A. Implementation Details

A.1. Pre-training

Encoders. Table 9 shows the architecture we use. The design follows CLIP [52]. Our image encoder involves ViT-B, -L, -H [20], using the same patch size as in [20] (16 for B and L, 14 for H). We use global average pooling after the image encoder. The corresponding text encoder is of a smaller size, following [52]. We train ViT-B/-L with 256 TPU-v3 cores, and ViT-H with 512 cores. Table 9 also shows the model size of the image encoder, text encoder, and the entire model (including output projection layers).

Hyper-parameters. Our default pre-training configuration is shown in Table 10. We use the linear learning rate scaling rule [24]: $lr = base\_lr \times \text{batchsize} / 256$. We observe that using this rule allows us to change the batch size in ablations without extra learning rate search. The numerical precision we use is float32 by default. We also experimented with bfloat16, but only observed a $\sim$1.1$\times$ speedup, which is consistent with the results reported in Google's blog [4].

Unmasked tuning, which is a form of pre-training while disabling masking, follows Table 10, except that we lower the base learning rate to 4e-8 and shorten the warmup schedule to 25.6M samples.

A.2. ImageNet Classification

Zero-shot. We follow the prompt engineering in [52]. Their code provides 80 templates.[5] We use a subset of 7 templates they recommend; using all 80 templates gives similar results but is slower at inference.

Linear probing and fine-tuning. The setting follows [29]. See Table 11 and Table 12.

A.3. Zero-shot Retrieval

We evaluate the performance of zero-shot retrieval on two standard benchmarks: Flickr30K [73] and COCO [42], respectively with 1K and 5K image-text pairs in their test sets. Following the protocol in CLIP [52], we extract the image and text embeddings from the corresponding encoders and perform retrieval based on the cosine similarities over candidate image-text pairs; no prompt is used.

A.4. Zero-shot Robustness Evaluation

In our zero-shot robustness evaluation on the ImageNet-related sets, we use the 7 prompts provided by [52], only except in IN-R we use all 80 prompts that are better than the 7 prompts by noticeable margins. The dataset preparation and split follow OpenCLIP [36].[6] In ObjectNet, we follow [52] to use the class names without prompts. In YTBB, we use the VOC prompts provided by [52].

Model	Embed dim	Vision Transformer layers	width	heads	Text Transformer layers	width	heads	# params (M) vision	text	total
B/16	512	12	768	12	12	512	8	86	53	141
L/16	768	24	1024	16	12	768	12	303	109	414
H/14	1024	32	1280	16	24	1024	16	631	334	967

Table 9. **Encoder specifics.**

config	value
optimizer	AdamW [45]
base learning rate	4e-6
weight decay	0.2
optimizer momentum	β_1, β_2=0.9, 0.95 [10]
learning rate schedule	cosine decay [44]
warmup (in samples)	51.2M (B/L), 256M (H)
numerical precision	float32

Table 10. **Pre-training setting.**

config	value
optimizer	LARS [72]
base learning rate	0.01
weight decay	0
optimizer momentum	0.9
batch size	16384
learning rate schedule	cosine decay
warmup epochs	10
training epochs	90
augmentation	RandomResizedCrop

Table 11. **Linear probing setting.**

config	value
optimizer	AdamW
base learning rate	5e-5
weight decay	0.05
optimizer momentum	β_1, β_2=0.9, 0.999
layer-wise lr decay [14]	0.75
batch size	1024
learning rate schedule	cosine decay
warmup epochs	5
training epochs	50 (L/H)
augmentation	RandAug (9, 0.5) [15]
label smoothing [58]	0.1
mixup [76]	0.8
cutmix [75]	1.0
drop path [34]	0.2 (L/H)

Table 12. **Fine-tuning setting.**

A.5. More Zero-shot Datasets

For the experiments in Table 4, we use the prompts provided by [52].[7] We follow the data preparation scripts provided by [25] and [46] and load data using Tensorflow Datasets. Following [52], we report the mean accuracy per class for FGVC Aircraft, Oxford-IIIT Pets, Caltech-101, and Oxford Flowers 102 datasets; we report the mean of top-1 and top-5 accuracy for Kinetics-700, ROC AUC for Hateful Memes, and 11-point mAP for Pascal VOC 2007 Classification; we report top-1 accuracy for the rest of the

[4] https://cloud.google.com/blog/products/ai-machine-learning/bfloat16-the-secret-to-high-performance-on-cloud-tpus

[5] https://github.com/openai/CLIP/blob/main/notebooks/Prompt_Engineering_for_ImageNet.ipynb

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

datasets. We note that the Birdsnap dataset on Internet is shrinking over time and only 1850 test images are available for us (*vs*. 2149 images tested in [52], and 2443 originally).

A.6. Captioning

We build a sequence-to-sequence encoder-decoder transformer model on top of the ViT image encoder, with 3 encoder layers and 3 decoder layers following [7]. Specifically, the ViT image features are first linearly projected to a 384-dimensional sequence and further encoded by a 3-layer transformer encoder (of 384 width and 6 heads). For auto-regressive caption generation, we discard the pre-trained text encoder in FLIP and use a randomly initialized 3-layer transformer decoder (of 384 width and 6 heads) with cross-attention to encoder outputs. The model is trained to predict the next text token using the tokenizer in [52].

For simplicity, we supervise the image captioning model only with teacher forcing using a word-level cross-entropy loss [7]; we do not use the CIDEr score optimization in [7]. The full model is fine-tuned end-to-end with the AdamW optimizer, a batch size of 256, a learning rate of 1e-4 for newly added parameters, a weight decay of 1e-2, a warmup of 15% iterations, and a cosine decay learning rate schedule. The learning rate for the pre-trained ViT parameters is set to 1e-5 for ViT-L (and 5e-6 for ViT-H). The input image size is 512×512 for ViT-L/16 and 448×448 for ViT-H/14 (to keep the same sequence lengths).

All models are fine-tuned for image captioning on the COCO training split of [38] for 20 epochs. During inference, the image captions are predicted with auto-regressive decoding, and we report their performance on the COCO test split of [38] under different metrics.

To evaluate how the COCO-trained models generalize to novel objects, we evaluate these models directly on the nocaps [1] validation set, with no further fine-tuning.

A.7. Visual Question Answering

In our VQA experiments, we follow the architecture described in [21]. Specifically, the VQA task is casted as a classification problem over all answer classes. The input images are encoded by the ViT encoders. The input questions are encoded by a pre-trained RoBERTa text encoder [43], following the practice in [21]. A multimodal fusion Transformer (4 layers, 768-d, 12 heads, with merged attention [21]) is applied to combine the image and text representations. A two-layer MLP is applied on the class token of the fusion module to obtain the VQA output [21].

We fine-tune the VQA model end-to-end. The loss function is a binary sigmoid loss using soft scores [60]. We use a batch size of 256, a learning rate of 1e-4 for randomly initialized parameters, and a learning rate of 1e-5 (ViT-L) or 5e-6 (ViT-H) for the pre-trained ViT parameters. We use a weight decay of 1e-2, a warmup of 15% of iterations, and a cosine decay learning rate schedule. The input image size is 512×512 for ViT-L/16 and 448×448 for ViT-H/14.

All models are fine-tuned for 20 epochs on the VQAv2 train+val set, with additional question-answer pairs from Visual Genome [39], following [60]. We report results on the test-dev split from the evaluation server.

[7] https://github.com/openai/CLIP/blob/main/data/prompts.md