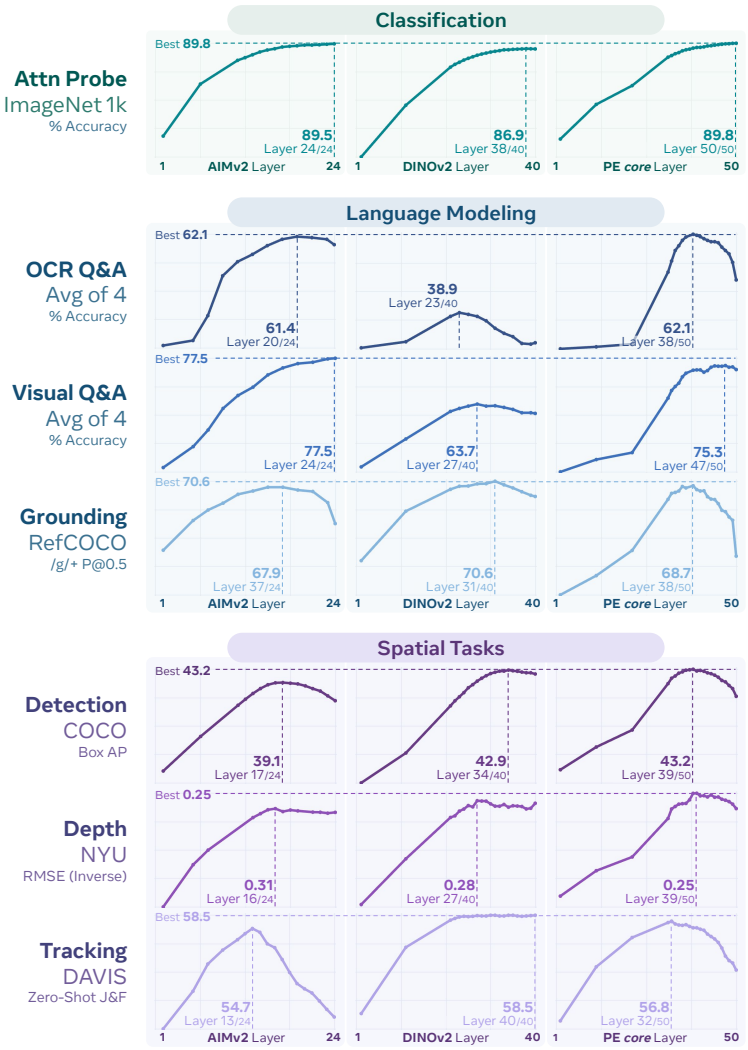


Figure 8 Layer Analysis. Evaluating intermediate layers as frozen features across tasks for different pretraining methods: captioning (AIMv2-3B [35], left), spatially self-supervised (DINOv2-g [95], middle), and our contrastive recipe (PE_{core}G, right). Vertical lines denote the best layer and horizontal lines the best performance across models. As expected, AIMv2 performs well on language but not spatial, and DINOv2 performs well on spatial but not language. But surprisingly, *intermediate layers* of PE_{core}G perform well on *both* language modeling and spatial tasks.

An Alignment Problem. This analysis reveals several insights. First, as expected, AIMv2 performs well at classification and the best at visual Q&A language tasks. Similarly, DINOv2 performs the well on spatial tasks like detection, depth, and even performs the best at grounding through an LLM. Then as already established by other works: DINOv2 lacks performance on OCR tasks [130]. This is no secret, but what is interesting is that its performance *peaks in the middle of the network* and then drops significantly by the end. And so does the performance of other models for other downstream tasks (AIMv2: tracking, grounding, detection; DINOv2: VQ&A, grounding).

PE_{core} exhibits similar behavior, but with unexpected results. Unlike the others, in earlier layers of the network PE_{core} *performs well on all tasks, often matching or exceeding the leading models*. Remarkably, PE has intermediate layers that perform near to or on par with AIMv2 for language tasks and DINOv2 for spatial tasks, despite being trained with contrastive loss. Depth estimation is particularly noteworthy, as contrastive encoders are not typically considered state-of-the-art in that area.

However, in almost all cases this strong performance *diminishes rapidly* towards the end of the network. In fact, the performance of PE_{core} in the final layer is *abysmal* for certain tasks, such as LLM-based grounding (the reason for which will become apparent in §5). This behavior is less pronounced the closer the downstream task is to the pretraining method, suggesting an *alignment problem*. Specifically, a well-tuned large-scale contrastive model can learn general embeddings in the process of fitting its objective, *but it fails to output them*. Therefore, to reveal these embeddings, the model must be subsequentially aligned to downstream tasks.

Analysis. The finding that pure CLIP models possess features which match the performance of state-of-the-art pretraining methods in their specialized domains is new. In fact, recent work [29] has shown the opposite—that CLIP models fail to scale on downstream tasks. We next investigate how our approach yields these results.

To start, we perform layerwise frozen feature analysis on COCO detection. PE_{core} was particularly "peaky" on this task in Fig. 8, with its best layer on par with DINOv2, but last layer significantly worse. We already ablated each change we made from vanilla CLIP in Fig. 2 using a ViT-L/14 model. So to retrace our steps, we run frozen feature analysis on those checkpoints. For efficiency, we perform this experiment at a lower resolution and only sample even layers. In Fig. 9, we report COCO box mAP for the the last and best layers for each cumulative ablation, along with the index of the best layer. Further, we plot the layerwise performance for each change in Fig. 10.

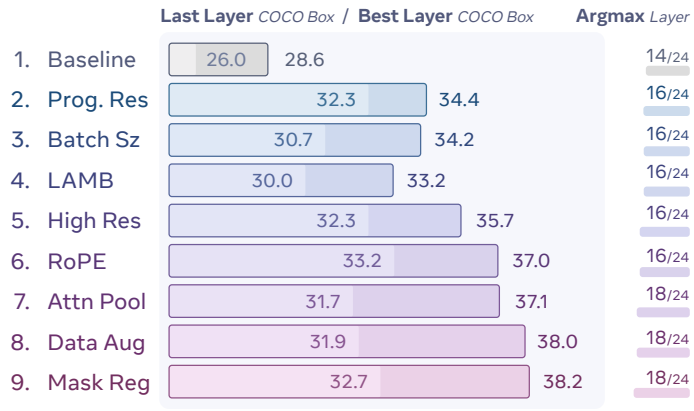


Figure 9 The Downstream Effects of Robust Pretraining. The ViT-L/14 checkpoints from Fig. 2 evaluated as frozen features on COCO [74] using Mask R-CNN [41]. We report the last layer performance, best layer performance, and the best layer's index.

Surprisingly, the simple changes we made in §2.1 to construct our pretraining recipe overall improved the best layer's performance by *almost 10 mAP* over vanilla CLIP! Some changes like high resolution (5) and RoPE (6) improving spatial features is to be expected, but unexpectedly data augmentation (8) and *especially* progressive resolution (2) help considerably. It is possible that contrastive pretraining is prone to overfit to the "global" nature of the task through "global tokens" [21]. However, as the model cannot maintain global tokens in the same place due to the resolution progressively changing, it is forced to be more robust. Also of note is that both progressive resolution (2) and attention pooling (7) move the argmax layer deeper into the network (rightmost column of Fig. 9). Attention pooling in particular alters the whole shape of the layerwise performance curve (Fig. 10), while the other changes typically only raise or lower it.

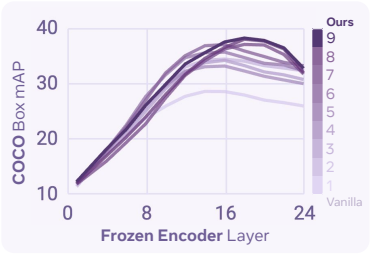


Figure 10 Layer Analysis corresponding to the results presented in Fig. 9.

Potentially more interesting is what did not improve performance: specifically, increasing the batch size (3) and using LAMB with a high learning rate (4). Both of these changes explicitly help the model fit the CLIP loss better, which after a certain point may not improve the general features. Moreover, while the best layer overall improved significantly, the last layer performance stagnated after (2). This suggests that constructing the global CLIP token requires a substantial "decoder" (in this case, 6 layers for the final L/14 model). Although the features of this decoder are beneficial for some tasks (e.g., Visual Q&A as shown in Fig. 8), they are not general. Nevertheless, this does not prevent the model from learning general features; it merely limits their expression in the output.

Scaling Behavior. Finding a simple, easily scalable vision pretraining method that produces generally useful features has been the white whale of the vision community for a while. Evidently, our robust recipe can enable contrastive pretraining to produce general features. So that begs the question, "does it scale?"

We can answer this question in the same way: by performing frozen feature layer analysis of our S/14, B/14, and L/14 scaling ablation checkpoints from Fig. 3. We report the result of that analysis in Fig. 11. We also include our final PE_{core}G model using the same setup, but note this is an estimate as our ablation and final schedules are different.

Immediately, we see a stark contrast between the scaling behavior of the vanilla CLIP recipe and ours. While the vanilla recipe quickly plateaus at L scale (300M), the best layer of our robust pretraining recipe demonstrates scaling to G scale (2B) and potentially beyond—despite being trained with a decidedly non-spatially aligned global contrastive loss. However, this is the *best* layer. The *last* layer performance still stagnates for both the vanilla recipe and ours. This may be why prior work [29] finds contrastive pretraining to not scale for downstream tasks—CLIP loss obfuscates its general features even with our recipe, placing them several layers deep.

However, this is just for a single spatial task. To see whether the trend is consistent, we repeat this scaling analysis on a wide variety of downstream language modeling tasks using the same frozen evaluation setup as Fig. 8 and report the results in Fig. 12. Surprisingly, the simple change in pretraining recipe improves scaling for most language tasks as well—including output-side grounding (RefCOCO). Note that in this benchmarking setup, the LLM never sees videos during training so the Video Q&A per-layer results are noisy. Yet, the best layer trend is still the same.

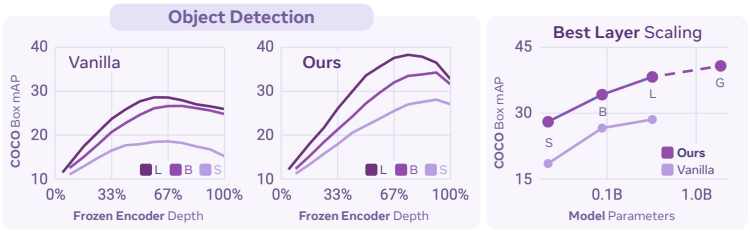


Figure 11 The Downstream Scalability of Robust Pretraining. Left: frozen feature layer analysis of the S/14, B/14, and L/14 models from Fig. 3 using the same setup as Fig. 9. Right: scaling behavior of the *best layer* for each model. Note: G is our final model and has a different schedule.

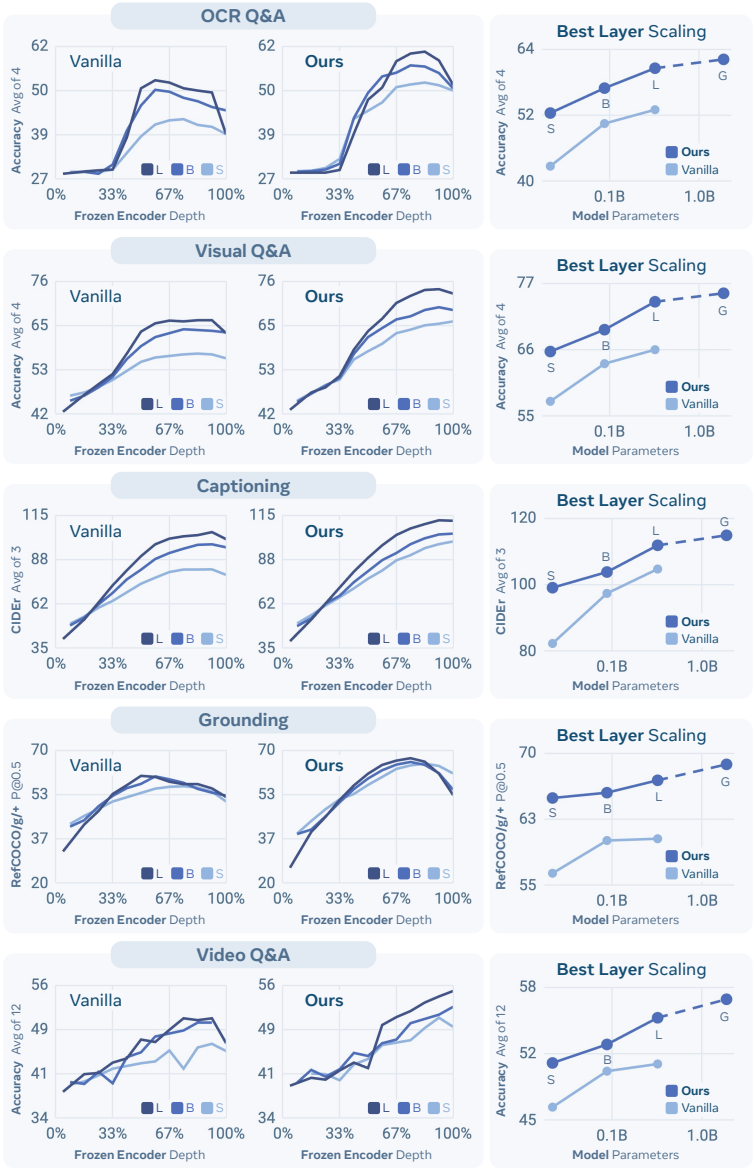


Figure 12 Further Scalability Analysis. We repeat the analysis from Fig. 11 on a wide range of downstream tasks by adapting to a language model. Each category is an average of several downstream tasks (see §4).

Clearly, contrastive pretraining with our robust recipe produces strong general features that scale. However, these features are not going to be much use stuck in the middle of the network. To remedy this, in the remaining sections we will discuss methods for *aligning* these general features to the output of the network for both language modeling and spatial tasks.

4 Perception Encoder: Language Alignment

In §3 we have seen that PE_{core} already possesses useful features for vision-language modeling. In this section, we *lift* these features through *alignment tuning* to construct a new encoder, PE_{lang}, specialized for multimodal large language models (MLLMs). Our principle is to design not only the most performant, but also the most *general* vision encoder for use in MLLM development. To this end, we want a single language-aligned encoder that performs well across language models, across input resolutions, and for a wide variety of MLLM tasks.

MLLM Evaluation Tasks. In this section, our main testbed is to adapt vision encoders to MLLMs and test on various MLLM tasks. We evaluate the downstream performance of each MLLM across five task categories: (1) *OCR, Chart, Document Q&A* on ChartQA [160], DocVQA [88], InfoVQA [89] and AI2D [55]; (2) *Visual Q&A* on TextVQA [121], OK-VQA [115], POPE [71], and VQAv2 [38]; (3) *Captioning* on Flicker [152], COCO [74], and No Cap [1]; (4) *Video Understanding* on VideoMME [36], STAR [143], TGIF-QA [51], EgoSchema [87], MVBench [66], and PerceptionTest [102]; and finally (5) *Grounding* on RefCOCO [54].

4.1 Language Alignment Method

We begin by searching for the optimal language alignment method. We design our alignment tuning based on the *midtraining* stage of Perception Language Model (PLM) [19], which is to adapt PE_{core} to a pretrained decoder-only LLM (Llama 3 [80]) connected by a vision projector. We start with "warmup" training stage with autoregressive next-token prediction loss on 1M image-text samples from pretraining, where everything but the projector is frozen. Then, we proceed to finetune all parameters on 70M data samples [19] covering natural images, documents/charts/diagrams, and videos, using the same next-token prediction loss. After completing this language alignment, we extract the vision encoder from the model and refer to it as PE_{lang}.

To arrive at the optimal training configuration presented in PLM [19], we first conduct ablation studies using a 20M subset of the data. In Tab. 9, we ablate the LLM sizes, training parameters, vision projector types, output layers to project, and encoder regularization. We evaluate across OCR Q&A, Captioning, Visual Q&A, and Video Q&A and find the best configuration.

LLM scale	LLM unfrozen	Regularization?	Projector	Layer	Avg.	OCR Q&A *Average of 4*	Captioning *Average of 3*	Visual Q&A *Average of 4*	Video Q&A *Average of 6*
LLM Setup									
1B			MLP	47	76.5	60.7	115.1	76.0	54.0
3B			MLP	47	78.1	65.9	115.7	76.6	54.1
3B	✓		MLP	47	78.4	65.8	117.6	76.3	53.7
Vision Projector									
3B			Linear	47	77.2	64.5	114.1	76.5	53.7
3B			MLP	47	78.1	65.9	115.7	76.6	54.1
PE Output Layer									
3B			MLP	50	75.9	56.6	116.7	76.5	53.7
3B			MLP	47	78.1	65.9	115.7	76.6	54.1
3B			MLP	41	76.9	65.5	112.8	75.4	53.9
PE Regularization									
3B		✓	MLP	47	79.9	69.0	117.5	77.4	55.6
3B	✓	✓	MLP	47	**80.1**	68.7	118.3	77.0	56.3

Table 9 Language Alignment. We find the best configuration to language align PE_{core}G using autoregressive language training.

LLM Setup. We explore different *scales* (1B or 3B parameters) and *freezing* weights of the LLM. We observe that going from 1B to 3B parameters increases average score by 1.6 points (76.5→78.1). Unfreezing the LLM boosts this number to 78.4.

Vision Projector. Using a *2-layer MLP* vision projector instead of a *linear layer* improves the average score from 77.2 to 78.1, while only adding few parameters (13.5M → 27M).

PE Output Layer. As shown in §3, PE_{core}G has intermediate layers that perform significantly better than the last layer when used as features for certain tasks. However, it is not clear if that same behavior applies when finetuning. We test applying the projector to layers 41, 47, and 50 (the last layer), and find that layer 47 works best. Incidentally, this is also the optimal layer for frozen VQ&A in Fig. 8.

PE Regularization. We apply LayerScale [131] and DropPath [48] to the vision encoder during the alignment, for stabilizing training. This improves the 78.1 average score to 79.9 (+1.8 points). Unfreezing the LLM boosts this number further to 80.1. We choose this configuration (last row) as our final alignment setup.

To construct PE_{lang}, we scale this recipe up the 70M samples mentioned above (more details in [19]). In summary, we use a pretrained Llama3.2 3B, unfrozen, with a 2-layer MLP as a vision projector on top of layer PE_{core}G layer 47 (with the last 3 discarded) and regularize the encoder with LayerScale and DropPath. Compared to the 20M sample ablation setting in Tab. 9, the final PE_{lang} trained on 70M total samples gives another +2.1 points to 82.2 on the average across OCR Q&A, Captioning, Visual Q&A, and Video Q&A.

Effects. The goal of alignment tuning is to *lift* the strong features found in intermediate layers of PE_{core} described in §3 to the end of the network. To see if we actually accomplished that, we perform the same layerwise

analysis as in Fig. 8 on our final PE_{lang}G model and compare it to the original PE_{core}G checkpoint it was initialized from. We present the results of this analysis in Fig. 13, and immediately we see that language alignment was a success: across all categories, the performing layer for the aligned model was the last, no matter the performance of the original checkpoint. Notably, our PE_{lang} training mix did *not* contain grounding data, which means that this significantly lifted grounding performance is entirely due to the strong intermediate grounding features in PE_{core} now being aligned to the end of the network. Moreover, specific domains such as OCR Q&A that *were* represented in the training mix see a significant boost to performance compared to even the best layer of PE_{core}, which was already strong. Thus, with an order of magnitude fewer samples compared to pretraining, we were able to *language align* PE_{core}G to create a single, strong encoder for all visual language modeling tasks. Following this success, we align PE_{core}L in a similar manner to construct PE_{lang}L (see [19]).

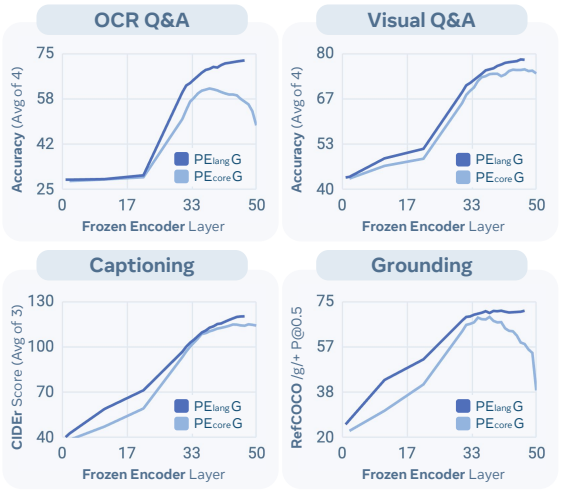


Figure 13 Language Alignment. We analyze how language alignment changes the the internal features of PE. Similar to our PE_{core} analysis in Fig. 12, we extract PE_{lang} and adapt each layer to a new LLM.

4.2 Comparisons with Existing Vision Encoders

We compare PE_{core} and PE_{lang} with other vision encoders that are popular choices in MLLM literature: MetaCLIP [147], SigLIP2 [134], CLIP [103], AIMv2 [35], DINOv2 [95], and InternViT2.5 [16]. Overall, these encoders span several different pretraining losses (e.g., contrastive, captioning, self-supervised, and mixed supervision), encoder sizes (from 300M to 6B parameters), and resolutions (from 224 to 512). *For all vision encoders, we find the best intermediate layers to train MLLM for fair comparison* (more in Appendix B.2).

MLLM Benchmarking Setup. We connect each vision encoder, including PE_{lang}, to a language decoder with a fresh 2-layer MLP projector. Similar to the alignment stage, we first train only the projector on a subset of 1M image-text pairs from pretraining. Then, we train both the projector and LLM on 2.6M visual Q&A pairs,

			OCR / Chart / Doc. Q&A						*Visual Q&A*						*Captioning*						*Video*						
Model	Encoder Params	Resolution / Patch Size	**Avg. OCR QA**	ChartQA *Acc.* [160]	DocVQA *Acc.* [88]	Info. QA *Acc.* [89]	AI2D *Acc.* [55]	**Avg. VQA**	TextVQA *Acc.* [121]	OK-VQA *Acc.* [115]	POPE *Acc.* [71]	VQAv2 *Acc.* [38]	**Avg. Cap.**	Flicker *CIDEr* [152]	COCO *CIDEr* [74]	No Cap *CIDEr* [1]	**Avg. Ground.** *RefCOCO/g/+* [54]	**Avg. Video**	VideoMME *Acc.* [36]	STAR *Acc.* [143]	TGIF-QA *Acc.* [51]	EgoSchema *Acc.* [87]	MVBench *Acc.* [66]	PerceptionTest *Acc.* [102]
256 Tokens per Image																											
MetaCLIP-L [147]	0.3B	224/14	44.9	47.9	33.0	28.7	70.2	68.4	47.6	62.5	86.9	76.5	110.5	87.5	130.0	114.1	60.6	53.9	46.1	51.0	66.4	58.6	49.4	51.9
MetaCLIP-G [147]	1.8B	224/14	44.8	47.6	33.1	27.9	70.6	68.8	48.2	63.5	86.5	76.9	111.1	86.5	132.1	114.8	60.5	53.1	45.0	50.7	66.4	56.0	48.7	51.9
PE_{lang} G†	1.7B*	224/14	53.7	61.3	47.1	32.2	74.1	71.8	55.1	65.3	86.8	79.8	116.4	91.0	136.9	121.2	65.7	55.5	47.3	55.7	68.9	59.6	48.6	52.9
576 Tokens per Image																											
CLIP [103]	0.3B	336/14	53.5	61.7	49.5	32.8	70.1	72.7	60.7	63.9	87.3	78.9	113.3	92.0	132.9	115.0	65.0	54.2	46.3	52.1	68.6	57.4	48.5	52.3
AIMv2-L [35]	0.3B	336/14	53.3	61.6	48.0	32.1	71.4	73.7	62.7	64.3	87.7	80.1	115.2	90.9	135.6	119.2	63.3	52.5	44.3	50.9	67.5	54.4	44.9	53.2
AIMv2 L Dist. [35]	0.3B	336/14	53.7	61.1	49.4	31.5	72.7	74.1	62.8	64.8	88.3	80.3	117.8	94.7	137.5	121.2	62.6	53.8	44.3	52.4	65.0	57.4	50.0	53.6
SigLIP2-so [134]	0.4B	384/16	58.9	69.0	58.3	35.2	73.1	76.8	69.8	**67.2**	88.7	81.6	116.5	92.1	137.7	119.8	67.4	54.5	45.5	53.1	67.2	57.6	49.3	54.5
SigLIP2-g-opt [134]	1.1B	384/16	56.2	63.1	55.3	34.0	72.4	77.0	70.3	66.7	89.6	81.6	117.7	94.9	137.8	120.3	66.5	53.9	46.2	53.9	66.6	53.8	48.5	54.7
PE_{lang} G†	1.7B*	336/14	66.9	76.8	73.6	41.1	76.1	76.2	68.5	66.0	89.1	81.3	119.7	96.1	139.6	123.4	68.9	58.1	48.7	58.9	70.5	61.8	52.7	55.9
1024 Tokens per Image																											
InternViT 2.5 L [16]	0.3B	448/14	60.6	74.1	59.2	35.9	73.1	74.2	65.4	64.4	87.6	79.6	112.3	88.4	133.7	114.9	66.9	50.6	45.2	44.8	62.7	54.2	46.0	50.5
SigLIP2-so [134]	0.4B	512/16	63.3	72.1	69.3	39.0	72.7	77.9	74.8	66.0	89.0	**81.8**	117.4	93.5	138.3	120.2	69.6	55.8	46.2	55.4	67.0	**62.0**	50.0	54.5
PE_{core} L	0.3B	448/14	59.4	68.7	62.5	36.6	69.7	74.7	67.7	64.3	88.3	78.7	112.7	89.6	133.4	114.9	59.7	50.9	41.7	51.2	61.6	52.6	47.4	50.6
PE_{lang} L	0.3B	448/14	71.1	81.0	81.9	46.4	75.0	77.1	73.0	65.5	89.3	80.8	117.3	94.3	137.3	120.1	70.5	56.5	47.0	57.2	68.0	59.8	52.3	54.7
DINOv2-g [95]	1.1B	448/14	30.0	19.6	14.7	24.2	61.5	61.0	19.3	60.4	88.6	75.8	109.4	86.5	131.6	110.1	64.9	49.5	39.7	52.1	60.1	46.8	47.4	50.8
AIMv2 3B [35]	2.7B	448/14	48.9	40.5	53.9	33.9	67.2	73.0	64.1	64.0	85.2	78.9	115.7	93.8	135.2	118.1	36.1	54.6	45.1	54.5	66.7	55.4	51.7	54.3
InternViT2.5-6B [16]	5.5B	448/14	59.9	72.3	59.4	35.2	72.5	75.5	68.9	64.9	88.2	80.2	115.0	92.2	136.3	116.3	68.0	49.6	44.5	47.0	62.6	45.8	48.9	48.5
PE_{core} G	1.9B	448/14	60.8	69.9	65.4	36.7	71.1	73.3	65.9	60.7	88.4	78.0	112.5	91.6	133.6	112.4	66.6	52.0	42.3	53.1	62.9	51.4	48.8	53.6
PE_{lang} G	1.7B*	448/14	**72.4**	**80.5**	**84.4**	**48.3**	**76.4**	**78.1**	**75.2**	65.4	**90.1**	**81.8**	**120.1**	**96.6**	**140.0**	**123.6**	**71.3**	**58.0**	**48.0**	**60.1**	**69.4**	**62.0**	**52.4**	**56.0**

Table 10 MLLM Results with Llama 3.1 8B. We compare various vision encoders at their native resolution using Llama 3.1-instruct 8B [80] as the language model. The table compares models of similar class in number of vision tokens and parameters. PE_{lang} shows strong performance across all benchmarks, including against models 3× its size. *PE_{lang} has 1.7B parameters since we discard the last 3 layers during language alignment. †Interpolated without extra training.

image captions, and image grounding samples (see Appendix B.2 for details). We benchmark at the native resolution of each encoder (with higher resolution tiling results in Appendix C.4). Finally, we ablate over two language decoders, Llama 3.1 8B [80] and QwenLM 2.5 7B [150], to measure generalization across LLMs.

Results. Tab. 10 shows benchmarks results for native resolution input across existing encoders, PE_{core} and PE_{lang}. Notably, AIMv2 [35], InternViT2.5 [16], SigLIP2 [134] and PE_{lang} are trained jointly with a language decoder using next token prediction objective, and thus they perform better overall compared to the base contrastive and self-supervised models across all the metrics. However, PE_{lang} uses a fraction of the training FLOPs for language alignment tuning, while significantly outperforming all vision encoders by large margin (an average of +3.5 points for G and +2.0 points for L). Similarly, when tiling with 4 tiles and 1 thumbnail (see Appendix Tab. 30), both PE_{lang}L and PE_{lang}G outperform all existing vision encoders, including InternViT2.5 [16], which was specifically pretrained in a tiling setting and with grounding data. Appendix C.4, shows a breakdown of the RefCOCO results, as well as results for tiling with higher resolution.

Transferability. As PE_{lang} is aligned with Llama 3.2-instruct 3B, we conduct a separate set of experiments to check if our model performs well with a different base LLM. In Tab. 11 we repeat the native resolution comparison with QwenLM 2.5 7B [150]. Interestingly, PE_{lang} not only outperforms all vision encoders in this setting, but it also outperforms InternViT2.5 [16], which is specifically aligned to QwenLM 2 [149] throughout midtraining. In fact, PE_{lang}G with QwenLM even improves its performance with Llama in some cases like with OCR Q&A and video benchmarks, emphasizing the generality of our language alignment.

			OCR / Chart / Doc. Q&A					*Visual Q&A*						*Captioning*					*Video*						
Model	Encoder Params	Resolution *Patch Size*	**Avg. OCR QA**	ChartQA *Acc. [160]*	DocVQA *Acc. [88]*	Info. QA *Acc. [89]*	AI2D *Acc. [55]*	**Avg. VQA**	TextVQA *Acc. [121]*	OK-VQA *Acc. [115]*	POPE *Acc. [71]*	VQAv2 *Acc. [38]*	**Avg. Cap.**	Flicker *CIDEr [152]*	COCO *CIDEr [74]*	No Cap *CIDEr [1]*	**Avg. Ground.** *RefCOCO/g/+ [54]*	**Avg. Video**	VideoMME *Acc. [36]*	STAR *Acc. [143]*	TGIF-QA *Acc. [51]*	EgoSchema *Acc. [87]*	MVBench *Acc. [66]*	PerceptionTest *Acc. [102]*
576 Tokens per Image																									
SigLIP2-so [134]	0.4B	384/16	60.5	72.0	59.1	36.7	74.3	76.2	69.0	65.4	89.2	81.1	116.3	91.6	137.3	120.0	70.0	57.0	51.3	55.8	66.0	61.0	51.9	55.7
SigLIP2-g-opt [134]	1.1B	384/16	60.8	71.0	60.4	36.7	75.2	76.8	70.3	65.6	89.5	81.8	118.8	96.4	139.0	121.1	69.9	58.3	52.0	57.6	68.1	62.0	52.8	57.4
PE_{lang}G†	1.7B*	336/14	66.8	77.5	72.4	41.1	76.4	76.0	67.9	65.4	89.1	81.5	118.8	94.6	139.5	122.3	70.1	60.2	54.6	61.7	69.8	63.6	54.3	57.2
1024 Tokens per Image																									
InternViT2.5 [16]	0.3B	448/14	60.3	75.4	61.1	36.2	68.4	74.2	65.6	63.7	87.8	79.5	112.1	88.5	133.5	114.1	68.1	55.8	50.3	54.7	66.6	59.0	50.6	53.8
SigLIP2-so [134]	0.4B	512/16	66.3	77.2	71.9	42.4	73.9	**77.9**	74.2	65.6	89.9	81.8	117.1	93.0	138.0	120.3	70.5	55.9	50.3	57.3	67.2	62.6	50.3	47.4
PE_{core}L	0.3B	448/14	63.5	73.9	67.4	40.5	72.2	75.7	69.2	64.0	89.4	80.2	113.3	88.7	135.2	115.9	66.5	57.3	49.6	57.8	67.7	60.8	52.3	55.5
PE_{lang}L	0.3B	448/14	70.2	80.6	80.7	46.0	73.5	76.8	72.8	64.1	89.4	81.0	116.4	93.4	137.6	118.1	70.4	58.3	51.6	59.8	67.4	62.2	53.4	55.4
DINOv2 [95]	1.1B	448/14	31.3	21.7	14.7	24.6	64.3	61.0	18.9	59.5	88.9	76.9	110.1	87.3	132.1	110.8	69.3	54.3	46.9	56.5	63.4	56.8	49.7	52.2
AIMv2 3B [35]	2.7B	448/14	66.0	76.7	70.5	41.4	75.2	**77.9**	74.2	**66.2**	89.4	**81.9**	**119.2**	**96.4**	139.2	122.0	67.6	56.3	45.9	58.0	67.8	60.8	51.4	53.9
InternViT2.5 [16]	5.5B	448/14	64.2	78.2	65.3	39.6	73.6	76.4	70.1	64.5	89.3	81.7	117.6	95.9	138.4	118.6	**72.8**	56.1	50.3	59.1	67.3	56.6	51.1	52.2
PE_{core}G	1.9B	448/14	64.8	75.9	68.8	41.6	72.9	75.2	67.9	62.4	89.7	80.7	113.1	91.7	135.2	112.3	70.5	57.0	48.7	58.3	66.9	60.8	52.9	54.5
PE_{lang}G	1.7B*	448/14	**72.9**	**81.6**	**83.7**	**49.5**	**76.7**	**77.9**	**74.9**	64.5	**90.3**	**81.9**	118.9	94.6	**139.8**	**122.3**	72.1	**60.4**	**54.1**	**62.5**	**68.3**	**66.6**	**54.2**	**56.8**

Table 11 MLLM Results with QwenLM 2.5 7B. Same setting as Tab. 10, but with QwenLM2.5 7B [150] as the language model. Although PE_{lang} is aligned to Llama3.2 3B, the language alignment transfers well to a different language model.

System-Level MLLM Comparison. In Tab. 12, we conduct a system-level comparison to the state-of-the-art open-access MLLMs: LLaVA-OneVision 7B [64], Gemma3 12B [128], Molmo-D 7B [23], Qwen2 VL 7B [139], InternVL 2.5 8B [16] and the very recent InternVL 3 8B [162]. Each baseline uses a contrastively pretrained ViT (SigLIP-so400M [155], CLIP-L [103], DFN-H [31], and InternViT 2.5 300M [16]). For our PLM-8B we use PE_{lang}G as the vision encoder with 36 tiles for images and 32 frames for video and Llama 3.1-instruct 8B as the language decoder (more details in [19]). We show numbers from their respective works or evaluate them ourselves if they are not reported (except for Gemma and InternVL 3). PLM-8B outperforms all other models tested, emphasizing that PE_{lang}G can be used to drive strong results across a wide range of tasks.

		OCR / Chart / Doc. Q&A					*Visual Q&A*						*Captioning*					*Video*						
Model	Encoder	**Avg. OCR QA**	ChartQA *Acc. [160]*	DocVQA *Acc. (test) [88]*	Info. QA *Acc. (test) [89]*	AI2D *w/o mask [55]*	**Avg. VQA**	TextVQA *Acc. [121]*	OK-VQA *Acc. [115]*	POPE *Acc. [71]*	VQAv2 *Acc. (val) [38]*	**Avg. Cap.**	Flicker *CIDEr [152]*	COCO *CIDEr [74]*	No Cap *CIDEr [1]*	**Avg. Video**	VideoMME *Acc. [36]*	STAR *Acc. [143]*	TGIF-QA *Acc. [51]*	EgoSchema *(test) Acc. [87]*	MVBench *Acc. [66]*	PerceptionTest *Acc. (test) [102]*
LLaVA-OV 7B [64]	SigLIP-so400M	81.4	80.0	86.7	68.8	90.1	79.9	77.3	**69.6**	89.2	83.5	79.5	55.7	70.7	112.1	63.8	57.7	66.0	77.2	65.2	57.1	58.1
Gemma3 12B [128]	SigLIP-so400M	-	75.7	87.1	64.9	-	-	67.7	-	-	71.6	-	-	-	-	-	-	-	-	-	-	54.9
Qwen2 VL 7B [139]	DFN-H	86.6	83.6	94.5	76.5	91.7	80.9	83.6	67.9	88.3	83.8	93.7	79.9	102.5	98.7	67.7	62.9	67.3	81.8	65.4	61.6	66.9
InternVL 2.5 8B [16]	InternViT 2.5-300M	87.0	84.6	93.0	77.6	**92.8**	79.9	79.3	69.2	90.6	80.6	113.0	96.5	125.8	116.7	72.9	60.6	77.6	91.3	66.2	72.6	68.9
InternVL 3 8B [162]	InternViT 2.5-300M	87.2	**86.6**	92.7	76.8	92.6	-	80.2	-	**91.1**	-	-	-	-	-	-	**66.3**	-	-	-	75.4	-
PLM-8B	**PE_{lang}G**	**88.4**	85.5	**94.6**	**80.9**	92.7	**82.9**	**86.5**	**69.6**	89.9	**85.6**	**127.4**	**105.6**	**146.7**	**129.9**	**77.9**	58.3	**84.9**	**95.5**	**68.8**	**77.1**	**82.7**

Table 12 MLLM System-Level Comparison. We show a system-level comparison between PLM-8B based on PE_{lang}G and popular open-access models of similar LLM scale using existing encoders. We report test set results where specified.

5 Perception Encoder: Spatial Alignment

While language alignment with a pretrained LLM decoder is well-established, the best way to spatially align a model is not obvious. As shown in §3, PE_{core} already has features that perform well for spatial tasks. However, the layer that performs the best for higher level spatial tasks like detection or depth estimation (layer ∼40) is vastly different than the layer that performs the best for a pure spatial task like tracking (layer ∼30). While we were able to ignore this disparity during language alignment by aligning to an LLM decoder that could do all tasks, classical spatial tasks have decoders that come in all shapes and sizes. It would be impractical to simply align the model using all downstream decoders mirroring language alignment. Thus, we must first answer the question, what is happening in the features at those layers to make them useful for spatial tasks?

5.1 Core Feature Analysis

We begin by analyzing the spatial properties of the features for PE_{core}G in the range of layers where it performed optimally for zero-shot tracking in §3. In Fig. 14, we plot (1) the pairwise feature cosine similarity between the pink token and all others, (2) the head average attention map for that token, and (3) the full attention matrix ($HW \times HW$).

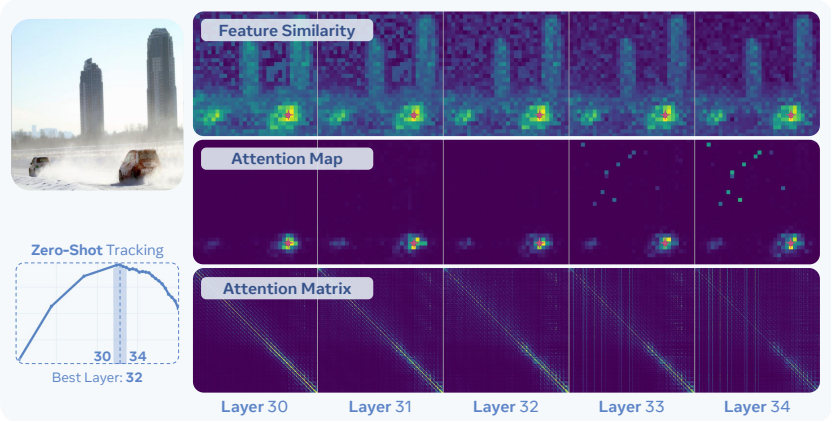


Figure 14 PE_{core}G Feature Analysis. To understand the dichotomy between optimal PE_{core} features for spatial tasks observed in Fig. 8, we analyze the spatial properties of the features between layers 30 and 34.

An 18 Layer Decoder. Remarkably, the cause for the tracking performance peak at layer 32 is abundantly clear from observing the visualizations. Up until layer 32, the attention maps remain local. However, that changes abruptly at layer 33, at which point several tokens in the background of the image become "global" tokens. As shown by the vertical lines in the full attention matrix, starting from layer 33 every token attends to them. Thus, every layer 33 and up become part of a *decoder* for global information.

This is not a new phenomenon. Recent work [21] shows this happening in all modern vision transformers above L scale. But notably these "global tokens" are not necessarily harmful. Given the optimal layer for most tasks in Fig. 8 lies within the global token region, the information they aggregate is useful downstream. However, tracking in §3 is zero-shot and relies purely on spatial correspondences, meaning it cannot make use of the global tokens. This explains why tracking peaks right before their introduction, while tasks that rely on semantic understanding or have larger decoders that can benefit from them do well with the later layers.

5.2 Spatial Alignment Method

Given the analysis in §5.1, we have two objectives in creating a spatial alignment method: (1) we must preserve the optimal *semantic information* of the model (including the global tokens) that peaks around layer 40, and (2) we must do so while emphasizing *local alignment* in service of spatial tasks with shallow decoders. The first can be easily achieved by aligning with the model's own features (e.g., with MaskFeat [142]), but the second is more challenging. To accomplish this, we employ the Segment Anything Model (SAM) 2.1 [108] in a novel way to enforce spatial correspondence information in PE.

Retaining Semantics. To retain the strong semantic features from PE_{core}, we finetune the model with itself as a teacher. Specifically, we train the model to minimize the cosine similarity between its *last layer* and the frozen layer 41 features of its initialization (a layer around the peak for many tasks in Fig. 8). On its own this would be a tautology, so we apply heavy regularization to the student: DropPath [48] and LayerScale [131] similar to language alignment, as well as performing MaskFeat [142] with 75% masking. We keep the teacher

fixed in contrast to other state-of-the-art spatial models, which all employ an EMA teacher [95, 134]. This could potentially help, but we opt for simplicity.

Encouraging Locality. While we could "retain" locality by self-distilling from layer 32 features, that may be less effective as we are already distilling another layer of the model. Instead, we turn to a model that is explicitly tuned for locality: SAM [56, 108]. Notably, several works [107, 113, 116] have shown SAM to *not* be an effective teacher when distilling from multiple sources (though recently [43] has shown it can help with some tricks). However, upon observation of the raw features of SAM 2.1-L (Fig. 15), the main problem may be the same one we are currently trying to solve: *SAM has global tokens as well*! In this case, they appear as dark spots in a grid-like arrangement across all examples in Fig. 15 raw features.

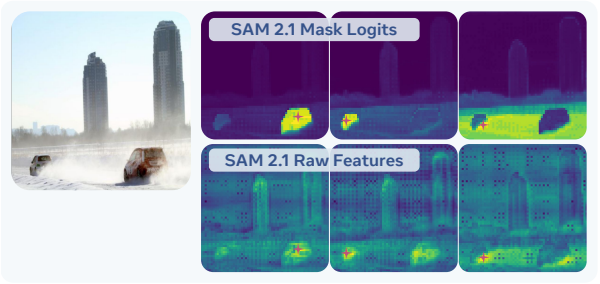


Figure 15 SAM 2.1 Feature Similarity. The cosine similarity between the pink marked token and all others for SAM 2.1-L [108] features *vs.* our proposed mask logit features.

Using the features of a model that itself has global tokens to mitigate the effect of global tokens tokens is dubious at best. But, we don't have to use SAM's *features* to learn locality. At its core, SAM is a model that transforms points into spatially contiguous masks of select object. If what we want is smooth, locally consistent features, we can use the *mask predictions* themselves. Specifically, we query SAM 2.1-L with 1024 points arranged in a 32×32 grid. For each point, SAM returns a $H \times W$ mask logit the size of the image, which it normally would threshold and NMS. However, we instead concatenate those logits into a $H \times W \times 1024$ tensor and use *that* as the feature map for alignment. This explicitly produces locally well-aligned features compared to the underlying feature space and has no spatial artifacts caused by global tokens, as shown in Fig. 15.

Then to align, we distill the spatial correspondences between tokens by computing their pairwise cosine similarity for both the student and the teacher (creating a $HW \times HW$ matrix for each) and aligning them with MSE loss. Unlike SAM's underlying feature space (which [43] shows may be brittle to interpolation), the mask logit features are robust to interpolation, so we simply interpolate them down and train at the PE_{core} model's original 448px resolution. Finally, like for self-distillation we add the same masking and regularization. For both teachers, we apply loss to all tokens and add no extra parameters other than LayerScale.

Effects. Again, the goal of alignment is to *lift* the strong features already learned by the core model as shown in §3. Thus, like we did for language alignment in §4.1, we perform layerwise frozen feature analysis on spatial tasks in Fig. 16. This time, we evaluate the original $\text{PE}_{\text{core}}\text{G}$ checkpoint as well $\text{PE}_{\text{core}}\text{G}$ aligned to its own layer 41, to SAM 2.1 mask logits, and finally both. We denote aligning to both as $\text{PE}_{\text{spatial}}\text{G}$.

Aligning purely based on the original model's layer 41 features performs well on detection, depth, and semantic segmentation, but falls short for zero-shot tracking, where precise locality is necessary to define boundaries between objects. In contrast, aligning to SAM 2.1 mask logits lowers last layer performance on every task except for tracking, where it significantly improves performance. Understandably, this is because the mask logits have little semantics (see Fig. 17). Thus, the optimal approach is to combine both teachers. As a result, $\text{PE}_{\text{spatial}}\text{G}$ not only lifts the features for all tasks to the end of the network, but it also improves over self-alignment alone. Notably, $\text{PE}_{\text{spatial}}\text{G}$'s tracking performance is lower than the SAM-aligned model, but it is still ahead of other methods while being a generally good model, see §5.3.

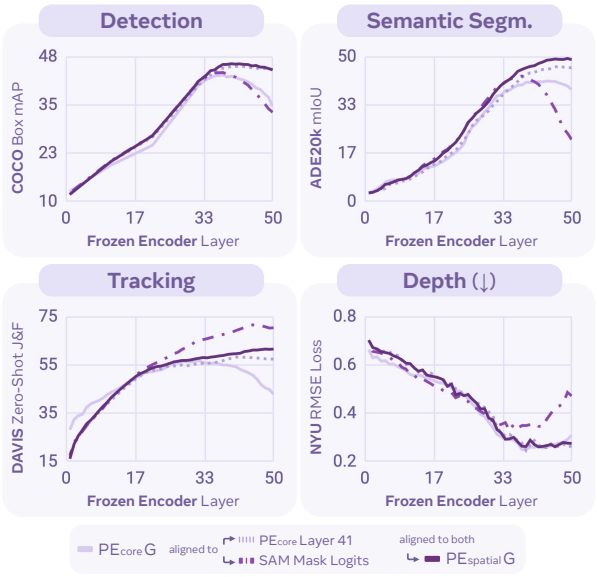


Figure 16 Spatial Alignment. We analyze how our two spatial alignment methods individually change the internal features of $\text{PE}_{\text{core}}\text{G}$. Then we combine both alignment methods to create $\text{PE}_{\text{spatial}}\text{G}$ (see Appendix B.3.1).

Last Layer Feature Visualization. In Fig. 17, we visualize the last layer features for the PE_{core}G and the 3 aligned models, with similar colors denoting similar features. In the first column, we see why the last layer performance of PE_{core} is so poor: while the last layer features contain information about the salient objects, they seem to have lost spatial coherence. Aligning to the model's own layer 41 features fixes this, but its spatial quality is lacking. In contrast, the model aligned to SAM 2.1 mask logits has locally clear features, but without semantics (similar objects have dissimilar features, see row 1 cats and row 2 cows). $\text{PE}_{\text{spatial}}$, using both teachers at once, retains the semantics of PE_{core} while producing high quality spatial features.

Figure 17 Last Layer Visualization for the models in Fig. 16 using 3 dimensional PCA to map features to LCh color space (see Appendix B.3.2). More examples in Appendix C.5.

			Tracking			Segmentation			Depth		
			DAVIS (↑) [101]			ADE20k (↑) [161]			NYU (↓) [119]		
Encoder	Params	Resolution	*Best*	*Last*	*Idx*	*Best*	*Last*	*Idx*	*Best*	*Last*	*Idx*
OAI CLIP-L [103]	0.3B	224/14	39.4	37.1	17/24	39.4	38.3	19/24	.366	.397	19/24
AIMv2-3B [35]	2.7B	448/14	54.7	29.3	13/24	41.6	31.9	20/24	.311	.326	16/24
SigLIP-so [155]	0.4B	384/14	48.7	36.3	16/27	40.1	38.3	22/27	.339	.369	21/27
SigLIP2-so [134]	0.4B	512/16	51.4	45.3	15/27	44.0	42.9	24/27	.306	.329	25/27
SigLIP2-g-opt [134]	1.1B	384/16	43.5	38.8	32/40	42.1	41.3	34/40	.302	.324	34/40
DINOv2-L [95]	0.3B	448/14	58.7	58.2	23/24	47.3	47.3	24/24	.297	.308	23/24
DINOv2-g [95]	1.1B	448/14	58.5	58.5	40/40	48.7	48.4	37/40	.279	.290	27/40
PE_{core}G	1.9B	448/14	56.8	42.8	32/50	41.5	38.6	44/50	**.249**	.309	39/50
$\text{PE}_{\text{spatial}}$G	1.9B	448/14	**61.5**	**61.5**	50/50	**49.3**	**48.9**	49/50	.262	**.275**	46/50

Table 13 Frozen Feature Dense Prediction including zero-shot tracking, semantic segmentation and depth estimation. We report best and last layer performance, along with which layer was best for each model. See Appendix B.3.3 for experimental settings.

		Pretrain	LVIS [39]		COCO [74]	
Encoder	Params	Resolution	AP_{box}	AP_{mask}	AP_{box}	AP_{mask}
OAI CLIP-L [103]	0.3B	224/14	45.0	41.9	54.0	47.5
MetaCLIP-G [147]	1.8B	224/14	45.1	41.9	53.2	46.7
SigLIP-so [155]	0.4B	224/14	45.0	41.9	54.4	47.6
MAE-L [42]	0.3B	224/14	46.1	43.9	55.6	49.3
EVA02-L [33]	0.3B	224/14	49.3	45.2	54.9	48.2
SigLIP2-so [134]	0.4B	512/16	49.3	45.6	56.0	49.4
SigLIP2-g-opt [134]	1.1B	384/16	52.9	48.5	57.1	50.2
DINOv2-L [95]	0.3B	518/14	46.7	43.5	55.7	49.0
DINOv2-g [95]	1.1B	518/14	51.5	47.3	57.2	50.0
PE_{core}G	1.9B	448/14	51.9	47.9	57.0	49.8
$\text{PE}_{\text{spatial}}$G	1.9B	448/14	**54.2**	**49.3**	**57.8**	**50.3**

Table 14 End-to-End Finetuning Detection and Segmentation using Mask R-CNN [41] and VitDet [70] in a controlled setting. Details in Appendix B.3.4.

5.3 Comparisons with Existing Vision Encoders

Frozen Feature Dense Prediction. In Tab. 13, we compare different vision encoder's frozen features on three dense prediction tasks: DAVIS tracking [101] (J&F) following the training-free setting from [50, 104], ADE20k semantic segmentation [161] (mIoU) linear probing, and NYU depth estimation [119] (RMSE) with a DPT head [106]. For each model, we report both its best layer and last layer performance. Across the board, $\text{PE}_{\text{spatial}}$ performs outperforms other state-of-the-art spatial models, with its best features being much better aligned to the last layer than the PE_{core} it started from. Notably, SigLIP2, which during pretraining combines spatial, captioning, and contrastive losses [134] is *not* aligned well to the last layer in comparison.

End-to-End Finetuning Detection and Segmentation. In Tab. 14, we compare PE_{core} and $\text{PE}_{\text{spatial}}$ with other popular vision encoders in the standard *full-finetuning* ViTDet [70] Mask-RCNN [41] setting using COCO [74] and LVIS [39] as benchmarks. In this controlled experiment, $\text{PE}_{\text{spatial}}$ is state-of-the-art among various vision backbones. This is significant, as contrastive encoders (especially large ones like MetaCLIP-G [147]) usually perform very poorly on detection, with smaller models often performing better. Typically, encoders only scale for detection if using spatial pretraining or a significant amount of detection data [95] is used to align them directly to downstream tasks. In contrast, $\text{PE}_{\text{spatial}}$ *uses no detection data for alignment*, making it general.

System-Level Detection. In Tab. 15, we provide a system-level end-to-end finetuning comparison *vs.* the absolute state-of-the-art in COCO detection. With only Object365 [117] as extra detection data, $\text{PE}_{\text{spatial}}$ can match the performance of more complex models tuned for detection, while only using a simple DETR-style decoder [11, 96]. $\text{PE}_{\text{spatial}}$ marks the first general, contrastively pretrained model to accomplish this.

Encoder	Params	Detector	COCO AP_{box}
SwinV2-G [78]	3.0B	HTC++ [13]	62.5
Swin-L [77]	284M	DINO [156]	63.2
MAE-H [42]	632M	Cascade [10]	61.3
EVA02-L [33]	304M	Cascade [10]	64.1
InternImage-G [140]	3.0B	DINO [156]	65.3
$\text{PE}_{\text{spatial}}$G	1.9B	DETA [96]	**65.5**

Table 15 System-Level Comparison on Detection. Comparing to the leading results on COCO [74] val2017. See Appendix B.3.5 for training recipe.

6 Related Work

Learning vision-semantic representations has long been the leading approach for developing foundational models in perception. By aligning visual and textual representations, these models excel not only in vision tasks such as zero-shot image classification and image-text retrieval [49, 103, 114], open-vocabulary detection [61, 91, 92] and segmentation [20, 26], but also serve as the basis for multi-modal large language models (MLLMs) [3, 5, 76, 90, 98, 130].

Contrastive Language-Image Pretraining. The early works of Virtex [25], ICMLM [112], and ConVIRT [158] developed the techniques for learning through contrastive objectives between vision and language modalities. Subsequently, vision encoders such as CLIP [49, 103] and ALIGN [52] scaled these techniques to much larger datasets and model sizes, popularizing vision-language contrastive learning. A series of open-weight contrastive models have been developed to enhance the performance and robustness of CLIP [31, 69, 114, 125, 147, 155]. For instance, SigLIP [155] replaces the traditional softmax with a sigmoid function in contrastive learning, while FLIP [72] employs masking techniques to expedite the training process. We are among this effort and build a state-of-the-art open Perception Encoder (PE) (§2.1). Other objectives that have proven useful for building visual encoders include captioning loss, which learns to predict image descriptions using a language model decoder and transfers well to downstream multi-modal language modeling tasks [35, 133]. Many works are now attempting to combine two or more objectives to address different downstream tasks through pretraining with multiple objectives [35, 153] or training sequentially [17, 64].

Efficient Training. Various axes of efficient training of clip models have been explored. BASIC [99] and LAION [114] explored scaling the batch size up to 160K, and shows the benefits of large batch sizes during training. EVA-CLIP [126] uses LAMB optimizer [151] for large batch training of clip models. Rotary positional embedding (RoPE) [123] has been successfully adopted in large language models. In vision transformers [2, 46] adopted 2D rotatory positional embeddings. For data engine, a series of works focus on large-scale sourcing and filtering through efficient data curation [31, 37, 114, 147] and explore recaptioning training images using MLLMs or VLMs [30, 62, 93, 146]. We extend these concepts to build a video data engine and scale our model to function as one strong model for both image and video (§2.2).

Best Embedding Layer Inside the Network. Typically, most vision encoders rely on the last layer to extract features for the task it is trained on. However, when trained on proxy or self-supervised tasks, the last layer is often not the ideal candidate for other tasks. For example, when using image colorization as pretraining objective, [157] showed that the middle layers were better at image classification compared to last layers. Subsequently, in iGPT [14], when trained for next token prediction, intermediate layers performed better at image classification. AIMv1 [28] also showed similar behavior for image based next token prediction with patch normalized MSE loss. Toto [104] showed this can be extended for next token prediction in videos, and intermediate layers are best for image classification, video classification, tracking and robotics. REPA [154] showed this behavior for image generation models, where the intermediate layers of SiT [83] has better linear probing accuracy compared to earlier or later layers. In CLIP models, CLIPer [124] identified that early layers in CLIP possess good spatial understanding. In contrast to these lines of work, in this paper, we first show this behavior is not limited to one class of encoders. Specifically, we show this behavior exists in a spatially self-supervised model [95], generative captioning model [35], and also in our own PE. Then we study this behavior for PE encoder in depth, and show it is possible for CLIP training to produce rich spatial and semantic features in intermediate layers (§3).

Alignment Tuning. We explore alignment tuning for language (§4) and for spatial understanding (§5). For language alignment, we focus on adapting to multimodal large language models (MLLMs); for spatial alignment, we employ self-distillation of the models own features combined with a teacher for locality. In MLLM literature, *midtraining*—i.e., a middle stage of training used to exploit large-scale multimodal data—has been actively studied. LLaVA-OneVision [64], InternVL series [16, 17], QwenVL series [3, 139], and several other leading MLLMs [80, 128] adopt this paradigm. Our PE_{lang} can be seen as a variant of midtraining, but with one critical difference in principle: our goal is *not* to build the best MLLM, but to make the vision encoder the most *general*. Throughout § 4, we benchmark our PE_{lang} across different language models, input resolution, on various tasks for image and video to show this generality. For spatial tasks, we utilize the hidden embeddings

in the intermediate layers. Recently, several works showed the effectiveness of distilling teacher model via representation alignment with cosine similarity. REPA [154] distilled an early layer features of DINO for image diffusion models, RADIO [107] used multi-teacher distillation (DINO, CLIP and SAM). The key idea is to borrow semantic understanding (*e.g.*, CLIP) and spatial understanding (*e.g.*, SAM, DINO) of a pretrained vision encoders. In our $\text{PE}_{\text{spatial}}$, we exploit the intermediate features of PE_{core} for semantics, and a novel way to use SAM for spatial understanding.

7 Conclusion

We have presented Perception Encoders (PE), a family of best-in-class foundation models comprising PE_{core}, PE_{lang}, and $\text{PE}_{\text{spatial}}$. We have shown that PE_{core} can outperform models trained with WebLI and JFT-3B, which were previously the undisputed leaders in zero-shot image recognition, while also excelling in zero-shot video recognition. We have demonstrated that PE_{lang} can be used to build a multimodal language model [19] that is at the forefront of the field in terms of performance. We have established that $\text{PE}_{\text{spatial}}$ can match the long-standing state-of-the-art in object detection with a significantly simpler decoder. Throughout all of this, one conclusion is abundantly clear: Perception Encoder unlocks the potential to scale simple contrastive vision-language pretraining to address a wide range of downstream vision tasks.

Additional Contributors and Acknowledgments. We would like to thank Abhimanyu Dubey, Adel Ahmadyan, Andrew Westbury, Arkabandhu Chowdhury, Azita Shokrpour, Babak Damavandi, Chay Ryali, Cyprien de Lichy, Didac Suris Coll-Vinent, Dong Wang, Filip Radenovic, George Orlin, Han Zou, Harry Tran, Jitendra Malik, Joelle Pineau, Joseph Greer, Kavya Srinet, Kirmani Ahmed, Laura Gustafson, Lu Zhang, Muhammad Maaz, Natalia Neverova, Nicolas Carion, Oleksandr Maksymets, Ramya Raghavendra, Romy Luo, Ronghang Hu, Sam Doud, Sasha Mitts, Sean Bell, Shane Moon, Shuming Hu, Soerian Lieve, Stephane Kasriel, Vanessa Stark, Vignesh Ramanathan, Vivian Lee, Xuan Hu, Yang Li, and Ziyang Wang for their contributions and support for the project. And we thank you, the reader, for reading this far.

A Video Data Engine

A.1 Video Caption

LLM Summarization prompt

LLM Summarization prompt 72 tokens

```
Create a concise caption of a video using the provided metadata, video caption, and frame captions.
TASK: Extract key information from the captions and combine it into an alt text format using single phrase or set of phrases that includes all relevant details.
Steps to Follow:
1. Review the metadata (title and description) for general context, you can rely it for entity names but do not rely on it as the primary source of information for your caption.
2. Blend title / description with video caption and frame captions for the main storyline
3. Extract the most relevant and concise information.
4. Combine extracted information into a alt text format using short phrase or set of phrases with approximately 120 tokens, considering special characters like comma as part of the token count.
5. Prioritize including all key information over sentence structure or grammar.
6. Minimize the use of special characters and focus of key information.
What to Avoid:
- Avoid adding or inferring information not present in the original metadata and captions.
- Avoid using complex sentence structures or prioritizing sentence flow.
Create a concise caption of the video based on the metadata, video caption, and frame captions.
```

A.2 PE Video Dataset Details

PE Video is a dataset that we collected and curated from a licensed data source. The videos are high-resolution and high-quality with a focus on motion. The total number of videos is 1M. Among these, 120K videos have human-refined video captions, and we selected 15K from the 120K videos as a benchmark.

A.2.1 Video Data Filtering Pipeline

The goal of video data filtering is to identify videos that contain motions such as object motion, camera motion, interaction between objects, human actions, sequences of actions, and manipulation of objects, while rejecting videos with static scenes, like landscapes, or those that are artificial or highly edited.

To achieve this, we created a video filtering pipeline consisting of the following steps:

Step 1: Compute motion features. For each video, we compute a list of features from video frames, including frames per second (fps), number of frames, number of I-frames, motion vector magnitude, and motion vector variance, using off-the-shelf tools like OpenCV [9].

Step 2: Extract video frame features. For each video, we uniformly sample three frames and encode them using a DINOv2 model [95] and a SigLIP model [155].

Step 3: LLM Features. For each video, we also run a multimodal large language model (LLM) like Llava-Onevision QwenLM 2 0.5B [64] to extract MLLM features. We composed a list of 26 questions and performed MLLM inference on the videos. The questions can be found here in §A.2.2.

Step 4: Video Quality Scoring. We combine all the features collected so far and use a random forest model to predict a score between 0 and 5. To train the model, we manually annotated approximately 1,000 videos with scores between 0 and 5. A low score indicates that the video is almost static and can be nearly summarized by a single frame, while a high score indicates that there are multiple temporal events in the video, requiring several frames to accurately caption it. We use these annotated videos as training data to fit a random forest model for video quality score prediction.

Step 5: We apply k-means clustering to the videos and rank them within each cluster. By selecting the top-ranked videos from each cluster, we effectively reduce the number of duplicated videos in the final dataset.

A.2.2 LLM Feature Extraction

```
LLM Feature extraction question list
Is the camera capturing the scene static? Reply yes or no.
Is the camera capturing the scene moving? Reply yes or no.
Is the video capturing a landscape? Reply yes or no.
Is the video capturing a static scene? Reply yes or no.
Is the scene captured from a distance? Reply yes or no.
Is the video captured with a drone? Reply yes or no.
Is the video computer-generated? Reply yes or no.
Is the video content abstract? Reply yes or no.
Is there something moving through the scene? Reply yes or no.
Is there someone doing something in the video? Reply yes or no.
Are there several things moving in the video? Reply yes or no.
Is there an object that is being manipulated? Reply yes or no.
Are there animals in the video? Reply yes or no.
Is the scene mostly static? Reply yes or no.
Are things occluding each other in this video? Reply yes or no.
Is there something obstructing the view apart from the watermark? Reply yes or no.
Is there a large number of things in the video? Reply yes or no.
Are there more than 5 different objects in the video? Reply yes or no.
Is it hard to keep track of some entities because they are moving so much? Reply yes or no.
Is someone looking at a phone, a tablet or a computer screen? Reply yes or no.
Are they looking at a phone, a tablet or a computer screen during the whole video? Reply yes or no.
Are there several moving persons in this video? Reply yes or no.
Are there several moving animals in this video? Reply yes or no.
Are there several objects in this video? Reply yes or no.
Are there several similar-looking objects in the video? Reply yes or no.
Do they look similar? Reply yes or no.
```

We use LLaVA-OneVision [76] model to extract LLM features from the videos. For each video, we prompt with 26 different questions to extract features ranging from, "is the video a landscape video?" to, "are there any moving objects in the video?" The features are then used by a random forest model to determine the video quality score.

A.2.3 PVD Benchmark Distribution

Category	Number of videos	Avg. Caption Length
Hand Actions	2143	54.2
Object Interactions	1864	42.6
Food Preparation	1691	56.8
Work Activities	1689	47.8
Outdoor Scenes	1558	50.7
Animals	1423	50.9
Water Scenes	1337	44.6
Object Handling	1307	51.6
Close-up Shots	1122	45.1
Nature Scenes	866	38.4

Table 16 PVD Benchmark Statistics. We created a dataset of 15K videos together with human-verified captions. The videos are motion-centered, covering both first-person and third-person views with a wide coverage of scenes.

Category: *Hand Actions*

Caption: The video captures a closeup shot of person typing on a keyboard. The camera moves from the left side of the keyboard to the right, an animation of the revolving globe and some numbers can be seen in the frame and the video ends.

Category: *Object Interactions*

Caption: The video shows a black and white spiral that is spinning. The spiral is made up of alternating black and white stripes that are evenly spaced and symmetrical.

Category: *Food Preparation*

Caption: The video shows a person cutting an green color item into small pieces. They are using a knife to slice the pickle into thin pieces, and then chopping those pieces into smaller cubes. The person is working on a wooden cutting board, and the Hands are visible from the left side of the frame with pink nail paint on their nails.

Category: *Work Activities*

Caption: The video shows a person using a shovel to clean the ashes from a fireplace. They are scooping up the ashes and removing them from the fireplace.

Category: *Outdoor Scenes*

Caption: The video shows a tall, pointed structure in the middle of a field. and the structure is surrounded by trees and other vegetation. The field is divided into sections, with some areas covered in green grass and others covered in white material. The video shows the structure and the field from a distance, with the camera moving around it.

Category: *Animals*

Caption: The video shows a white and gray adult cat and two kittens. The adult cat is grooming the kitten closest to it with its tongue, and the kitten is looking around. A hand reaches out from the frame's upper left to pet the two kittens.

Category: *Water Scenes*

Caption: The video shows a large school of fish swimming in a water body towards the right frame. The camera too pans a little to the right.

Category: *Object Handling*

Caption: The video shows a person putting a bowl of something into an oven. The person then closes the oven door. The background is blurry.

Category: *Close-up Shots*

Caption: The video shows a white counter with two brown buckets and a yellow bucket. Then a person's right hand wearing a green glove enters the frame from top right side and place a yellow flower near to yellow watering can. The person then places the flower, in front of the buckets and exits the frame. In the background is a brown wall, and the camera is static throughout the clip.

Category: *Nature Scenes*

Caption: The video shows a pile of branches and leaves on fire in a field. The fire is burning brightly, with flames licking at the edges of the pile. The smoke from the fire rises into the air, billowing up into the sky.

Figure 18 More PE Video Dataset Examples. For each of the ten categories, we randomly pick one video and show its video caption. The captions were generated by our video data pipeline and then refined by human annotators.

B Implementation Details

B.1 PE Core

We provide additional implementation details for building PE_{core}. Our implementation is based on OpenCLIP[4].

B.1.1 Architecture and Training Setups

Model Architecture. Following CLIP, PE_{core} comprises a Transformer-based [137] vision and a text encoder. We employ customized Transformer configurations as detailed in Tab. 17. For pooling, we an attention pooling block in the style of SigLIP [155] *with 8 heads* from the last-layer feature to construct image and video embeddings. Regarding positional embedding, we use 2D RoPE [123] for relative positional embeddings and 2D learnable absolute positional embeddings (abs) the same size as the model's input resolution. We interpolate positional embeddings to enable support for various resolutions beyond the default. The text context length is 72 for G-scale and 32 for B and L-scale models. Originally a bug, we find it optimal to *not disable the class token* when using attention pooling for smaller models. Thus, the B and L models use a class token, then the attention pooling layer probes all features at once (class token included). Finally, we use an input mean and standard deviation of $(0.5, 0.5, 0.5)$ for simplicity.

Scale	Tower	Params	Width	Depth	MLP	Heads	CLIP Dim	Pooling	Positional Embedding	Resolution & Context Len	Patch Size	Class Token Register
B	Vision	0.09B	768	12	3072	12	1024	Attn Pool	RoPE+Abs	224	16	✓
	Text	0.31B	1024	24	4096	16		EOS Token	Abs	32	-	-
L	Vision	0.32B	1024	24	4096	16	1024	Attn Pool	RoPE+Abs	336	14	✓
	Text	0.31B	1024	24	4096	16		EOS Token	Abs	32	-	-
G	Vision	1.88B	1536	50	8960	16	1280	Attn Pool	RoPE+Abs	448	14	✗
	Text	0.47B	1280	24	5120	20		EOS Token	Abs	72	-	-

Table 17 PE Model Configurations with full details.

PE Core Training. As discussed in §2.4, the training of PE_{core} involves three stages: 1) image pretraining; 2) image and video finetuning; and 3) an additional model distillation for smaller models. These three stages work together to develop a robust and effective PE_{core} model.

We first provide training recipes for 1) image pretraining in Tab. 18 and 2) video finetuning in Tab. 19.

config	values
optimizer	LAMB
β_1, β_2	(0.9, 0.95)
weight decay	0.05
learning rate	2e-3
batch size	131,072
warm-up steps	2K
training steps	443K (B, L) / 656K (G)
data quantity	5.4B
samples seen	58B (B, L) / 86B (G)
max logit scale	100
mask reg ratio	0.4
mask reg batch	8192
progressive res	112-160-224 (B) 98-154-224-336 (L) 98-154-224-336-448 (G)
data aug	aspect jitter ar(0.75,1.33) rand crop s(0.08,1) color jitter j(0.32,0,0.32,0) hflip p(0.5)

Table 18 Image Pretraining.

config	values
optimizer	LAMB
β_1, β_2	(0.9, 0.95)
weight decay	0.05
learning rate	1e-6
batch size	4096
warm-up steps	2K
training steps	5.4K
data quantity	22M
samples seen	22M
max logit scale	100
number of frames	8
data aug	aspect jitter ar(0.75,1.33) rand crop s(0.08,1) color jitter j(0.32,0,0.32,0) hflip p(0.5)

Table 19 Video Finetuning.

config	values
optimizer	LAMB
β_1, β_2	(0.9, 0.95)
weight decay	0.05
learning rate	1e-6
batch size	16384
warm-up steps	2K
training steps	269K
data quantity	5.4B
samples seen	4.4B
max logit scale	100
teacher logit scale	200 (§C.3)
data aug	None

Table 20 Distillation.

After training the largest G-scale model, we train the smaller models with image pretraining, then distill with image distillation in Tab. 20, then finally apply video finetuning at the end.

[4] https://github.com/mlfoundations/open_clip

B.1.2 Zero-Shot Classification and Retrieval

Zero-Shot Evaluation on Images and Videos. We use CLIPBench[5] for zero-shot classification and retrieval benchmarking. The benchmark datasets and splits are obtained from the original dataset websites or HuggingFace. We extend the CLIPBench zero-shot evaluation to include video datasets such as MSR-VTT and Kinetics, and will release our model checkpoints, evaluation code, and scripts for reproducibility.

Prompt Design. For zero-shot image-text and video-text retrieval, we rely solely on the original captions without any additional prompts. In contrast, for zero-shot classification, we utilize task-specific prompts graciously provided by the InternVL [17] authors. All additional prompts will be released.

For example, we employ specific prompts for zero-shot image classification on various ImageNet benchmarks (e.g., ImageNet val, ImageNet v2) and video classification on Kinetics datasets (e.g., K400, K600, K700).

Zero-Shot Image Classification Prompts - ImageNet

```
a bad photo of a {c}. a photo of many {c}. a sculpture of a {c}. a photo of the hard to see {c}. a low resolution photo of the {c}. a rendering of a {c}. graffiti of a {c}. a bad photo of the {c}. a cropped photo of the {c}. a tattoo of a {c}. the embroidered {c}. a photo of a hard to see {c}. a bright photo of a {c}. a photo of a clean {c}. a photo of a dirty {c}. a dark photo of the {c}. a drawing of a {c}. a photo of my {c}. the plastic {c}. a photo of the cool {c}. a close-up photo of a {c}. a black and white photo of the {c}. a painting of the {c}. a painting of a {c}. a pixelated photo of the {c}. a sculpture of the {c}. a bright photo of the {c}. a cropped photo of a {c}. a plastic {c}. a photo of the dirty {c}. a jpeg corrupted photo of a {c}. a blurry photo of the {c}. a photo of the {c}. a good photo of the {c}. a rendering of the {c}. a {c} in a video game. a photo of one {c}. a doodle of a {c}. a close-up photo of the {c}. a photo of a {c}. the origami {c}. the {c} in a video game. a sketch of a {c}. a doodle of the {c}. a origami {c}. a low resolution photo of a {c}. the toy {c}. a rendition of the {c}. a photo of the clean {c}. a photo of a large {c}. a rendition of a {c}. a photo of a nice {c}. a photo of a weird {c}. a blurry photo of a {c}. a cartoon {c}. art of a {c}. a sketch of the {c}. a embroidered {c}. a pixelated photo of a {c}. itap of the {c}. a jpeg corrupted photo of the {c}. a good photo of a {c}. a plushie {c}. a photo of the nice {c}. a photo of the small {c}. a photo of the weird {c}. the cartoon {c}. art of the {c}. a drawing of the {c}. a photo of the large {c}. a black and white photo of a {c}. the plushie {c}. a dark photo of a {c}. itap of a {c}. graffiti of the {c}. a toy {c}. itap of my {c}. a photo of a cool {c}. a photo of a small {c}. a tattoo of the {c}.
```

Zero-Shot Video Classification Prompts - Kinetics

```
a photo of {c}. a photo of a person {c}. a photo of a person using {c}. a photo of a person doing {c}. a photo of a person during {c}. a photo of a person performing {c}. a photo of a person practicing {c}. a video of {c}. a video of a person {c}. a video of a person using {c}. a video of a person doing {c}. a video of a person during {c}. a video of a person performing {c}. a video of a person practicing {c}. a example of {c}. a example of a person {c}. a example of a person using {c}. a example of a person doing {c}. a example of a person during {c}. a example of a person performing {c}. a example of a person practicing {c}. a demonstration of {c}. a demonstration of a person {c}. a demonstration of a person using {c}. a demonstration of a person doing {c}. a demonstration of a person during {c}. a demonstration of a person performing {c}. a demonstration of a person practicing {c}.
```

Evaluation Method. Several works use different input transformations for different datasets when evaluating zero-shot performance (e.g., [31, 126, 134, 155]). To be as fair as possible, we follow [126] in evaluating with two transformations—center crop and non aspect ratio preserving resize ("squash")—and report the max between the two for all models and all datasets we evaluate. Additionally, ObjectNet has a red border around every image to facilitate deduplication, which we remove for evaluation. Finally, we follow [17] in using *retrieval reweighting* (DSL), applying the softmax score distribution to the similarities used for retrieval:

$$\texttt{scores} = \texttt{scores} * \texttt{softmax(scores, dim=0)} \tag{1}$$

This slightly improves retrieval for most models, so we do it for all models we evaluate for fairness. Notably, we were able to reproduce the reported numbers for most papers with these techniques, but for cases where we could not, we default to the reported number.

B.2 PE: Language Alignment

We provide details of the MLLM experimental setup in § 4. We describe *data*, *model*, and *training* separately.

Data. Our MLLM training contains *warmup* data and *supervised finetuning (SFT)* data. Our warmup data is a 1M subset image-text pairs of our PE_{core} pretraining dataset. For SFT data, we use a diverse data

[5] https://github.com/LAION-AI/CLIP_benchmark

mix consisting of 2.6M unique samples. This dataset is composed of 1.7M [6] visual QAs samples from the Cauldron [63], 0.5M grounded QA pairs from Visual Genome [58], Flickr-Entities [100] and Densely Captioned Images [135], 0.1M image-captioning pairs from COCO [74] and 0.3M text-only samples. This comprehensive data mix allows us to thoroughly assess our model's capabilities in various MLLM tasks.

Model. As described in § 4.1, we use a simple vision-language model architecture where a vision encoder and a pretrained decoder-only LLM are connected by a vision projector. For all tables, we use either Llama3.1-instruct 8B or QwenLM 2.5-instruct 7B as a language model, and 2-layer MLP as a vision projector. For fair comparison, we use the native resolution for image input. During inference, we evaluate the models on video tasks in *zeroshot* manner: We concatenate all video frames into a sequence and feed to language model, without seeing video samples during SFT. For all video tasks, we use 8 frames with the same native resolution of height and width. For PE_{core} and PE_{lang}, this makes $448 \times 448 \times 8$ input and $32 \times 32 \times 8$ vision tokens.

Training. MLLM training consists of *warmup* and *supervised finetuning (SFT)* stages. In both stages, we freeze vision encoder and train vision projector and LLM. During warmup stage, we use a global batch size of 128 with a learning rate of 1×10^{-4}. We gradually increase the learning rate from 1×10^{-6} to 1×10^{-4} over 120 steps, and follow a cosine learning rate decay schedule to train a total of 8,000 steps. During SFT stage, we use a global batch size 256 with a learning rate of 1×10^{-5}. Similar to the warmup, we gradually increase the learning rate from 1×10^{-7} to 1×10^{-5} over 300 steps, and follow a cosine learning rate decay schedule to train a total of 12.5K steps. We truncate text-sequences longer than 2,048 tokens on top the visual tokens. This makes the maximum sequence length to be `(num. vision tokens)` $+ 2,048$. With 448×448 input resolution and patch size of 14, we set the maximum sequence length to $1,024 + 2,048 = 3,072$. To represent bounding boxes on output side for image grounding tasks, we simply use text tokens to represent each bounding box: each coordinate is normalized between `000` and `999`, in "`[x, y, x, y]`" box format for top-left and bottom-right corners (*e.g.*, `[012, 122, 633, 782]`).

For all baselines, we search for the **best** intermediate layer features to adapt to LLM. We search over $\{-1, -2, -4, -6, -8, -10, -12, -14, -16, -18, -20, -40\}$ layers (counting from last) and report the best result in average over OCR/Chart/Document Q&A, Visual Q&A, Image Captioning and Video Understanding.

B.3 PE: Spatial Alignment

B.3.1 Training Details

Loss Functions. For self-aligning to frozen PE_{core}G layer 41 features (L_{core}), we minimize cosine similarity:

$$L_{\text{core}} = \frac{1}{n_{\text{tok}}} \sum \left(\frac{(S_{50})(T_{41})^T}{||S_{50}|| \cdot ||T_{41}||} \right) \tag{2}$$

where S_{50} denotes the last layer features of the student, T_{41} denotes frozen layer 41 features from PE_{core}G, and n_{tok} represents the number of tokens. Note that we chose 41 fairly arbitrarily (it is layer 40 when written with indexing from 0). Judging by Fig. 8, any layer around 40 should work (and 39 may be slightly better).

For the encouraging locality loss (L_{loc}), we compute the pairwise cosine similarity between a model's own tokens and itself. This forms a "spatial correspondence map" for what tokens should be considered similar. We then compute the same for the student, and minimize the difference between the two with MSE loss:

$$L_{\text{loc}} = \frac{1}{n_{\text{tok}}^2} \sum \left(\frac{(S_{50})(S_{50})^T}{||S_{50}||^2} - \frac{(T_{\text{SAM}})(T_{\text{SAM}})^T}{||T_{\text{SAM}}||^2} \right)^2 \tag{3}$$

where T_{SAM} denotes the "SAM Mask Logits" constructed in §5.2. We also find using a temperature (t) on the SAM teacher's pairwise cosine similarity term (x) useful: $e^{t(x-1)}$. The full loss is $L_{\text{spatial}} = L_{\text{core}} + L_{\text{loc}}$.

Hyperparameters. In Tab. 21 we show the training hyperparameters for spatial alignment, finetuned on top of the initial PE_{core}G checkpoint. Then in Tab. 22 and Tab. 23, we show the settings for the two teachers and losses. Note that when running the teachers, we run them on the exact same image as the student (same data

[6] We excluded multi-images samples.

aug and all). Additionally, because the SAM 2.1 teacher operates at a resolution of 1024, we upsample the image, generate the mask logits, and then downsample the result. Both teachers are frozen.

config	values
optimizer	LAMB
β_1, β_2	(0.9, 0.95)
weight decay	0.05
learning rate	5e-4
batch size	12,288
warm-up steps	0
training steps	24K
data quantity	5.4B (PE_{core} PT Data)
samples seen	300M
resolution	448
mask ratio	0.75
mask size	2×2 tokens
droppath	0.4
layerscale	0.1
data aug	aspect jitter ar(0.75,1.33) color jitter j(0.32,0,0.32,0) hflip p(0.5)

Table 21 Spatial Alignment.

config	values
model	SAM 2.1-L
layer	mask logits
resolution	1024 (interp→448)
loss	Eq. 3
loss weight	1
temperature	20
sample points	32×32 (1024)
pred iou threshold	0
stability score threshold	0
mask threshold	0

Table 22 SAM 2.1 Teacher.

config	values
model	$PE_{core}G$
layer	41
resolution	448
loss	Eq. 2
loss weight	1

Table 23 $PE_{core}G$ Teacher.

B.3.2 Visualization Method

To visualize the features in Fig. 17 and Fig. 20, our goal is to map a 1536-dimensional space down to 3 dimensions to view how the model encodes each token in relation to each other. One naive approach would be to apply PCA with 3 dimensions across all token in the image. However, we find this alone can be misleading.

Specifically, if the model has rich semantics, it should be the case that most of those 1536 features have some useful information in them. Some of that information could be spatially contiguous, some of it not. We want PCA to only select the *spatially contiguous* information, since we are trying to evaluate the spatial quality of the features. However, naively applying PCA will not necessarily do that, especially for models with information aggregated in "global tokens" (§5.1). Despite these tokens carrying important information, they are not spatially contiguous. Thus, if PCA dedicates a large portion of its 3 dimensions to global tokens, the features will *look* like their spatial quality is bad, despite the features containing good spatial information.

So, how do we select for only the *spatially contiguous* information to visualize? The answer is simple: by definition, the spatially contiguous information will be… spatially contiguous. To keep the spatially contiguous information while lowering the impact of the global tokens, we can simply apply a low pass filter to the features (specifically, a gaussian blur with kernel size 3 and a σ of 1). To retain the detail of the original features, we can average the two together. Thus, to visualize features, we use the 3D PCA of the of the following. x denotes the model's output features, and $g(x)$ denotes gaussian blur.

$$0.5x + 0.5g(x, k = 3, \sigma = 1) \tag{4}$$

We show the impact of this in Fig. 19. Blurring the features make them appear more detailed! In reality, that information was always there, just PCA did not show it. Thus, great care must be taken when visualizing high dimensional feature spaces. If they were easy to map to 3 dimensions—you wouldn't need 1536 of them!

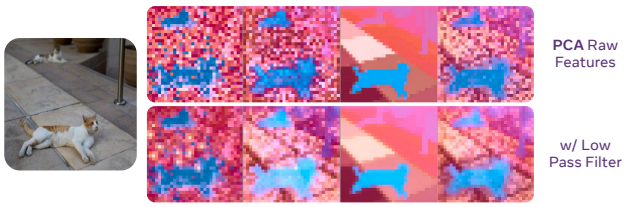


Figure 19 Feature Visualization Ablation. With raw features (top row), PCA misses spatially contiguous parts of the feature space and instead focuses on global tokens (which carry information but are not spatially coherent). By applying a simple low pass filter (bottom row), we can reveal spatial information that PCA originally missed (see column 2: with raw features, the background looks like a mess, with the low pass filter the tiles become visible).

Then, to map the PCA dimensions to RBG pixel values, we map each PCA component to a corresponding channel in LCh color space, then convert those LCh colors to RGB to get the final image. Note that we use LCh instead of RGB directly for aesthetic reasons, and also because LCh is a cylindrical color space—where smooth changes to the values look like smooth changes in colors to humans—and thus is easier to discern.

B.3.3 Frozen Feature Dense Prediction

We discuss the detailed settings of the results for dense prediction with frozen features in Tab. 13. Each model is evaluated with its native resolution up to 448 or 448 (whichever is optimal).

Zero-Shot Tracking. We evaluate our pretrained models on label propagation task using the protocols in [50, 104] on DAVIS dataset [101]. This evaluation does not require any finetuning or probing, therefore preserves the spatial features in the model. Following Toto [104], we use the features from the last $n = 7$ frames to find the nearest neighbor patch in the current frame, and then propagate the masks from the previous frames to the current frame. Note that this evaluation method does not require any training.

Semantic Segmentation. For semantic segmentation, we evaluate our pretrained models on ADE20K [161] semantic segmentation task. We use a linear layer and convolutional layer to map intermediate spatial features to segmentation masks following [95]. The models are evaluated and then features are resized to 518×518. We only use features from single layer. The probing layers are finetuned with AdamW [81] with a learning rate of 0.001.

Depth Estimation. For depth estimation on NYUv2 [119], we follow [73, 95]. We use a DPT-head [106] on top of our frozen pretrained model and use only single layer features. We scale the size of the DPT-head for each models based on the hidden size for each architecture. Because NYU is a small dataset and the models we evaluate are large, we observe the results for most models are noisy and prone to overfitting. Thus, for fair comparison we train *all models* for 20 epochs and for *all models* take the lowest validation loss over all epochs.

Frozen Detection. For the frozen feature detection results presented in §3, we evaluated using Mask R-CNN [41] as a probe. We used a resolution of 1024 for Fig. 8 and 768 for the remainining experiments in §3. Because the backbones were frozen, we did not add any global attention and instead simply tiled the input image with a window size of 32 for the 1024px experiments and 24 for the 768px experiments. All models were interpolated to patch 16. Finally, the backbones were frozen and only the FPN and R-CNN heads trained for 15 epochs on COCO with a stepwise decay LR without drop path.

B.3.4 End-to-End Finetuning Detection and Segmentation

We provide a detailed discussion of settings of end-to-end finetuning on detection and segmentation presented in Tab. 14. The hyperparameters can be found in Tab. 24. We find that the default 100-epoch protocol in ViTDet [70, 144] causes overfitting problems in COCO experiments especially for billion-level parameter vision encoders, so we tune the training epochs, learning rate, drop path and learning rate decay accordingly.

The LVIS experiment setting is the same as COCO except all L-size models use learning rate of 2e-4 and all g-size and G-size models use 75 epochs.

config	values
optimizer	AdamW
optimizer momentum	(0.9, 0.999)
weight decay	0.1
learning rate	→
learning rate schedule	Step-wise decay
learning rate decay	→
batch size	64
image size	1024 × 1024
augmentation	LSJ [0.1, 2.0]
epochs	→
drop path	→
postional embedding	abswin [7]
patch size	16
window size	→
global window index	→

model	lr	epochs	drop path	lr decay	layers	global window index	window size
OpenAI CLIP-L	1e-4	100	0.4	0.8	24	(5, 11, 17, 23)	14
MetaCLIP-L	1e-4	100	0.4	0.8	24	(5, 11, 17, 23)	14
MetaCLIP-G	5e-5	75	0.5	0.9	48	(11, 23, 35, 47)	14
SigLIP-so	1e-4	100	0.4	0.8	27	(2, 10, 18, 26)	14
EVA02-L	1e-4	100	0.4	0.8	24	(5, 11, 17, 23)	14
MAE-L	1e-4	100	0.4	0.8	24	(5, 11, 17, 23)	14
SigLIP2-so	1e-4	100	0.4	0.8	27	(2, 10, 18, 26)	14
SigLIP2-g	5e-5	75	0.5	0.9	40	(9, 19, 29, 39)	14
DINOv2-L	1e-4	100	0.4	0.8	24	(5, 11, 17, 23)	32
DINOv2-g	5e-5	36	0.5	0.9	40	(9, 19, 29, 39)	32
PE$_{\text{core}}$G	5e-5	75	0.5	0.9	50	(12, 24, 36, 49)	32
PE$_{\text{spatial}}$G	5e-5	36	0.5	0.9	50	(12, 24, 36, 49)	32

Table 24 Settings for End-to-End Finetuning Detection and Segmentation.

B.3.5 System-Level Comparison on Detection

We describe our implementation for system-level comparison to the state-of-the-arts on COCO object detection in Tab 15. Our implementation is based on the DETA repository[7]. We replace the vision encoder with our $\text{PE}_{\text{spatial}}$ and maintain the same hyperparameters as in the end-to-end finetuning settings, while keeping the detector unchanged. The training process consists of three stages:

Test-Time Aug	AP_{box}
No TTA	64.9
+ More Queries	65.0
+ SoftNMS [6]	65.3
+ Flip Aug	65.4
+ Multiscale Aug	**65.5**

Table 25 Test-Time Aug for system-level comparison on COCO in Tab. 15.

1. **Initial Training**: Train on Objects365 for 12 epochs with an image resolution of 1024×1024, a total batch size of 256, and a learning rate of 2e-4, which is divided by 10 at the 10th epoch.
2. **Increasing Resolution**: Continue training on Objects365 for 6 epochs with a resolution of 1536×1536, a total batch size of 128, and a learning rate of 5e-5, which is divided by 10 at the 5th epoch.
3. **Finetuning**: Finetune on the COCO dataset for 12 epochs with an image resolution of 1536×1536, a total batch size of 64, and a learning rate of 5e-5, which is divided by 10 at the 10th epoch.

We apply a series of test-time augmentation techniques to further improve the performance, detailed in Tab. 25.

C Additional Results

C.1 PE_{core}: Robust Image Pretraining

In Tab. 26, we present the raw data for the robustness metrics in Fig. 2. Across the board, each change improved almost all metrics (with the exception of progressive resolution slightly hurting the average and mask regularization slightly hurting ImageNet Adversarial). The fact that there were no tradeoffs to these changes, indicate that their improvements to the features are general. This could be why most of these changes improved performance for downstream tasks as well.

Note that in §2.1, we only discuss changes that we know to work. There are several changes that we have tried that do not work (i.e., do not improve performance or lower performance). For instance: average pooling instead of using a class token, increasing the text tower size, using hue or contrast jitter, and maintaining the same resolution throughout training but dropping tokens instead of progressive resolution (FLIP-style).

We also find increasing batch size and increasing training iterations for an L scale model to have equivalent effects. This is in contrast to the batch size scaling observed by [155], but it is possible that this difference is down to a hyperparameter issue.

		Zero-Shot Classification						
	Step	**Avg Class.**	ImageNet *val* [24]	ImageNet *v2* [109]	ObjectNet *IN Classes* [4]	ImageNet *Adversarial* [45]	ImageNet *Renditions* [44]	ImageNet *Sketch* [138]
1	Baseline	75.3	78.9	71.9	73.7	68.3	91.1	67.8
2	Progressive Resolution	75.1	78.9	71.8	72.4	69.9	90.5	67.0
3	High Batch Size	76.2	79.5	72.8	74.1	71.8	91.0	68.1
4	LAMB and High LR	76.9	79.9	73.3	74.3	73.5	91.5	68.6
5	High Resolution (336)	78.3	80.4	73.8	75.6	79.2	92.0	68.8
6	2D RoPE	79.2	80.7	74.1	77.4	80.9	92.7	69.4
7	Attention Pooling	80.1	81.0	74.8	78.4	82.9	93.4	69.9
8	Data Augmentation	80.8	81.1	75.2	80.8	83.1	93.5	71.2
9	Mask Regularization	80.9	81.3	75.3	80.9	82.8	93.8	71.2

Table 26 Robust Image Pretraining Full Results. Raw results for the robustness metrics metrics in Fig. 2. Almost every change improves every metric, but some metrics are improved more than others (e.g., ObjectNet and ImageNet-A).

Video Data Size	Image Zero-Shot								Video Zero-Shot							
	Average Image	ImageNet *val* [24]	ImageNet *v2* [109]	ObjectNet *IN Classes* [4]	ImageNet *Adversarial* [45]	MS-COCO *txt→img* [74]	MS-COCO *img→txt* [74]	**Average Video**	Kinetics *400* [53]	Kinetics *600* [53]	Kinetics *700* [53]	UCF 101 [122]	HMDB 51 [60]	MSR-VTT *txt→vid* [148]	MSR-VTT *vid→txt* [148]
0M	77.0	83.9	78.6	86.6	90.3	52.1	70.3	57.0	70.3	69.4	61.6	78.5	47.4	40.5	31.4
3M	77.7	84.1	78.8	86.6	90.9	53.3	74.2	61.6	72.4	72.2	64.2	88.5	53.8	42.8	37.6
6M	78.0	84.2	79.0	86.7	91.1	54.0	72.7	63.6	73.5	73.4	66.0	88.9	54.6	44.9	43.6
8M	78.4	84.2	79.2	87.0	91.6	54.9	73.6	64.8	74.5	74.5	67.7	89.5	55.3	46.9	45.5
11M	78.6	84.2	79.2	87.2	91.8	55.4	73.8	65.2	75.1	75.0	67.6	89.7	55.6	47.7	45.8
14M	78.8	84.2	79.2	87.5	91.9	55.7	74.3	65.5	75.4	75.3	67.9	89.9	55.8	47.8	46.3
17M	78.9	84.2	79.2	87.7	92.0	55.8	74.3	65.8	75.7	75.5	68.2	90.2	56.0	48.3	46.7

Table 27 Scaling Video Data. Increasing the number of synthetic video data generated by our proposed video data engine consistently enhances the performance of image and video classification and retrieval tasks.

C.2 PE_{core}: Video Data Scaling

The detailed video data scaling results are presented in Tab. 27. Our experiments demonstrate that increasing the number of synthetic video data generated by the proposed video data engine enhances the performance of classification and retrieval on both image and video benchmarks. On image benchmarks, while improvements on ImageNet val and v2 plateaued earlier compared to ObjectNet and ImageNet Adversarial, MS-COCO retrieval performance continued to show gains. On video benchmarks, scaling synthetic video data consistently yields better performance for both classification and retrieval tasks. We expect that further scaling up the video data with our video data engine will continue to drive performance improvements.

C.3 PE_{core}: Smaller Models

Model	Teacher's Temp	Model Scale	Zero-Shot Classification						
			Avg Class.	ImageNet *val* [24]	ImageNet *v2* [109]	ObjectNet *IN Classes* [4]	ImageNet *Adversarial* [45]	ImageNet *Renditions* [44]	ImageNet *Sketch* [138]
vanilla pretrained model	-	B	66.2	74.2	67.4	62.5	50.2	83.0	59.8
distillation	×2	B	65.2	71.8	65.5	61.4	50.2	83.6	58.6
	×1	B	68.0	74.9	68.1	64.7	54.1	85.3	61.1
	×0.7	B	68.2	75.1	68.2	65.3	54.4	85.1	61.3
	×0.5	B	**68.3**	75.2	68.2	65.3	54.2	85.2	61.4

Table 28 Ablation Study on Teacher's Distribution Temperature. We evaluate the effect of varying temperatures on the teacher's distribution, using a pretrained vanilla CLIP model (ViT-B/14, resolution 224) as a baseline (details in §2.1). The models are finetuned via distillation with a short schedule of 50K steps.

Ablation: Distillation Temperature. To optimize the performance of smaller models (B and L-scales in Tab. 4), we utilize a distillation finetuning approach with PE_{core}G as the teacher model. During this process, both student and teacher models encode image and text inputs to compute image-to-text and text-to-image similarity distributions, similar to CLIP training [103]. The student's distributions are then optimized to match those of the teacher by minimizing KL-divergence loss on both image-to-text and text-to-image similarity distributions.

We find that using a fixed and smaller temperature (i.e., higher logit scale), which controls the range of logits in the softmax, significantly enhances the effectiveness of distillation. This results in a sharper distribution for the teacher's distributions. In contrast, the student's temperature remains learnable, consistent with our pretraining procedure and CLIP training.

In Tab. 28, we present an ablation study examining the impact of temperature on the teacher's distribution. For this analysis, we utilize a pretrained *vanilla* CLIP model (ViT-B/14, resolution 224), which serves as a baseline for comparison (see §2.1 for details). The models are finetuned using distillation with a concise schedule of 50K steps. Notably, our results show that employing a smaller temperature for the teacher's distributions yields improved performance on zero-shot ImageNet benchmarks.

Building strong smaller models. In Tab. 29, we demonstrate our step-by-step training strategy for building strong smaller models at the L scale, as discussed in §2.4. Specifically, we outline our approach to image pretraining,

[7] https://github.com/jozhang97/DETA

Model	Stage	Average Image	ImageNet val [24]	ImageNet v2 [109]	ObjectNet IN Classes [4]	ImageNet Adversarial [45]	MS-COCO txt→img [74]	MS-COCO img→txt [74]	Average Video	Kinetics 400 [53]	Kinetics 600 [53]	Kinetics 700 [53]	UCF 101 [122]	HMDB 51 [60]	MSR-VTT txt→vid [148]	MSR-VTT vid→txt [148]
		Image Zero-Shot							*Video Zero-Shot*							
SigLIP2-L/16 [134]	-	76.0	83.1	77.4	84.4	84.3	55.3	71.4	56.2	65.3	62.5	56.8	86.7	49.3	41.5	31.4
PE_{core}L	image pretraining	75.1	82.9	76.8	81.8	85.6	53.0	70.4	59.0	68.0	67.7	58.5	85.5	57.7	42.0	33.4
PE_{core}L	+image distillation from PE_{core}G	77.6	**83.6**	**78.1**	84.4	88.9	56.0	74.7	64.5	73.0	72.6	64.8	86.5	58.0	47.9	48.4
PE_{core}L	+video finetuning	**78.0**	83.5	77.9	**84.7**	**89.0**	**57.1**	**75.9**	**65.3**	**73.4**	**72.7**	**65.3**	**87.1**	**58.5**	**50.3**	**50.1**

Table 29 Building Strong Smaller Models. This table illustrates the step-by-step process of developing the PE_{core}L 336px model, as outlined in §2.4. Starting with the pretrained PE_{core}L, both image distillation, along with video finetuning, enhance performance across image and video benchmarks, resulting in a unified L-scale model.

image distillation, and video finetuning, and distillation. Leveraging the robust foundation established by our pretraining techniques (§2.1), we show that distilling from PE_{core}G, our strongest unified perception encoder, yields improvements on both image and video benchmarks. Furthermore, a short-scheduled video finetuning provides an additional boost in performance on both benchmarks.

C.4 PE_{lang}: Additional Results

Analogous to Tab. 10, in Tab. 30, we compare PE_{core} and PE_{lang} with *dynamic resolution* setting [75, 80]. More specifically, we use up to 4 tiles, following after a *thumbnail*, which is a whole image resized into 448×448. With the maximum number of tiles of 4, the model can cover $\{1 \times 1, 1 \times 2, 1 \times 3, 1 \times 4, 2 \times 1, 2 \times 2, 3 \times 1, 4 \times 1\}$ tile ratios. Similar to the Tab. 10, 11, 12 in the main paper, we show that PE_{lang} largely outperforms the baseline vision encoders by large margins across all categories of MLLM tasks. Note that PE_{lang} has been alignment-tuned with native resolution input, as opposed to *e.g.*, InternViT 2.5, which has been midtrained with dynamic tiling, which shows PE_{lang}'s strong generality for different input formats.

Next, in Tab. 31, 32, 33, we show the breakdowns of RefCOCO/+/g [54] with Llama 3.1-instruct 8B as language model, Qwen2.5 LM 7B as language model, and with Llama 3.1-instruct 8B and dynamic tiling (4+1), respectively. In our SFT data, we have VisualGenome [58], DCI [135], and Flickr30K [100] as grounding datasets, and RefCOCO/+/g are unseen. We therefore report zeroshot performance of the MLLMs to evaluate spatial understanding capability of the vision encoders. Overall, PE_{lang} L or G show the best performance across all RefCOCO splits, except with Qwen2.5 LM. This is because (1) InternViT 2.5 6B is midtrained with Qwen2 LM, and (2) during pre/mid-training the training data of RefCOCO/+/g are seen.

Model	Encoder Params	Resolution / Patch Size	Avg. OCR QA	ChartQA Acc. [160]	DocVQA Acc. [88]	Info. QA Acc. [89]	AI2D Acc. [55]	Avg. VQA	TextVQA Acc. [121]	OK-VQA Acc. [115]	POPE Acc. [71]	VQAv2 Acc. [38]	Avg. Cap.	Flicker CIDEr [152]	COCO CIDEr [74]	No Cap CIDEr [1]	Avg. Ground. RefCOCOg/+ [54]	Avg. Video	VideoMME Acc. [36]	STAR Acc. [143]	TGIF-QA Acc. [51]	EgoSchema Acc. [87]	MVBench Acc. [66]	PerceptionTest Acc. [102]
			OCR / Chart / Doc. Q&A					*Visual Q&A*					*Captioning*						*Video*					
256 Tokens per Tile																								
MetaCLIP-L [147]	0.3B	224/14	61.8	71.1	62.5	40.2	73.3	74.6	65.3	64.9	88.5	79.8	113.4	90.4	133.5	116.2	67.1	48.0	44.8	47.1	62.7	39.0	46.0	48.3
MetaCLIP-G [147]	1.8B	224/14	60.3	68.1	61.3	39.1	72.8	74.9	65.4	65.9	88.2	80.1	114.2	91.8	134.4	116.5	66.0	49.0	46.5	46.5	62.5	45.0	44.7	48.9
PE_{lang} G†	1.7B*	224/14	70.2	79.8	79.1	47.5	74.6	76.0	70.6	64.3	88.3	80.6	116.3	92.0	136.4	120.5	69.5	56.6	49.0	55.9	69.9	61.2	50.0	53.6
576 Tokens per Tile																								
CLIP [103]	0.3B	336/14	69.6	76.8	78.2	50.3	72.9	76.3	71.8	64.9	88.0	80.4	114.0	90.9	134.4	116.6	68.5	50.8	46.6	52.2	65.0	44.6	46.3	49.9
AIMv2-L [35]	0.3B	336/14	66.7	74.1	74.9	45.2	72.4	77.4	73.5	65.6	89.0	81.7	116.4	92.5	137.1	119.5	66.6	54.1	43.4	54.3	70.6	56.0	47.3	52.7
SigLIP2-so [134]	0.4B	384/16	55.5	61.4	54.9	33.3	72.3	76.5	70.1	66.0	88.6	81.2	118.0	**95.8**	138.3	119.8	66.5	54.3	44.9	52.8	66.8	58.6	49.6	53.3
SigLIP2-g-opt [134]	1.1B	384/16	56.2	63.1	55.3	34.0	72.4	77.0	70.3	**66.7**	89.6	81.6	117.7	94.9	137.8	120.3	66.5	53.9	46.2	53.9	66.6	53.8	48.5	54.7
PE_{lang} G†	1.7B*	336/14	77.5	82.1	88.5	61.8	77.4	79.7	80.2	66.4	89.8	82.5	120.3	97.4	140.2	123.2	71.9	59.8	49.4	62.7	74.1	64.0	53.1	55.6
1024 Tokens per Tile																								
SigLIP2-so [134]	0.4B	512/16	56.9	66.0	56.5	34.3	70.9	76.4	69.9	66.2	88.4	81.2	117.8	94.7	137.8	120.9	67.8	46.2	47.0	44.9	66.7	39.2	34.5	45.1
PE_{core} L	0.3B	448/14	67.1	72.4	78.3	46.4	71.2	76.4	74.0	63.7	88.8	79.0	113.9	91.5	134.5	115.7	62.9	51.4	47.0	51.2	62.7	49.6	47.8	50.1
PE_{lang} L	0.3B	448/14	78.3	82.8	89.3	65.2	75.9	78.5	78.8	64.4	89.6	81.3	117.8	94.7	138.1	120.7	71.6	56.5	47.0	57.2	68.0	59.8	52.3	54.7
AIMv2 3B [35]	2.7B	448/14	67.5	73.0	78.2	46.5	72.2	78.8	79.2	66.2	88.3	81.7	119.0	**95.8**	139.7	121.5	65.1	54.0	**49.6**	55.4	67.3	49.6	49.9	52.5
InternViT2.5 6B [16]	5.5B	448/14	67.4	74.6	74.3	47.6	72.9	75.9	71.3	64.8	87.7	79.7	110.4	85.3	132.5	113.5	56.8	52.0	46.0	49.6	65.0	50.6	49.6	51.3
PE_{core} G	1.9B	448/14	68.0	73.4	81.2	47.6	69.7	76.4	74.3	62.5	89.1	79.6	113.0	91.6	134.5	112.9	67.6	53.2	46.0	54.3	67.0	51.2	48.7	52.0
PE_{lang} G	1.7B*	448/14	**78.6**	**81.8**	**89.8**	**67.8**	**75.0**	**80.3**	**82.3**	**66.7**	**89.6**	**82.8**	**119.6**	95.2	**140.3**	**123.4**	**71.8**	**59.0**	**49.6**	**61.8**	**73.9**	**60.0**	**52.6**	**56.3**

Table 30 4+1 Tile Llama 8B MLLM Results. Llama 3.1-instruct 8B [80] is used as a language model. *PE_{lang} has 1.7B parameters since we discard the last 3 layers during language alignment. All MLLMs are trained with dynamic tiling for different image sizes and aspect ratio. We use up to 4 image tiles of 448×448 (or the corresponding resolution for each encoder). The image tiles follow after a *thumbnail* input, similar to prior work [75]. †Evaluation on an model that was interpolated without additional training (i.e., *zero-shot* resolution).

Model	Encoder Params	Resolution / Patch Size	Grounding: Avg. Ground.	RefCOCO *val* [54]	RefCOCO *testA* [54]	RefCOCO *testB* [54]	RefCOCO+ *val* [54]	RefCOCO+ *testA* [54]	RefCOCO+ *testB* [54]	RefCOCOg *val* [54]	RefCOCOg *test* [54]
256 Tokens per Image											
MetaCLIP-L [147]	0.3B	224/14	60.6	63.6	56.7	67.5	54.1	58.9	48.8	67.2	67.8
MetaCLIP-G [147]	1.8B	224/14	60.5	62.0	56.5	67.8	53.5	58.7	49.2	68.2	68.3
PE$_{\text{lang}}$ G†	1.7B*	224/14	65.7	67.7	64.4	70.9	58.3	62.0	56.6	73.2	74.4
576 Tokens per Image											
CLIP [103]	0.3B	336/14	65.0	66.7	61.4	71.6	57.6	62.5	54.5	73.2	72.8
AIMv2-L [35]	0.3B	336/14	63.3	65.4	61.6	69.6	55.0	60.0	52.0	71.1	71.5
AIMv2-L Dist. [35]	0.3B	336/14	62.6	64.8	61.0	69.4	54.4	59.0	51.3	70.8	70.0
SigLIP2-so [134]	0.4B	384/16	67.4	68.8	66.5	71.0	60.3	61.8	58.5	76.2	76.0
SigLIP2-g-opt [134]	1.1B	384/16	66.5	67.9	66.1	70.1	58.8	61.7	57.1	75.5	75.0
PE$_{\text{lang}}$ G†	1.7B*	336/14	68.9	69.8	67.5	73.2	61.5	64.0	60.8	77.3	77.7
1024 Tokens per Image											
InternViT2.5 L [16]	0.3B	448/14	66.9	69.3	66.7	72.6	58.3	63.1	57.2	74.2	74.0
SigLIP2-so [134]	0.4B	512/16	69.6	71.4	69.2	74.4	61.3	64.8	60.3	77.9	77.2
PE$_{\text{core}}$ L	0.3B	448/14	59.7	61.7	55.3	66.9	53.1	58.8	48.0	68.5	67.5
PE$_{\text{lang}}$ L	0.3B	448/14	70.5	71.8	**70.2**	73.0	63.7	66.1	62.7	78.8	78.9
DINOv2 [95]	1.1B	448/14	64.9	67.2	62.5	70.5	57.0	61.0	54.5	73.1	73.1
AIMv2 3B [35]	2.7B	448/14	36.1	37.6	34.1	40.7	32.7	36.2	32.0	36.9	38.6
InternViT2.5 6B [16]	5.5B	448/14	68.0	70.2	67.6	72.2	60.6	64.0	58.7	75.3	75.2
PE$_{\text{core}}$ G	1.9B	448/14	66.6	68.3	64.4	72.3	58.7	62.7	56.0	75.1	75.0
PE$_{\text{lang}}$ G	1.7B*	448/14	**71.3**	**71.9**	69.9	**75.1**	**64.2**	**67.3**	**63.0**	**79.4**	**79.2**

Table 31 Llama MLLM-Based Zeroshot RefCOCO. Llama 3.1-instruct 8B [80] is used for zeroshot RefCOCO/+/g grounding.

Model	Encoder Params	Resolution / Patch Size	Grounding: Avg. Ground.	RefCOCO *val* [54]	RefCOCO *testA* [54]	RefCOCO *testB* [54]	RefCOCO+ *val* [54]	RefCOCO+ *testA* [54]	RefCOCO+ *testB* [54]	RefCOCOg *val* [54]	RefCOCOg *test* [54]
576 Tokens per Image											
SigLIP2-so [134]	0.4B	384/16	70.0	73.6	73.0	74.3	60.9	62.7	59.9	78.4	77.2
SigLIP2-g-opt [134]	1.1B	384/16	69.9	73.3	72.4	73.6	60.5	62.3	60.7	78.4	78.2
PE$_{\text{lang}}$ G†	1.7B*	336/14	70.1	73.4	72.0	75.3	62.0	64.2	61.2	78.4	77.7
1024 Tokens per Image											
InternViT2.5 L [16]	0.3B	448/14	68.1	72.4	69.1	74.1	59.3	62.4	56.6	75.2	75.5
SigLIP2-so [134]	0.4B	512/16	70.5	74.1	73.7	74.4	61.7	62.9	61.0	78.6	77.9
PE$_{\text{core}}$L	0.3B	448/14	66.5	70.4	67.8	71.5	57.7	61.1	56.2	75.8	75.3
PE$_{\text{lang}}$L	0.3B	448/14	70.4	74.4	72.6	74.6	62.2	64.0	62.0	79.0	78.7
DINOv2 [95]	1.1B	448/14	69.3	73.4	71.1	73.9	60.0	63.9	59.0	76.4	76.7
AIMv2 3B [35]	2.7B	448/14	67.6	71.4	67.7	72.3	59.2	61.2	56.3	76.4	76.4
InternViT2.5 6B‡ [16]	5.5B	448/14	**72.8**	**77.7**	**76.5**	**77.1**	63.6	**66.0**	62.2	**80.0**	79.5
PE$_{\text{core}}$G	1.9B	448/14	70.5	74.0	71.8	75.8	61.5	64.8	60.1	78.5	77.3
PE$_{\text{lang}}$G	1.7B*	448/14	72.1	75.4	72.9	76.3	**64.2**	65.9	**62.9**	79.7	**79.7**

Table 32 Qwen MLLM-Based Zeroshot RefCOCO. QwenLM 2.5 7B [150] is used as a language model. All MLLMs report zeroshot results on RefCOCO/+/g datasets. ‡Trained with RefCOCO/+/g beforehand.

Model	Encoder Params	Resolution / Patch Size	Grounding: Avg. Ground.	RefCOCO *val* [54]	RefCOCO *testA* [54]	RefCOCO *testB* [54]	RefCOCO+ *val* [54]	RefCOCO+ *testA* [54]	RefCOCO+ *testB* [54]	RefCOCOg *val* [54]	RefCOCOg *test* [54]
256 Tokens per Tile											
MetaCLIP-L [147]	0.3B	224/14	67.1	69.3	65.0	73.2	60.5	64.9	56.5	74.3	73.4
MetaCLIP-G [147]	1.8B	224/14	66.0	67.9	63.2	71.9	59.2	62.9	55.8	73.8	73.1
PE$_{\text{lang}}$ G†	1.7B*	224/14	70.3	71.6	69.6	73.7	63.3	66.2	62.6	78.6	78.2
576 Tokens per Tile											
CLIP [103]	0.3B	336/14	68.5	70.7	66.6	74.1	61.1	65.9	58.1	76.0	75.1
AIMv2-L [35]	0.3B	336/14	66.6	68.4	65.5	71.4	59.3	63.4	56.5	74.2	74.2
SigLIP2-so [134]	0.4B	384/16	66.5	67.9	66.1	70.1	58.8	61.7	57.1	75.5	75.0
SigLIP2-g-opt [134]	1.1B	384/16	66.5	68.2	65.6	70.1	59.0	62.3	58.0	74.8	74.0
PE$_{\text{lang}}$ G†	1.7B*	336/14	71.9	73.6	71.5	74.9	64.8	67.3	63.9	80.4	80.6
1024 Tokens per Tile											
SigLIP2-so [134]	0.4B	512/16	67.8	69.2	67.8	71.2	59.9	62.5	59.0	76.9	76.0
PE$_{\text{core}}$L	0.3B	448/14	62.9	65.3	59.9	69.2	56.6	62.2	52.0	70.1	70.0
PE$_{\text{lang}}$L	0.3B	448/14	71.6	**73.0**	**70.8**	74.3	**65.2**	**67.2**	62.9	79.7	79.7
AIMv2 3B [35]	2.7B	448/14	65.1	66.9	62.9	71.1	58.1	62.4	55.6	71.8	72.2
InternViT2.5 6B‡ [16]	5.5B	448/14	56.8	61.0	56.4	65.8	51.0	57.0	46.1	58.0	58.9
PE$_{\text{core}}$G	1.9B	448/14	67.6	69.2	65.8	72.4	59.9	64.1	58.3	75.1	75.6
PE$_{\text{lang}}$G	1.7B*	448/14	**71.8**	72.6	70.7	**74.6**	64.8	66.6	**64.6**	**80.4**	**80.3**

Table 33 4+1 Tile Llama 8B MLLM-Based Zeroshot RefCOCO. Llama 3.1-instruct 8B [80] is used as a language model. All trained with dynamic tiling for different image sizes and aspect ratio. We use up to 4 image tiles of the encoder's native resolution, with a *thumbnail* image in front, similar to prior work [75]. ‡Trained with RefCOCO/+/g beforehand.

C.5 $\text{PE}_{\text{spatial}}$: Additional Qualitative Results

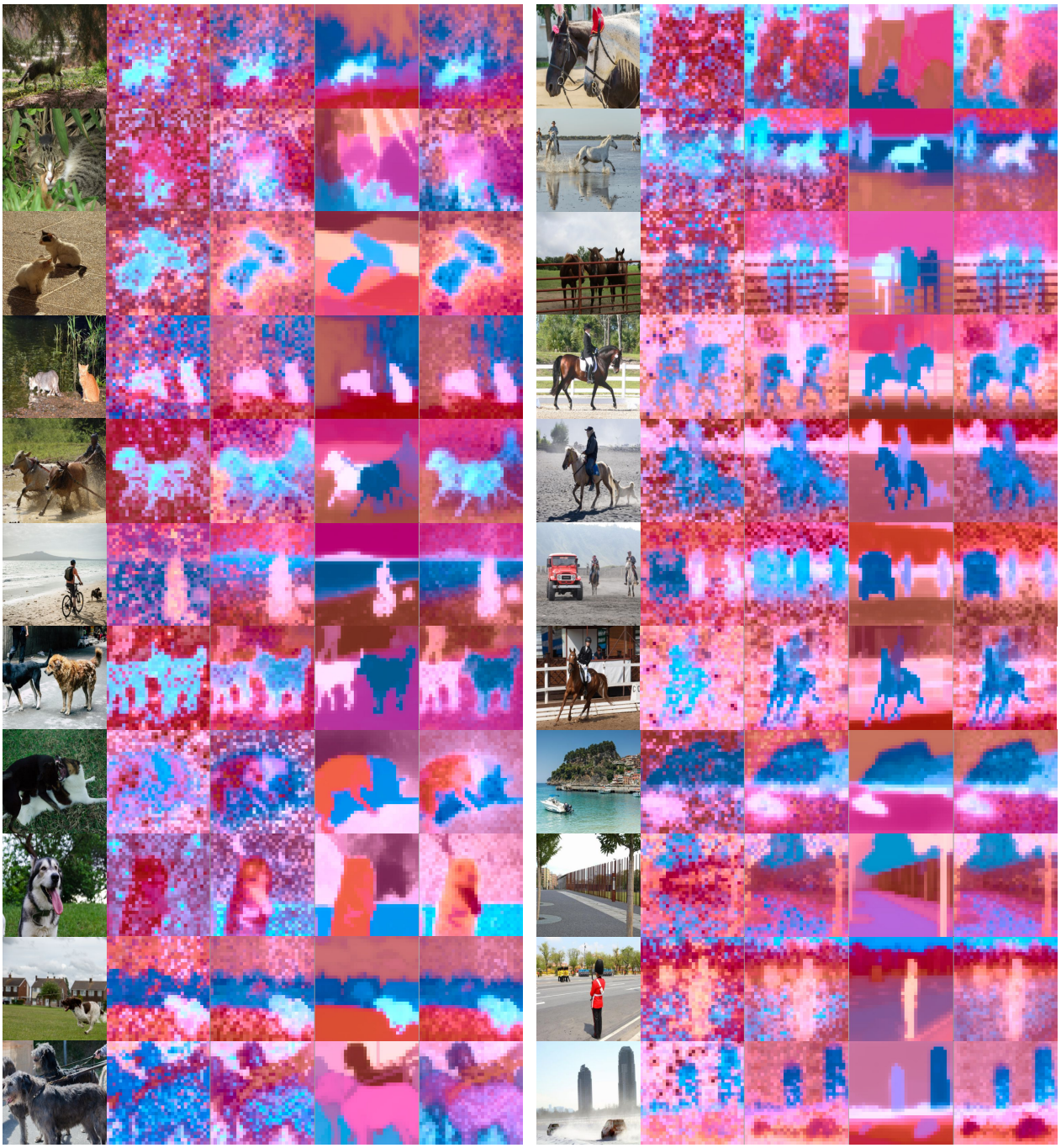


Figure 20 More Visualizations of the feature space following Fig. 17. After the image itself, column 1 is $\text{PE}_{\text{core}}\text{G}$ last layer features, column 2 is $\text{PE}_{\text{core}}\text{G}$ aligned to its own layer 41, column 3 is $\text{PE}_{\text{core}}\text{G}$ aligned to SAM 2.1-L [108] mask logits, and column 4 is $\text{PE}_{\text{core}}\text{G}$ aligned to both, denoted $\text{PE}_{\text{spatial}}\text{G}$. See §B.3.2 for visualization method.

References

[1] Harsh Agrawal, Karan Desai, Yufei Wang, Xinlei Chen, Rishabh Jain, Mark Johnson, Dhruv Batra, Devi Parikh, Stefan Lee, and Peter Anderson. Nocaps: Novel object captioning at scale. In *ICCV*, 2019. 14, 15, 16, 32

[2] Pravesh Agrawal, Szymon Antoniak, Emma Bou Hanna, Baptiste Bout, Devendra Chaplot, Jessica Chudnovsky, Diogo Costa, Baudouin De Monicault, Saurabh Garg, Theophile Gervet, Soham Ghosh, Amélie Héliou, Paul Jacob, Albert Q. Jiang, Kartik Khandelwal, Timothée Lacroix, Guillaume Lample, Diego Las Casas, Thibaut Lavril, Teven Le Scao, Andy Lo, William Marshall, Louis Martin, Arthur Mensch, Pavankumar Muddireddy, Valera Nemychnikova, Marie Pellat, Patrick Von Platen, Nikhil Raghuraman, Baptiste Rozière, Alexandre Sablayrolles, Lucile Saulnier, Romain Sauvestre, Wendy Shang, Roman Soletskyi, Lawrence Stewart, Pierre Stock, Joachim Studnia, Sandeep Subramanian, Sagar Vaze, Thomas Wang, and Sophia Yang. Pixtral 12b. *arXiv:2410.07073*, 2024. 20

[3] Jinze Bai, Shuai Bai, Shusheng Yang, Shijie Wang, Sinan Tan, Peng Wang, Junyang Lin, Chang Zhou, and Jingren Zhou. Qwen-VL: A versatile vision-language model for understanding, localization, text reading, and beyond. *arXiv:2308.12966*, 2023. 20

[4] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. ObjectNet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. In *NeurIPS*, 2019. 3, 4, 6, 8, 9, 10, 30, 31, 32

[5] Lucas Beyer, Andreas Steiner, André Susano Pinto, Alexander Kolesnikov, Xiao Wang, Daniel Salz, Maxim Neumann, Ibrahim Alabdulmohsin, Michael Tschannen, Emanuele Bugliarello, Thomas Unterthiner, Daniel Keysers, Skanda Koppula, Fangyu Liu, Adam Grycner, Alexey A. Gritsenko, Neil Houlsby, Manoj Kumar, Keran Rong, Julian Eisenschlos, Rishabh Kabra, Matthias Bauer, Matko Bosnjak, Xi Chen, Matthias Minderer, Paul Voigtlaender, Ioana Bica, Ivana Balazevic, Joan Puigcerver, Pinelopi Papalampidi, Olivier J. Hénaff, Xi Xiong, Radu Soricut, Jeremiah Harmsen, and Xiaohua Zhai. PaliGemma: A versatile 3b VLM for transfer. *arXiv:2407.07726*, 2024. 20

[6] Navaneeth Bodla, Bharat Singh, Rama Chellappa, and Larry S Davis. Soft-NMS–Improving object detection with one line of code. In *ICCV*, 2017. 30

[7] Daniel Bolya, Chaitanya Ryali, Judy Hoffman, and Christoph Feichtenhofer. Window attention is bugged: how not to interpolate position embeddings. In *ICLR*, 2023. 11, 29

[8] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101 – Mining discriminative components with random forests. In *ECCV*, 2014. 9

[9] Gary Bradski. The OpenCV library. *Dr. Dobb's Journal: Software Tools for the Professional Programmer*, 2000. 22

[10] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018. 19

[11] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 19

[12] Wenhao Chai, Enxin Song, Yilun Du, Chenlin Meng, Vashisht Madhavan, Omer Bar-Tal, Jeng-Neng Hwang, Saining Xie, and Christopher D. Manning. AuroraCap: Efficient, performant video detailed captioning and a new benchmark. In *ICLR*, 2025. 5

[13] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *CVPR*, 2019. 19

[14] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *ICML*, 2020. 20

[15] Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Nan Ding, Keran Rong, Hassan Akbari, Gaurav Mishra, Linting Xue, Ashish Thapliyal, James Bradbury, Weicheng Kuo, Mojtaba Seyedhosseini, Chao Jia, Burcu Karagol Ayan, Carlos Riquelme, Andreas Steiner, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. Pali: A jointly-scaled multilingual language-image model. In *ICLR*, 2023. 8, 9

[16] Zhe Chen, Weiyun Wang, Yue Cao, Yangzhou Liu, Zhangwei Gao, Erfei Cui, Jinguo Zhu, Shenglong Ye, Hao Tian, Zhaoyang Liu, Lixin Gu, Xuehui Wang, Qingyun Li, Yimin Ren, Zixuan Chen, Jiapeng Luo, Jiahao Wang, Tan Jiang, Bo Wang, Conghui He, Botian Shi, Xingcheng Zhang, Han Lv, Yi Wang, Wenqi Shao, Pei Chu, Zhongying Tu, Tong He, Zhiyong Wu, Huipeng Deng, Jiaye Ge, Kai Chen, Kaipeng Zhang, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. Expanding performance boundaries of open-source multimodal models with model, data, and test-time scaling. *arXiv:2412.05271*, 2024. 11, 15, 16, 20, 32, 33

[17] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. InternVL: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024. 1, 6, 7, 9, 10, 20, 26

[18] Gong Cheng, Junwei Han, and Xiaoqiang Lu. Remote sensing image scene classification: Benchmark and state of the art. *Proceedings of the IEEE*, 2017. 9

[19] Jang Hyun Cho, Andrea Madotto, Effrosyni Mavroudi, Triantafyllos Afouras, Tushar Nagarajan, Muhammad Maaz, Yale Song, Tengyu Ma, Shuming Hu, Hanoona Rasheed, Peize Sun, Po-Yao Huang, Daniel Bolya, Suyog Jain, Miguel Martin, Huiyu Wang, Nikhila Ravi, Shashank Jain, Temmy Stark, Shane Moon, Babak Damavandi, Vivian Lee, Andrew Westbury, Salman Khan, Philipp Krähenbühl, Piotr Dollár, Lorenzo Torresani, Kristen Grauman, and Christoph Feichtenhofer. Perceptionlm: Open-access data and models for detailed visual understanding. *arXiv*, 2025. 2, 5, 11, 14, 15, 16, 21

[20] Seokju Cho, Heeseong Shin, Sunghwan Hong, Anurag Arnab, Paul Hongsuck Seo, and Seungryong Kim. CAT-Seg: Cost aggregation for open-vocabulary semantic segmentation. In *CVPR*, 2024. 20

[21] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. In *ICLR*, 2024. 12, 17

[22] Mostafa Dehghani, Josip Djolonga, Basil Mustafa, Piotr Padlewski, Jonathan Heek, Justin Gilmer, Andreas Steiner, Mathilde Caron, Robert Geirhos, Ibrahim Alabdulmohsin, Rodolphe Jenatton, Lucas Beyer, Michael Tschannen, Anurag Arnab, Xiao Wang, Carlos Riquelme, Matthias Minderer, Joan Puigcerver, Utku Evci, Manoj Kumar, Sjoerd van Steenkiste, Gamaleldin F. Elsayed, Aravindh Mahendran, Fisher Yu, Avital Oliver, Fantine Huot, Jasmijn Bastings, Mark Patrick Collier, Alexey Gritsenko, Vighnesh Birodkar, Cristina Vasconcelos, Yi Tay, Thomas Mensink, Alexander Kolesnikov, Filip Pavetić, Dustin Tran, Thomas Kipf, Mario Lučić, Xiaohua Zhai, Daniel Keysers, Jeremiah Harmsen, and Neil Houlsby. Scaling vision transformers to 22 billion parameters. In *ICML*, 2023. 1, 9

[23] Matt Deitke, Christopher Clark, Sangho Lee, Rohun Tripathi, Yue Yang, Jae Sung Park, Mohammadreza Salehi, Niklas Muennighoff, Kyle Lo, Luca Soldaini, Jiasen Lu, Taira Anderson, Erin Bransom, Kiana Ehsani, Huong Ngo, YenSung Chen, Ajay Patel, Mark Yatskar, Chris Callison-Burch, Andrew Head, Rose Hendrix, Favyen Bastani, Eli VanderBilt, Nathan Lambert, Yvonne Chou, Arnavi Chheda, Jenna Sparks, Sam Skjonsberg, Michael Schmitz, Aaron Sarnat, Byron Bischoff, Pete Walsh, Chris Newell, Piper Wolters, Tanmay Gupta, Kuo-Hao Zeng, Jon Borchardt, Dirk Groeneveld, Crystal Nam, Sophie Lebrecht, Caitlin Wittlif, Carissa Schoenick, Oscar Michel, Ranjay Krishna, Luca Weihs, Noah A. Smith, Hannaneh Hajishirzi, Ross Girshick, Ali Farhadi, and Aniruddha Kembhavi. Molmo and pixmo: Open weights and open data for state-of-the-art multimodal models. *arXiv:2409.17146*, 2024. 16

[24] 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, 3, 6, 8, 9, 10, 30, 31, 32

[25] Karan Desai and Justin Johnson. VirTex: Learning visual representations from textual annotations. In *CVPR*, 2021. 20

[26] Jian Ding, Nan Xue, Gui-Song Xia, and Dengxin Dai. Decoupling zero-shot semantic segmentation. In *CVPR*, 2022. 20

[27] 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. In *ICLR*, 2020. 1, 8, 9

[28] 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. 20

[29] David Fan, Shengbang Tong, Jiachen Zhu, Koustuv Sinha, Zhuang Liu, Xinlei Chen, Michael Rabbat, Nicolas Ballas, Yann LeCun, Amir Bar, and Saining Xie. Scaling language-free visual representation learning. *arXiv:2504.01017*, 2025. 12, 13

[30] Lijie Fan, Dilip Krishnan, Phillip Isola, Dina Katabi, and Yonglong Tian. Improving CLIP training with language rewrites. In *NeurIPS*, 2023. 20

[31] Alex Fang, Albin Madappally Jose, Amit Jain, Ludwig Schmidt, Alexander Toshev, and Vaishaal Shankar. Data filtering networks. In *ICLR*, 2024. 1, 3, 9, 16, 20, 26

[32] Yuxin Fang, Wen Wang, Binhui Xie, Quan Sun, Ledell Wu, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA: Exploring the limits of masked visual representation learning at scale. In *CVPR*, 2023. 1

[33] Yuxin Fang, Quan Sun, Xinggang Wang, Tiejun Huang, Xinlong Wang, and Yue Cao. EVA-02: A visual representation for neon genesis. *Image and Vision Computing*, 2024. 1, 19

[34] Christoph Feichtenhofer. X3D: Expanding architectures for efficient video recognition. In *CVPR*, 2020. 4

[35] Enrico Fini, Mustafa Shukor, Xiujun Li, Philipp Dufter, Michal Klein, David Haldimann, Sai Aitharaju, Victor Guilherme Turrisi da Costa, Louis Béthune, Zhe Gan, Alexander T. Toshev, Marcin Eichner, Moin Nabi, Yinfei Yang, Joshua M. Susskind, and Alaaeldin El-Nouby. Multimodal autoregressive pre-training of large vision encoders. In *CVPR*, 2025. 1, 2, 10, 11, 15, 16, 19, 20, 32, 33

[36] Chaoyou Fu, Yuhan Dai, Yongdong Luo, Lei Li, Shuhuai Ren, Renrui Zhang, Zihan Wang, Chenyu Zhou, Yunhang Shen, Mengdan Zhang, Peixian Chen, Yanwei Li, Shaohui Lin, Sirui Zhao, Ke Li, Tong Xu, Xiawu Zheng, Enhong Chen, Rongrong Ji, and Xing Sun. Video-MME: The first-ever comprehensive evaluation benchmark of multi-modal llms in video analysis. *arXiv:2405.21075*, 2024. 14, 15, 16, 32

[37] Samir Yitzhak Gadre, Gabriel Ilharco, Alex Fang, Jonathan Hayase, Georgios Smyrnis, Thao Nguyen, Ryan Marten, Mitchell Wortsman, Dhruba Ghosh, Jieyu Zhang, Eyal Orgad, Rahim Entezari, Giannis Daras, Sarah Pratt, Vivek Ramanujan, Yonatan Bitton, Kalyani Marathe, Stephen Mussmann, Richard Vencu, Mehdi Cherti, Ranjay Krishna, Pang Wei Koh, Olga Saukh, Alexander Ratner, Shuran Song, Hannaneh Hajishirzi, Ali Farhadi, Romain Beaumont, Sewoong Oh, Alex Dimakis, Jenia Jitsev, Yair Carmon, Vaishaal Shankar, and Ludwig Schmidt. DataComp: In search of the next generation of multimodal datasets. In *NeurIPS*, 2023. 10, 20

[38] Yash Goyal, Tejas Khot, Douglas Summers-Stay, Dhruv Batra, and Devi Parikh. Making the v in VQA matter: Elevating the role of image understanding in visual question answering. In *CVPR*, 2017. 14, 15, 16, 32

[39] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 19

[40] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[41] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 11, 12, 19, 29

[42] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022. 1, 19

[43] Greg Heinrich, Mike Ranzinger, Hongxu, Yin, Yao Lu, Jan Kautz, Andrew Tao, Bryan Catanzaro, and Pavlo Molchanov. RADIOv2.5: Improved baselines for agglomerative vision foundation models. In *CVPR*, 2025. 1, 10, 18

[44] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, Dawn Song, Jacob Steinhardt, and Justin Gilmer. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *ICCV*, 2021. 3, 8, 9, 30, 31

[45] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *CVPR*, 2021. 3, 4, 8, 9, 30, 31, 32

[46] Byeongho Heo, Song Park, Dongyoon Han, and Sangdoo Yun. Rotary position embedding for vision transformer. In *ECCV*, 2024. 20

[47] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. In *NeurIPS Deep Learning Workshop*, 2015. 8

[48] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, 2016. 14, 17

[49] 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. 3, 20

[50] Allan Jabri, Andrew Owens, and Alexei Efros. Space-time correspondence as a contrastive random walk. In *NeurIPS*, 2020. 11, 19, 29

[51] Yunseok Jang, Yale Song, Youngjae Yu, Youngjin Kim, and Gunhee Kim. TGIF-QA: Toward spatio-temporal reasoning in visual question answering. In *CVPR*, 2017. 14, 15, 16, 32

[52] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *ICML*, 2021. 1, 20

[53] Will Kay, Joao Carreira, Karen Simonyan, Brian Zhang, Chloe Hillier, Sudheendra Vijayanarasimhan, Fabio Viola, Tim Green, Trevor Back, Paul Natsev, Mustafa Suleyman, and Andrew Zisserman. The kinetics human action video dataset. *arXiv:1705.06950*, 2017. 6, 9, 31, 32

[54] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 14, 15, 16, 32, 33

[55] Aniruddha Kembhavi, Mike Salvato, Eric Kolve, Minjoon Seo, Hannaneh Hajishirzi, and Ali Farhadi. A diagram is worth a dozen images. In *ECCV*, 2016. 14, 15, 16, 32

[56] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C. Berg, Wan-Yen Lo, Piotr Dollár, and Ross Girshick. Segment anything. In *ICCV*, 2023. 5, 18

[57] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *ICCV Workshop*, 2013. 9

[58] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalantidis, Li-Jia Li, David A. Shamma, Michael S. Bernstein, and Fei-Fei Li. Visual genome: Connecting language and vision using crowdsourced dense image annotations. *IJCV*, 2017. 27, 32

[59] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[60] Hildegard Kuehne, Hueihan Jhuang, Estíbaliz Garrote, Tomaso Poggio, and Thomas Serre. HMDB: a large video database for human motion recognition. In *ICCV*, 2011. 9, 31, 32

[61] Weicheng Kuo, Yin Cui, Xiuye Gu, A. J. Piergiovanni, and Anelia Angelova. F-VLM: open-vocabulary object detection upon frozen vision and language models. In *ICLR*, 2023. 20

[62] Zhengfeng Lai, Haotian Zhang, Bowen Zhang, Wentao Wu, Haoping Bai, Aleksei Timofeev, Xianzhi Du, Zhe Gan, Jiulong Shan, Chen-Nee Chuah, Yinfei Yang, and Meng Cao. VeCLIP: Improving CLIP training via visual-enriched captions. In *ECCV*, 2024. 5, 20

[63] Hugo Laurençon, Léo Tronchon, Matthieu Cord, and Victor Sanh. What matters when building vision-language models? In *NeurIPS*, 2024. 27

[64] Bo Li, Yuanhan Zhang, Dong Guo, Renrui Zhang, Feng Li, Hao Zhang, Kaichen Zhang, Yanwei Li, Ziwei Liu, and Chunyuan Li. LLaVA-OneVision: Easy visual task transfer. *TMLR*, 2025. 16, 20, 22

[65] Kunchang Li, Yali Wang, Yizhuo Li, Yi Wang, Yinan He, Limin Wang, and Yu Qiao. Unmasked teacher: Towards training-efficient video foundation models. In *ICCV*, 2023. 9

[66] Kunchang Li, Yali Wang, Yinan He, Yizhuo Li, Yi Wang, Yi Liu, Zun Wang, Jilan Xu, Guo Chen, Ping Luo, Limin Wang, and Yu Qiao. MVBench: A comprehensive multi-modal video understanding benchmark. In *CVPR*, 2024. 14, 15, 16, 32

[67] Liunian Harold Li, Pengchuan Zhang, Haotian Zhang, Jianwei Yang, Chunyuan Li, Yiwu Zhong, Lijuan Wang, Lu Yuan, Lei Zhang, Jenq-Neng Hwang, Kai-Wei Chang, and Jianfeng Gao. Grounded language-image pre-training. In *CVPR*, 2022. 1

[68] Xianhang Li, Zeyu Wang, and Cihang Xie. An inverse scaling law for CLIP training. In *NeurIPS*, 2023. 3

[69] Xianhang Li, Zeyu Wang, and Cihang Xie. CLIPA-v2: Scaling CLIP training with 81.1% zero-shot imagenet accuracy within a $10,000 budget; an extra $4,000 unlocks 81.8% accuracy. *arXiv:2306.15658*, 2023. 3, 20

[70] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *ECCV*, 2022. 11, 19, 29

[71] Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji-Rong Wen. Evaluating object hallucination in large vision-language models. In *EMNLP*, 2023. 14, 15, 16, 32

[72] Yanghao Li, Haoqi Fan, Ronghang Hu, Christoph Feichtenhofer, and Kaiming He. Scaling language-image pre-training via masking. In *CVPR*, 2023. 20

[73] Zhenyu Li, Xuyang Wang, Xianming Liu, and Junjun Jiang. Binsformer: Revisiting adaptive bins for monocular depth estimation. *TIP*, 2024. 29

[74] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 2, 6, 9, 12, 14, 15, 16, 19, 27, 31, 32

[75] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. LLaVA-NeXT: Improved reasoning, ocr, and world knowledge, 2024. 32, 33

[76] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. *NeurIPS*, 2024. 20, 23

[77] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*, 2021. 3, 19

[78] Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution. In *CVPR*, 2022. 19

[79] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A ConvNet for the 2020s. In *CVPR*, 2022. 1

[80] AI @ Meta Llama Team. The llama 3 herd of models. *arXiv:2407.21783*, 2024. 5, 14, 15, 16, 20, 32, 33

[81] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *ICLR*, 2019. 3, 29

[82] Huaishao Luo, Lei Ji, Ming Zhong, Yang Chen, Wen Lei, Nan Duan, and Tianrui Li. CLIP4Clip: An empirical study of clip for end to end video clip retrieval. *Neurocomputing*, 2021. 6, 9

[83] Nanye Ma, Mark Goldstein, Michael S Albergo, Nicholas M Boffi, Eric Vanden-Eijnden, and Saining Xie. SiT: Exploring flow and diffusion-based generative models with scalable interpolant transformers. In *ECCV*, 2024. 20

[84] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. Video-ChatGPT: Towards detailed video understanding via large vision and language models. In *ACL*, 2024. 5

[85] Muhammad Maaz, Hanoona Rasheed, Salman Khan, and Fahad Shahbaz Khan. VideoGPT+: Integrating image and video encoders for enhanced video understanding. *arXiv:2406.09418*, 2024. 5

[86] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arxiv:1306.5151*, 2013. 9

[87] Karttikeya Mangalam, Raiymbek Akshulakov, and Jitendra Malik. Egoschema: A diagnostic benchmark for very long-form video language understanding. *NeurIPS*, 2024. 14, 15, 16, 32

[88] Minesh Mathew, Dimosthenis Karatzas, and CV Jawahar. DocVQA: A dataset for vqa on document images. In *WACV*, 2021. 14, 15, 16, 32

[89] Minesh Mathew, Viraj Bagal, Rubèn Tito, Dimosthenis Karatzas, Ernest Valveny, and CV Jawahar. Infographicvqa. In *WACV*, 2022. 14, 15, 16, 32

[90] Brandon McKinzie, Zhe Gan, Jean-Philippe Fauconnier, Sam Dodge, Bowen Zhang, Philipp Dufter, Dhruti Shah, Xianzhi Du, Futang Peng, Floris Weers, Anton Belyi, Haotian Zhang, Karanjeet Singh, Doug Kang, Ankur Jain, Hongyu Hè, Max Schwarzer, Tom Gunter, Xiang Kong, Aonan Zhang, Jianyu Wang, Chong Wang, Nan Du, Tao Lei, Sam Wiseman, Guoli Yin, Mark Lee, Zirui Wang, Ruoming Pang, Peter Grasch, Alexander Toshev, and Yinfei Yang. MM1: methods, analysis and insights from multimodal LLM pre-training. In *ECCV*, 2024. 20

[91] Matthias Minderer, Alexey A. Gritsenko, Austin Stone, Maxim Neumann, Dirk Weissenborn, Alexey Dosovitskiy, Aravindh Mahendran, Anurag Arnab, Mostafa Dehghani, Zhuoran Shen, Xiao Wang, Xiaohua Zhai, Thomas Kipf, and Neil Houlsby. Simple open-vocabulary object detection with vision transformers. In *ECCV*, 2022. 1, 20

[92] Matthias Minderer, Alexey Gritsenko, and Neil Houlsby. Scaling open-vocabulary object detection. In *NeurIPS*, 2023. 20

[93] Thao Nguyen, Samir Yitzhak Gadre, Gabriel Ilharco, Sewoong Oh, and Ludwig Schmidt. Improving multimodal datasets with image captioning. In *NeurIPS*, 2023. 5, 20

[94] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *ICVGIP*, 2008. 9

[95] 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, Hervé Jégou, Julien Mairal, Patrick Labatut, Armand Joulin, and Piotr Bojanowski. DINOv2: Learning robust visual features without supervision. *TMLR*, 2024. 1, 2, 10, 11, 15, 16, 18, 19, 20, 22, 29, 33

[96] Jeffrey Ouyang-Zhang, Jang Hyun Cho, Xingyi Zhou, and Philipp Krähenbühl. NMSstrikes back. *arXiv:2212.06137*, 2022. 19

[97] Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *CVPR*, 2012. 9

[98] Zhiliang Peng, Wenhui Wang, Li Dong, Yaru Hao, Shaohan Huang, Shuming Ma, and Furu Wei. Kosmos-2: Grounding multimodal large language models to the world. *arXiv:2306.14824*, 2023. 20

[99] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Kenji Kawaguchi, Hanxiao Liu, Adams Wei Yu, Jiahui Yu, Yi-Ting Chen, Minh-Thang Luong, Yonghui Wu, Mingxing Tan, and Quoc V. Le. Combined scaling for zero-shot transfer learning. *Neurocomputing*, 2023. 1, 9, 20

[100] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*, 2015. 27, 32

[101] Jordi Pont-Tuset, Federico Perazzi, Sergi Caelles, Pablo Arbeláez, Alex Sorkine-Hornung, and Luc Van Gool. The 2017 DAVIS challenge on video object segmentation. *arXiv:1704.00675*, 2017. 19, 29

[102] Viorica Pătrăucean, Lucas Smaira, Ankush Gupta, Adrià Recasens Continente, Larisa Markeeva, Dylan Banarse, Skanda Koppula, Joseph Heyward, Mateusz Malinowski, Yi Yang, Carl Doersch, Tatiana Matejovicova, Yury Sulsky, Antoine Miech, Alex Frechette, Hanna Klimczak, Raphael Koster, Junlin Zhang, Stephanie Winkler, Yusuf Aytar, Simon Osindero, Dima Damen, Andrew Zisserman, and João Carreira. Perception test: A diagnostic benchmark for multimodal video models. In *NeurIPS*, 2024. 14, 15, 16, 32

[103] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, 2021. 1, 3, 8, 9, 15, 16, 19, 20, 31, 32, 33

[104] Jathushan Rajasegaran, Ilija Radosavovic, Rahul Ravishankar, Yossi Gandelsman, Christoph Feichtenhofer, and Jitendra Malik. An empirical study of autoregressive pre-training from videos. *arXiv:2501.05453*, 2025. 19, 20, 29

[105] Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, and Mark Chen. Hierarchical text-conditional image generation with CLIP latents. *arXiv:2204.06125*, 2022. 1

[106] René Ranftl, Alexey Bochkovskiy, and Vladlen Koltun. Vision transformers for dense prediction. In *ICCV*, 2021. 11, 19, 29

[107] Mike Ranzinger, Greg Heinrich, Jan Kautz, and Pavlo Molchanov. AM-RADIO: Agglomerative vision foundation model–reduce all domains into one. In *CVPR*, 2024. 1, 18, 21

[108] Nikhila Ravi, Valentin Gabeur, Yuan-Ting Hu, Ronghang Hu, Chaitanya Ryali, Tengyu Ma, Haitham Khedr, Roman Rädle, Chloe Rolland, Laura Gustafson, Eric Mintun, Junting Pan, Kalyan Vasudev Alwala, Nicolas Carion, Chao-Yuan Wu, Ross Girshick, Piotr Dollár, and Christoph Feichtenhofer. SAM 2: Segment anything in images and videos. In *ICLR*, 2024. 2, 5, 17, 18, 34

[109] Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *ICML*, 2019. 3, 6, 8, 9, 30, 31, 32

[110] William A. Gaviria Rojas, Sudnya Diamos, Keertan Ranjan Kini, David Kanter, Vijay Janapa Reddi, and Cody Coleman. The dollar street dataset: images representing the geographic and socioeconomic diversity of the world. In *NeurIPS Datasets and Benchmarks*, 2022. 10

[111] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *CVPR*, 2022. 1

[112] Mert Bulent Sariyildiz, Julien Perez, and Diane Larlus. Learning visual representations with caption annotations. In *ECCV*, 2020. 20

[113] Mert Bulent Sariyildiz, Philippe Weinzaepfel, Thomas Lucas, Diane Larlus, and Yannis Kalantidis. UNIC: Universal classification models via multi-teacher distillation. In *ECCV*, 2024. 18

[114] Christoph Schuhmann, Romain Beaumont, Richard Vencu, Cade W Gordon, Ross Wightman, Mehdi Cherti, Theo Coombes, Aarush Katta, Clayton Mullis, Mitchell Wortsman, Patrick Schramowski, Srivatsa R Kundurthy, Katherine Crowson, Ludwig Schmidt, Robert Kaczmarczyk, and Jenia Jitsev. LAION-5b: An open large-scale dataset for training next generation image-text models. In *NeurIPS Datasets and Benchmarks*, 2022. 20

[115] Dustin Schwenk, Apoorv Khandelwal, Christopher Clark, Kenneth Marino, and Roozbeh Mottaghi. A-OKVQA: A benchmark for visual question answering using world knowledge. In *ECCV*, 2022. 14, 15, 16, 32

[116] Jinghuan Shang, Karl Schmeckpeper, Brandon B May, Maria Vittoria Minniti, Tarik Kelestemur, David Watkins, and Laura Herlant. Theia: Distilling diverse vision foundation models for robot learning. In *CoRL*, 2024. 18

[117] Shuai Shao, Zeming Li, Tianyuan Zhang, Chao Peng, Gang Yu, Xiangyu Zhang, Jing Li, and Jian Sun. Objects365: A large-scale, high-quality dataset for object detection. In *ICCV*, 2019. 19

[118] Oleksii Sidorov, Ronghang Hu, Marcus Rohrbach, and Amanpreet Singh. Textcaps: a dataset for image captioning with reading comprehension. In *ECCV*, 2020. 10

[119] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 19, 29

[120] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[121] Amanpreet Singh, Vivek Natarjan, Meet Shah, Yu Jiang, Xinlei Chen, Devi Parikh, and Marcus Rohrbach. Towards VQA models that can read. In *CVPR*, 2019. 14, 15, 16, 32

[122] Khurram Soomro, Amir Roshan Zamir, and Mubarak Shah. UCF101: A dataset of 101 human actions classes from videos in the wild. *arXiv:1212.0402*, 2012. 9, 31, 32

[123] Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. 4, 20, 25

[124] Lin Sun, Jiale Cao, Jin Xie, Xiaoheng Jiang, and Yanwei Pang. CLIPer: Hierarchically improving spatial representation of CLIP for open-vocabulary semantic segmentation. *arXiv:2411.13836*, 2024. 20

[125] Quan Sun, Yuxin Fang, Ledell Wu, Xinlong Wang, and Yue Cao. EVA-CLIP: Improved training techniques for clip at scale. *arXiv:2303.15389*, 2023. 20

[126] Quan Sun, Jinsheng Wang, Qiying Yu, Yufeng Cui, Fan Zhang, Xiaosong Zhang, and Xinlong Wang. EVA-CLIP-18B: Scaling clip to 18 billion parameters. *arXiv:2402.04252*, 2024. 1, 9, 10, 20, 26

[127] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *ICML*, 2019. 1, 3, 4

[128] Gemma Team. Gemma 3 technical report. *arXiv:2503.19786*, 2025. 16, 20

[129] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. YFCC100M: The new data in multimedia research. *Communications of the ACM*, 2016. 9

[130] Shengbang Tong, Ellis Brown, Penghao Wu, Sanghyun Woo, Manoj Middepogu, Sai Charitha Akula, Jihan Yang, Shusheng Yang, Adithya Iyer, Xichen Pan, Ziteng Wang, Rob Fergus, Yann LeCun, and Saining Xie. Cambrian-1: A fully open, vision-centric exploration of multimodal llms. In *NeurIPS*, 2024. 11, 20

[131] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *ICCV*, 2021. 14, 17

[132] Hugo Touvron, Matthieu Cord, and Hervé Jégou. DeiT III: Revenge of the ViT. In *ECCV*, 2022. 3

[133] Michael Tschannen, Manoj Kumar, Andreas Steiner, Xiaohua Zhai, Neil Houlsby, and Lucas Beyer. Image captioners are scalable vision learners too. In *NeurIPS*, 2023. 1, 20

[134] Michael Tschannen, Alexey Gritsenko, Xiao Wang, Muhammad Ferjad Naeem, Ibrahim Alabdulmohsin, Nikhil Parthasarathy, Talfan Evans, Lucas Beyer, Ye Xia, Basil Mustafa, Olivier Hénaff, Jeremiah Harmsen, Andreas Steiner, and Xiaohua Zhai. SigLIP 2: Multilingual vision-language encoders with improved semantic understanding, localization, and dense features. *arXiv:2502.14786*, 2025. 2, 7, 8, 9, 10, 15, 16, 18, 19, 26, 32, 33

[135] Jack Urbanek, Florian Bordes, Pietro Astolfi, Mary Williamson, Vasu Sharma, and Adriana Romero-Soriano. A picture is worth more than 77 text tokens: Evaluating CLIP-style models on dense captions. In *CVPR*, 2024. 27, 32

[136] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *CVPR*, 2018. 10

[137] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 25

[138] Haohan Wang, Songwei Ge, Zachary Lipton, and Eric P Xing. Learning robust global representations by penalizing local predictive power. In *NeurIPS*, 2019. 3, 8, 9, 30, 31

[139] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Yang Fan, Kai Dang, Mengfei Du, Xuancheng Ren, Rui Men, Dayiheng Liu, Chang Zhou, Jingren Zhou, and Junyang Lin. Qwen2-VL: Enhancing vision-language model's perception of the world at any resolution. *arXiv:2409.12191*, 2024. 16, 20

[140] Wenhai Wang, Jifeng Dai, Zhe Chen, Zhenhang Huang, Zhiqi Li, Xizhou Zhu, Xiaowei Hu, Tong Lu, Lewei Lu, Hongsheng Li, Xiaogang Wang, and Yu Qiao. InternImage: Exploring large-scale vision foundation models with deformable convolutions. In *CVPR*, 2023. 19

[141] Yi Wang, Kunchang Li, Xinhao Li, Jiashuo Yu, Yinan He, Guo Chen, Baoqi Pei, Rongkun Zheng, Zun Wang, Yansong Shi, Tianxiang Jiang, Songze Li, Jilan Xu, Hongjie Zhang, Yifei Huang, Yu Qiao, Yali Wang, and Limin Wang. InternVideo2: Scaling foundation models for multimodal video understanding. In *ECCV*, 2024. 2, 9

[142] Chen Wei, Haoqi Fan, Saining Xie, Chao-Yuan Wu, Alan L. Yuille, and Christoph Feichtenhofer. Masked feature prediction for self-supervised visual pre-training. In *CVPR*, 2022. 4, 17

[143] Bo Wu, Shoubin Yu, Zhenfang Chen, Joshua B Tenenbaum, and Chuang Gan. STAR: A benchmark for situated reasoning in real-world videos. In *NeurIPS*, 2021. 14, 15, 16, 32

[144] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019. 29

[145] Jianxiong Xiao, Krista A. Ehinger, James Hays, Antonio Torralba, and Aude Oliva. SUN database: Exploring a large collection of scene categories. *IJCV*, 2014. 9

[146] Hu Xu, Po-Yao Huang, Xiaoqing Ellen Tan, Ching-Feng Yeh, Jacob Kahn, Christine Jou, Gargi Ghosh, Omer Levy, Luke Zettlemoyer, Wen tau Yih, Shang-Wen Li, Saining Xie, and Christoph Feichtenhofer. Altogether: Image captioning via re-aligning alt-text. In *EMNLP*, 2024. 5, 20

[147] Hu Xu, Saining Xie, Xiaoqing Ellen Tan, Po-Yao Huang, Russell Howes, Vasu Sharma, Shang-Wen Li, Gargi Ghosh, Luke Zettlemoyer, and Christoph Feichtenhofer. Demystifying clip data. In *ICLR*, 2024. 1, 3, 8, 15, 19, 20, 32, 33

[148] Jun Xu, Tao Mei, Ting Yao, and Yong Rui. MSR-VTT: A large video description dataset for bridging video and language. In *CVPR*, 2016. 6, 7, 31, 32

[149] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, Guanting Dong, Haoran Wei, Huan Lin, Jialong Tang, Jialin Wang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Ma, Jin Xu, Jingren Zhou, Jinze Bai, Jinzheng He, Junyang Lin, Kai Dang, Keming Lu, Keqin Chen, Kexin Yang, Mei Li, Mingfeng Xue, Na Ni, Pei Zhang, Peng Wang, Ru Peng, Rui Men, Ruize Gao, Runji Lin, Shijie Wang, Shuai Bai, Sinan Tan, Tianhang Zhu, Tianhao Li, Tianyu Liu, Wenbin Ge, Xiaodong Deng, Xiaohuan Zhou, Xingzhang Ren, Xinyu Zhang, Xipin Wei, Xuancheng Ren, Yang Fan, Yang Yao, Yichang Zhang, Yu Wan, Yunfei Chu, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zhihao Fan. Qwen2 technical report. *arxiv:2407.10671*, 2024. 16

[150] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su,

Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *arXiv:2412.15115*, 2024. 16, 33

[151] Yang You, Jing Li, Sashank J. Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training BERT in 76 minutes. In *ICLR*, 2020. 3, 20

[152] Peter Young, Alice Lai, Micah Hodosh, and Julia Hockenmaier. From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. *TACL*, 2014. 9, 14, 15, 16, 32

[153] Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. CoCa: Contrastive captioners are image-text foundation models. *TMLR*, 2022. 1, 9, 20

[154] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. In *ICLR*, 2025. 20, 21

[155] Xiaohua Zhai, Basil Mustafa, Alexander Kolesnikov, and Lucas Beyer. Sigmoid loss for language image pre-training. In *ICCV*, 2023. 1, 4, 7, 9, 16, 19, 20, 22, 25, 26, 30

[156] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M Ni, and Heung-Yeung Shum. DINO: DETR with improved denoising anchor boxes for end-to-end object detection. In *ICLR*, 2023. 19

[157] Richard Zhang, Phillip Isola, and Alexei A Efros. Colorful image colorization. In *ECCV*, 2016. 20

[158] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D. Manning, and Curtis P. Langlotz. Contrastive learning of medical visual representations from paired images and text. In *MLHC*, 2022. 20

[159] Long Zhao, Nitesh Bharadwaj Gundavarapu, Liangzhe Yuan, Hao Zhou, Shen Yan, Jennifer J. Sun, Luke Friedman, Rui Qian, Tobias Weyand, Yue Zhao, Rachel Hornung, Florian Schroff, Ming Yang, David A. Ross, Huisheng Wang, Hartwig Adam, Mikhail Sirotenko, Ting Liu, and Boqing Gong. VideoPrism: A foundational visual encoder for video understanding. In *ICML*, 2024. 9

[160] Hanwen Zheng, Sijia Wang, Chris Thomas, and Lifu Huang. Advancing chart question answering with robust chart component recognition. In *WACV*, 2025. 14, 15, 16, 32

[161] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ADE20K dataset. In *CVPR*, 2017. 19, 29

[162] Jinguo Zhu, Weiyun Wang, Zhe Chen, Zhaoyang Liu, Shenglong Ye, Lixin Gu, Yuchen Duan, Hao Tian, Weijie Su, Jie Shao, Zhangwei Gao, Erfei Cui, Yue Cao, Yangzhou Liu, Weiye Xu, Hao Li, Jiahao Wang, Han Lv, Dengnian Chen, Songze Li, Yinan He, Tan Jiang, Jiapeng Luo, Yi Wang, Conghui He, Botian Shi, Xingcheng Zhang, Wenqi Shao, Junjun He, Yingtong Xiong, Wenwen Qu, Peng Sun, Penglong Jiao, Lijun Wu, Kaipeng Zhang, Huipeng Deng, Jiaye Ge, Kai Chen, Limin Wang, Min Dou, Lewei Lu, Xizhou Zhu, Tong Lu, Dahua Lin, Yu Qiao, Jifeng Dai, and Wenhai Wang. InternVL3: Exploring advanced training and test-time recipes for open-source multimodal models. *arxiv:2504.10479*, 2025. 2, 16