A Supplementary

A.1 Details of total compute calculations

The average FLOPs per learner update for an IID training run is given by:

$$C_{\text{IID}} = 3F_{\text{learn}}$$

where F_{learn} is the cost of a single learner inference pass and a gradient update costs $\sim 3\times$ inference passes [18]. The average FLOPs per learner update for easy-reference prioritisation is:

$$C_{\text{easy\_ref}} = \left(3F_{\text{learn}} + \frac{F_{\text{ref}}}{\text{SPI}}\right)\beta + 3F_{\text{ref}}$$

where $\text{SPI} = b/B$ ($= 0.5$ for our experiments) and β is the learner speedup vs. uniform sampling. The average FLOPs per learner update for RHO is:

$$C_{\text{RHO}} = \left(3F_{\text{learn}} + \left(F_{\text{learn}} + \frac{F_{\text{ref}}}{\text{SPI}}\right) + \right)\beta + 3F_{\text{ref}}$$

And the average FLOPs per update for *ActiveCLIP* / *ClassAct* is:

$$C_{\text{ClassAct / ActiveCLIP}} = \left(3F_{\text{learn}} + 2\frac{F_{\text{ref}}}{\text{SPI}}\right)\beta + 3F_{\text{ref}}$$

The conditions for compute positivity are therefore:

$$\left(3F_{\text{learn}} + \frac{F_{\text{act}}}{\text{SPI}}\right)\beta + 3F_{\text{ref}} < 3F_{\text{learn}}$$

where F_{act} is one of F_{ref}, $\left(F_{\text{learn}} + F_{\text{ref}}\right)$ or $2F_{\text{ref}}$ for "easy reference", RHO or ClassAct / ActiveCLIP, respectively.

A.2 Implementation details: ClassAct

Dataset: We pretrain on the JFT-300M dataset which contains labels for 18,292 classes collected in a semi-automated fashion. JFT labels are therefore noisy (consistently with other web-scale datasets), with approximately 20% of the examples being mislabeled. In the "standard" setting both reference models and subsequent ClassAct models are trained on the full JFT-300M dataset.

Architectures: We use standard vision transformers trained with 224×224 image resolution and a patch size of 16×16. We consider the standard ViT-Base and ViT-Large variants as learners in our in our main study, and smaller-scale variants (ViT-S and ViT-Tiny) when reducing the scoring-model computation. We introduce even smaller variants to scale down actor computation further, and detail the configurations of this "ViT-Micro" family in section A.7.

Learner Training: We train visual classifiers with softmax cross-entropy and label smoothing of 0.1. We use the AdamW optimizer [23] with a cosine learning-rate decay and warmup period of 10k iterations. The maximum learning rate is 0.001 and the weight decay is 10^{-3}.

Evaluation: We evaluate the performance of the learned model by applying it to a held-out set of JFT images, and measuring top-1 accuracy. Previous work has found that in-domain performance on JFT is highly predictive of out-of-distribution generalization [43], hence we leave the study of transfer performance to the multimodal setting (Sec. A.3).

A.3 Implementation details: ActiveCLIP

Datasets: We use paired image-text datasets for contrastive pretraining. We experiment with ALIGN, composed of 1.8B image-text pairs, LTIP, introduced in [2] and consisting of 312M higher-quality images with longer descriptions, and JFT-300M which despite being a classification dataset can be used for contrastive learning by using the labels as the paired text for each image [42].

Architectures: The model consists of a vision and text encoder in the standard contrastive learning setup introduced in [30]. We a vision transformer as the visual encoder and a text transformer encoder. We use the Base variant for both the vision and text transformers on the learner, following [9]. The smaller variants used by scoring models are the same as in the classification setup described above.

Learner Training: As described in Section 3.4 we optimize the model using the standard contrastive loss. We train the entire model with the AdamW optimiser and decay the learning rate using a cosine schedule (following a linear warmup period) with a maximum value of 10^{-3}. We also use a learnable temperature parameter as described in [30] and clip the gradients to a maximum global norm of 1.0. We use a batch size of 16,384 on the learner for each dataset.

Evaluation: We evaluate the model's zero-shot transfer results on standard multimodal benchmarks—image-text retrieval on COCO [20] and zero-shot image classification on ImageNet. For COCO, we use the standard technique of comparing image and text representations across the dataset to find the most similar text (or image) for each image (or text). For ImageNet, we precompute the representations for all predefined labels and calculate the most similar label for each image. For zero-shot classification, we also apply the prompt engineering described in [30] using the released prompts*. We average the embeddings for each label across all prompts before calculating similarity with the images.

* https://github.com/openai/CLIP/blob/main/notebooks/Prompt_Engineering_for_ImageNet.ipynb

A.4 Online ClassAct / ActiveCLIP

ClassAct / ActiveCLIP can also be run in a single training loop, by training the reference model on the super-batch:

Algorithm 2 Online ClassAct / ClassCLIP

Input: Randomly initialized learner model θ_l, small online and reference models θ_o and θ_r. Models use loss ℓ_{act} for scoring data and ℓ_{learn} for computing updates. Dataset $\mathcal{D}$, batch size B, sub-batch size $b < B$. Fast and slow learning rates α and β.
while training **do**
  $X \sim \mathcal{D}$, where $|X| = B$ ▷ Sample uniformly
  $S = \ell_{\text{act}}(X; \theta_o) - \ell_{\text{act}}(X; \theta_r)$ ▷ Get scores
  $\theta_r \leftarrow \text{Adam}_\alpha[\nabla_{\theta_r} \ell_{\text{learn}}(X|\theta_l)]$ ▷ Update ref. model
  $I \sim \text{SoftMax}(S)$, where $|I| = b$ ▷ Sample indices
  $Y = X[I]$ ▷ Collect sub-batch
  $\theta_l \leftarrow \text{Adam}_\beta[\nabla_{\theta_l} \ell_{\text{learn}}(Y|\theta_l)]$ ▷ Update learner model
  $\theta_o \leftarrow \text{Adam}_\beta[\nabla_{\theta_o} \ell_{\text{learn}}(Y|\theta_o)]$ ▷ Update online model
end while

We found that $B = 10b$ (SPI=0.1) was sufficient to reproduce the pre-trained reference (Algorithm 1) case where $B = 2b$ (SPI=0.5), as described Section 4.4. It is possible that the super-batch size could be shrunk, but we did not evaluate SPI values in between these values.

A.5 Analyzing learnability scores

Here we examine learnability scores in the case of small differences between the online and reference models θ^t, θ^*. In this case, we can apply a Taylor expansion of the reference-model loss:

$$\ell(\boldsymbol{x}_i|\theta^*) \approx \ell(\boldsymbol{x}_i|\theta^t) + (\theta^* - \theta^t) \cdot \nabla_\theta \ell(\boldsymbol{x}_i|\theta^t) \tag{9}$$

and learnability scores simplify to the alignment between the gradient of the loss with respect to the online parameters, and the difference between online and reference parameters:

$$s^{\text{learn}}(\boldsymbol{x}_i|\theta^t, \theta^*) = \ell(\boldsymbol{x}_i|\theta^t) - \ell(\boldsymbol{x}_i|\theta^*) \tag{10}$$

$$\approx (\theta^t - \theta^*) \cdot \nabla_\theta \ell(\boldsymbol{x}_i|\theta^t). \tag{11}$$

This yields a simple explanation for the relevance of learnability scores: data points whose gradient aligns well with the "direction of travel" (*i.e.* the difference between the current model state and the fully-trained one) will be prioritized. We can further simplify this expression in the case where the reference model is

the result of a single, global update with respect to gradients from a batch of data $\{\boldsymbol{x}_j\}$:

$$\theta^* = \theta^t - \lambda \nabla_\theta \ell(\{\boldsymbol{x}_j\}|\theta) \tag{12}$$

$$s^{\text{learn}}(\boldsymbol{x}_i|\theta^t, \theta^*) \approx \lambda \nabla_\theta \ell(\boldsymbol{x}_i|\theta^t) \cdot \nabla_\theta \ell(\{\boldsymbol{x}_j\}|\theta^t) \tag{13}$$

In this case, the learnability of an example reduces to the alignment of its gradient to those of the batch. In particular, these scores will de-prioritize examples which are trivially solved (small gradients) or noisy (mis-aligned with the batch gradient).

Finally, we note that learnability scores reduce to gradient-norm prioritization [28] if we make the further (more stringent) assumption that the reference model is the result of a single update with respect to *this example's gradient only*:

$$\theta^* = \theta^t - \lambda \nabla_\theta \ell(\boldsymbol{x}_i|\theta) \tag{14}$$

$$s^{\text{learn}}(\boldsymbol{x}_i|\theta^t, \theta^*) \approx s^{\text{grad}}(\boldsymbol{x}_i|\theta^t) = \lambda \left|\left|\nabla_\theta \ell(\boldsymbol{x}_i|\theta^t)\right|\right|^2 \tag{15}$$

While this prioritization effectively discards examples with small gradients, it does not benefit from the denoising properties of the more general formulation of learnability, as noisy examples can exhibit large gradients.

A.6 Learning infrastructure

Our learning infrastructure Figure 8 draws inspiration from distributed reinforcement learning (DRL) [10]. In contrast to DRL, where run loops generate data through interactions with an environment, run loops in our system are modified to read offline data. In addition to obtaining data, run loops run inference on the data via remote inference servers and write the inference outputs to prioritized replay data stores [5]. The distributed nodes of our learning setup are configured and connected using [41].

Data stored is sampled from the prioritized replay based on priorities determined during inference. Remote inference servers continuously run inference at specific batch sizes, in parallel to learning. Parameters of the inference servers are updated after each learning step.

This setup allows us to run ActiveCLIP at the same speed as uniform sampling experiments, even without approximating the scoring models. This distributed setup may be advantageous over the sequential online batch selection setup (where the actors and learners run on the same hardware) in cases where time, not compute, is the bottleneck; adding computation to parallelize the data filtration is an alternative to increasing the batch size, which is known to saturate [44].

For stability, we control the ratio of data sampled to data inserted in all of our experiments. For example, a samples-per-insert (SPI) ratio of 0.5 means that half of the inserted data does not get sampled, ensuring the learner focuses on the most relevant data. Each data item is sampled at most once. These two

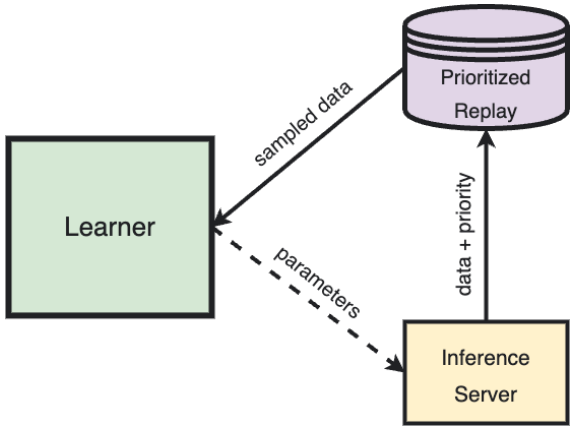

Fig. 8: The main nodes of our distributed learning infrastructure are: (1) a learner that continuously updates model parameters based on sampled data, (2) an inference server which computes priorities of uniformly sampled data, and (3) a prioritized replay which receives data and priorities from the inference server and samples prioritized data for learning. The critical aspect of our system is adaptive data prioritization which is enabled by keeping inference parameters in sync with the learner. We employ separate prioritized replay and inference server nodes per dataset. We also employ run loop nodes (not shown in the diagram) for reading and writing data. Run loop nodes can run as threads on the inference server node or remotely, to prevent extra load on inference servers.

constraints impose minimum inference requirements to fully saturate the learner, which we separately tune for each experiment by scaling up the topology of the inference server.

Our infrastructure is designed to work with multiple datasets. In this case, we use separate remote inference server, run loops, and prioritized replay for each dataset, and set the batch size per dataset.

To continuously assess the performance of our models, we add data evaluator nodes which measure performance on held-out datasets while periodically pulling the latest parameters from the learner node.

A.7 Small ViT models

We extended the suite of ViT models presented in [43] by further scaling down the ViT-Ti model as shown in Table 4.

B Appendix: supplementary results

B.1 FLOP efficient contrastive training

Our previous results (Table 2) showed that it is possible to improve the performance of contrastve pre-training using active data selection via ActiveCLIP.

Name	Width	Depth	MLP	Heads	Mio--Params	GFLOPs 224^2
L	1024	24	4096	16	304	61.6
B	768	12	3072	12	86	17.6
S	384	12	3072	12	22	4.6
Ti	192	12	768	3	5.6	1.3
Mu_6-3	192	6	768	3	2.8	1.24
Mu_12-2	128	12	512	2	2.5	1.23
Mu_6-2	128	6	512	2	0.66	0.48
Mu_12-1	64	12	256	1	1.3	0.23
Mu_6-1	64	6	256	1	0.36	0.203
Mu_3-1	64	3	256	1	0.22	0.065
Mu_2-1	64	2	256	1	0.16	0.033
Mu_1-1	64	1	256	1	0.12	0.011

Table 4: ViT model details. Configurations, parameter counts, and FLOPs associated with small ViT models.

However, these results used a ViT-B policy (reference and online models) to train a ViT-B learner. To explore whether similar gains could be achieved with fewer FLOPs, we repeated out analyses using ViT-Ti policies (Table 5). As in our ClassAct JFT experiments, our results show that performance degrades marginally, eve though the ViT-Ti policy requires 50x fewer FLOPs.

		IN-1K	COCO	
Method	Train ex.	ZS Top-1	im2txt	txt2im
CLIP [30]	13B	68.3	52.4	33.1
EVA-CLIP [37]	3B	69.7		
SigLIP [44]	3B	68.8	56.7	40.2
ActiveCLIP-B	3B	**70.3**	**57.8**	**42.6**
ActiveCLIP-Ti	3B	**69.9**	**57.2**	**42.4**

Table 5: Comparison of *ActiveCLIP* to public multimodal pre-taining methods Extension of Table 2, replacing the ViT-B reference and online models with a ViT-Ti. The ViT-Ti policy marginally underperforms the ViT-B policy, but is also SOTA vs. previous baselines.

We also repeated this ablation for our out-of-domain contrastive pre-training experiments shown in Fig. 6). Again, ViT-Ti policies produces similar speedups and were also compute-positive, mirroring our ClassAct JFT experiments (Figure 5), 4).

Method	Ref. model	ImageNet 0-Shot Speed -up	ImageNet 0-Shot FLOP saving	ImageNet 0-Shot Top 1 Acc.	COCO (im2txt) Speed -up	COCO (im2txt) FLOP saving	COCO (im2txt) R@1	COCO (txt2im) Speed -up	COCO (txt2im) FLOP saving	COCO (txt2im) R@1
IID		0.0%	0.0%	0.386	0.0%	0.0%	0.256	0.0%	0.0%	0.457
ActiveCLIP	ViT-Ti	43.3%	26.2%	0.428	38.7%	20.8%	0.292	38.3%	20.3%	0.509
ActiveCLIP	ViT-B	51.0%	-163.4%	0.449	48.3%	-172.4%	0.305	47.3%	-175.6%	0.532

Table 6: Compute-positive contrastive pre-training. Reproduction of experiments from Figure 6, replacing the ViT-B reference and online models with ViT-Ti. As in our ClassAct JFT experiments (Figure 5), 4), the ViT-Ti policy produced nearly the same learner speedups as the ViT-B policy, but unlike the ViT-B policy also produced net FLOP savings. In all cases the reference model is trained on LTIP and the learner is trained on ALIGN.