Co-training 2^L Submodels for Visual Recognition

supplemental material

A. Training details

A.1. Hyper-parameters

We use by default the DeiT-III training procedure from Touvron *et al*. [51], which uses separate recipes for Imagenet1k and Imagenet21k. The most noticeable difference is the loss, which is by default a binary cross-entropy when training on Imagenet1k, versus a cross-entropy when pre-training with Imagenet21k and fine-tuning with Imagenet1k. We depart from the choices of DeiT-III as follows. First, we systematically set the weight decay to 0.02, independent of whether we pre-train on Imagenet21k or train or fine-tune on Imagenet1k. This does not change significantly the results with cosub. Our long schedule on Imagenet21k is systematically set to 270 epochs.

Batch size and learning rate. Second, the default batch size is by default set to 2048. However we need to reduce it to limit the memory consumption for larger models or higher resolution. In particular, during the fine-tuning stage we adjust the learning accordingly and employ a square-root scaling rule compatible with AdamW [36]: we fix the base learning rate as

$$\mathrm{LR}_{\mathrm{train}} = 10^{-3}\sqrt{\frac{\mathrm{BS}}{2048}}, \tag{A.1}$$

for pre-training on Imagenet21k with 90 epochs or training on Imagenet1k from scratch (400 or 800 epochs). When fine-tuning from Imagenet21k to Imagenet1k, we divide by 10 the base learning rate when starting from an existing model. Therefore we set

$$\mathrm{LR}_{\mathrm{finetune}} = 10^{-4}\sqrt{\frac{\mathrm{BS}}{2048}} \times C_{\mathrm{LD}}, \tag{A.2}$$

where we set the constant C_{LD} =1 by default. This constant is modified when using LayerDecay [10], see below.

LayerDecay is used in fine-tuning stages of recent self-supervised methods [5, 22]. It decreases the learning rate in a geometrically decreasing manner: the learning for each block l is given by $\mathrm{LR}(l) = \mathrm{LD}^{l-L}$, where LR is the layer-wise decay factor, and L is the total number of blocks of the network (e.g., 32 for a ViT-H). Hence the LR of the all layers is affected, except those in the final block.

model	ViT-S	ViT-M	ViT-B	ViT-L	ViT-H
τ: Imnet1k train	0.05	0.1	0.2	0.45	0.6
τ: Imnet21k pre-train	0.05	0.05	0.1	0.3	0.5
LayerDecay	0.7	0.75	0.75	0.8	0.85

Table A.1. Hyper-parameters that are set depending on the model size.

BeiT sets the LayerDecay parameter to LD = 0.65 or 0.75 based on the model size, like OmniMAE [21]. MAE [22] sets LD to 0.75. Another recent work uses 0.65 [3]. From our own preliminary experiments, we concur with the choice of Bao *et al*. [5], who adjust this parameter depending on the model size when fine-tuning from Imagenet21k to Imagenet1k. Hence we gradually increase the LD value from smaller to larger models. Table A.1 gives the value of this parameter for each model size.

We set $C_{\mathrm{LD}} = 2$ if we use LayerDecay in the fine-tuning stage: if a given learning rate was initially optimized without LayerDecay, it is necessary to compensate the overall reduction of updates. This was suggested by Bao *et al*. [5], but without any guideline on how to adjust the learning rate. From a few experiments, we notice that the simple formulaic choice of multiplying the learning by a constant 2 generally gives reasonable results. They could likely be further improved by further cross-validation, however this would require a much heavier set of experiments per model size.

Hyper-parameter τ. The other hyper-parameter that depends on the model size is the so-call drop-path rate τ associated with stochastic depth [31]. We report these values in Table A.1. In particular, the value τ is inherently intertwined with our approach, as it is used to instantiate submodels. A value of τ means that we instantiate two identical submodels, which zeroes the cross-entropy and therefore cancels our method cosub. More generally, higher values of τ provide submodels that have less layers in common. Therefore, we observe that it is beneficial to increase τ compared to the values suggested in the DeiT-III training method, especially for the smaller model Vit-S for which τ was initially set to 0. We further increase this rate by +0.05 when more regularization is needed, i.e., when pre-training during 270 epochs on Imagenet21k or for large resolutions.

A.2. Transfer Learning datasets

For the transfer learning tasks we fine-tune our ViT models pre-trained at resolution 224×224 on ImageNet-1k only

Dataset	Train size	Test size	#classes
iNaturalist 2018 [30]	437,513	24,426	8,142
iNaturalist 2019 [29]	265,240	3,003	1,010
Flowers-102 [38]	2,040	6,149	102
Stanford Cars [32]	8,144	8,041	196
CIFAR-100 [33]	50,000	10,000	100
CIFAR-10 [33]	50,000	10,000	10

Table A.2. Datasets used for our different transfer-learning tasks.

with cosub on the 6 transfer learning datasets used in Touvron et al. [51]. In Table A.2, we give the characteristics of these datasets and corresponding references.

B. Supplemental experiments

B.1. New baselines for models of the literature

Table B.1 provides the results we obtained with minimal adjustments to our training recipe based on DeiT-III combined with submodel co-training. The only parameter that absolutely needs to be re-adjusted is τ. For this purpose, we first tried existing parameter setting from the literature when existing ($\tau = 0$ disables cosub therefore we use 0.05 in such cases). Otherwise we make a guess based on the model size and adjusts by step of 0.1 when the training curve exihbits some overfitting. This minimum hyper-parameter modification with a coarse step for τ is most likely suboptimal and could certainly be improved, but it would require much more compute capacity to optimize it with a proper cross-validation for each model.

As one can see, our method improves the results for most of the models from the literature that we tested, therefore we hope that they could serve as improved baselines when comparing architectures. We also notice that our results on Imagenet-v2 are generally better than those reported in the literature. For instance our ConvNext-B training is comparable to that the original paper on Imagenet-val, but cosub's result on Imagenet-v2 is more than 1% higher, which suggests that our training recipe overfits significantly less.

As a disclaimer, we believe that Table B.1 should not be used as a way to compare the merits of architectures, since our training procedure may favor certain of them. As importantly and as mentioned above, we have put a minimal effort to obtained these results and it is highly likely that our hyper-parameters are very suboptimal for some models.

B.2. Trade-offs between resolution and model size

Since both the model size and the resolution increase the accuracy and the complexity, the question is which combination (model,resolution) we should use. This question was noticeably analyzed by Bello *et al*. [6] for ResNet, who pointed out that the Pareto-optimal resolution is typically lower than what was employed when the measure of complexity are FLOPS. We report trade-offs for different complexity measures in Figure B.1, see also Table 9 in the main paper. Selecting a ViT operating at resolution 224 generally seems a good strategy. It is unclear whether this choice is absolutely good, or if it is better just because most of the hyper-parameter tuning effort in this paper and previous ones has been carried out at this specific resolution.

Model	params (M)	FLOPS ($\times 10^9$)	previous top1 acc.	cosub acc. -val	-v2
ResNet-152 [24]	60	11.6	$82.0^{(1k)}$ [55]	83.1	73.1
RegNet-16GF [42]	84	16.0	$82.2^{(1k)}$ [55]	84.2	74.7
PiT-B -distilled [26]	74	12.5	$84.5^{(1k)}$	85.8	76.8
ConvNext-S [35]	50	8.7	$83.1^{(1k)}$	85.2	76.0
ConvNext-B [35]	89	15.4	$85.8^{(21k)}$	85.8	76.9
XCiT-S12 [17]	26	4.9	$83.3^{(1k)}$	84.2	74.9
XCiT-M24 [17]	84	16.2	$84.3^{(1k)}$	86.5	77.9
XCiT-L24 [17]	189	36.1	$84.9^{(1k)}$	87.2	77.8
Swin-B [34]	88	15.4	$85.2^{(21k)}$	86.2	77.2
Swin-L [34]	197	34.5	$86.3^{(21k)}$	87.1	78.1

Table B.1. **New baselines for multiple architectures at resolution 224:** trained with cosub on Imagenet21k data. We adopt the same pre-training recipe (90 epochs of Imnet21k pretraining and 50 epochs of fine-tuning) and adjust the τ parameter per architecture based on prior choices or best guess based on model size. These choices could most likely be improved by cross-validated grid search. We report good results reported in literature (in some cases obtained by training on Imagenet-train only:$^{(1k)}$)

B.3. Comparison with BerT-like approaches

Although it is fully supervised, our cosub method shares some similarities with purely self-supervised approaches such as DINO [9], MAE [22], or BeiT [5]. Indeed, the cosub loss on submodels can be seen as an unsupervised or self-supervised loss. In contrast to cosub, DINO does not backpropagate on the model that serves as a teacher, since this one is obtained by EMA: we cannot differentiate because it is based on past models that are not stored anymore.

In Table B.2 we compare cosub with BerT-like pretraining approaches as they are known to be very effective with vision transformers. We find that our approach outperforms these competitive approaches when we can pretrain on Imagenet21k. Our cosub approach could potentially be used for unsupervised training, or to finetune self-supervised models: for instance, BeiT is typically finetuned on Imagenet-21k during a large number of epochs. We leave this exploration to future work.

B.4. Significance of the results

We measure avg/std for a few points. Since computing multiple results for each number of Table 1 requires a lot of resources, we measure the uncertainty in a cheaper setting: we train during 400 epochs for 5 seeds (0 – 4) at resolution 112. The results in top-1 accuracy are as follows:

	Vit-S@112	Vit-B@112	Vit-L@112	Vit-H@112
baseline	72.63±0.367⋆	77.03±0.122	79.46±0.130	80.94±0.066
cosub	73.15±0.098	77.75±0.098	80.64±0.150	82.29±0.088

⋆: *This high std-dev is due to a single underperforming model. Best seed gets 72.83.*

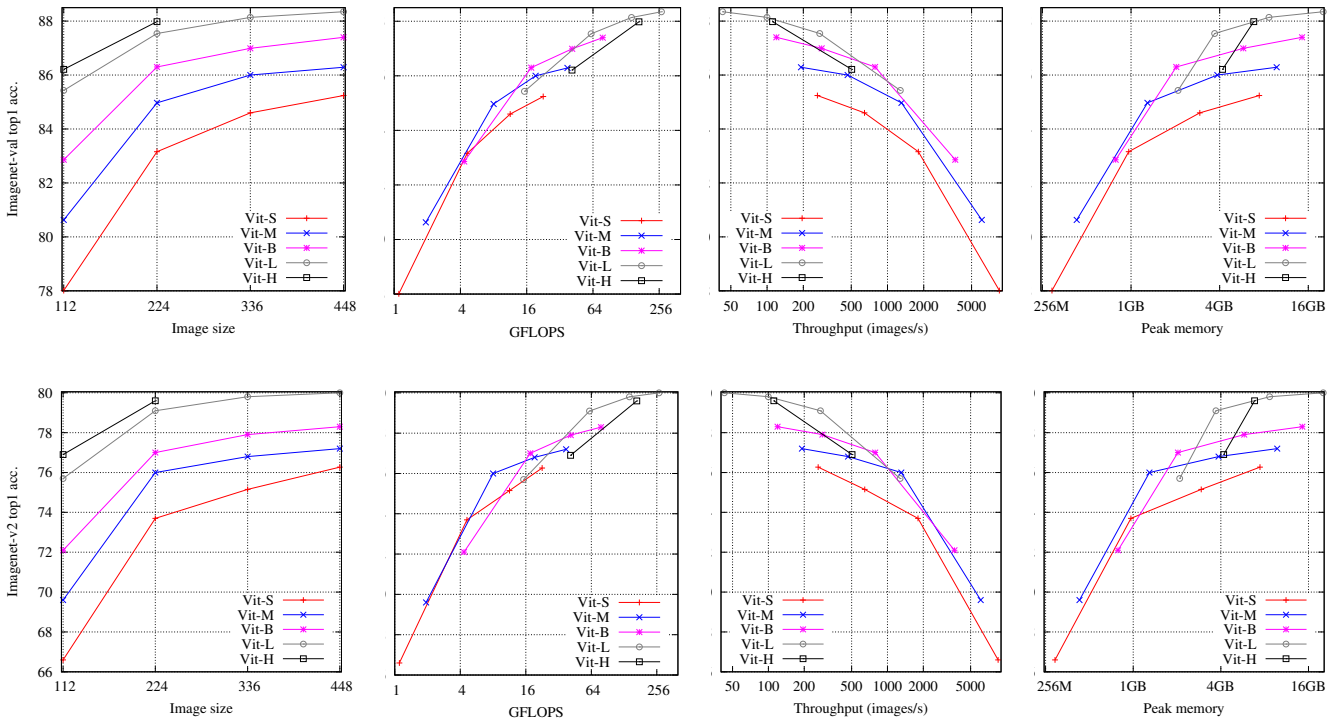

Figure B.1. **Flops/accuracy trade-offs:** We measure the accuracy on (*top*) Imagenet1k-val and (*bottom*) Imagenet-v2 as a function of different measures of complexity, which we vary for each model by increasing the resolution: 112×112, 224×224, 336×336, 448×448 (except for ViT-H, where we stop at 224×224). All those models were pre-trained on Imagenet21k and fine-tuned on Imagenet1k.

C. Efficient stochastic depth

Quantization of stochastic depth rate. Our ESD variant of stochastic depth determines a reduced batch size per GPU as follows: we multiply the input local batch size per GPU (LBS) by the requested drop-path hyper-parameter τ, which produces the actual local batch size as

$$\text{LBS}_{\text{ESD}} = \lfloor \tau \times \text{LBS} \rfloor. \tag{C.1}$$

If LBS is large enough, this rounding has little effect. However, for large models or high-resolution images, the local batch size can become small, thereby leading to a more aggressive rounding when computing LBS_{ESD}. This leads to a coarse approximation of the stochastic depth parameter τ, as shown in Figure C.1. For example, for a local GPU batch size of 8, the only possible values of τ are: $\tau \in \{0, 0.125, 0.250, 0.375, 0.5, 0.625, 0.75, 0.875, 1\}$, because the actual batch size is floored to an integer. In this figure, we report the mapping between the effective drop-rate and the actual one. In practice, this quantization effect must be taken into consideration in extreme cases (very large models or higher resolution images). In such cases, we compute the effective stochastic depth when computing the ratio $1/(1-\tau)$ for the inference-time model.

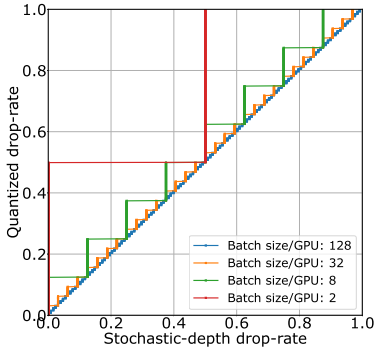

Figure C.1. Efficient stochastic depth: we measure the effect of quantization on the effective drop path rate depending on the batch-size.

Complementary complexity measurements. We can optimize the setting for ESD by using a larger batch size or less nodes. Below we report measurements repeating and complementing Table 7:

Setting	#GPUs (V100)	batch size	peak Mem	time/epoch
Vit-L baseline	4×8	2048	22.5 GB	8 min 56 s
Vit-L ESD	4×8	2048	15.1 GB	9 min 05 s
Vit-L ESD	2×8	2048	26.9 GB	15 min 30 s
Vit-L ESD	4×8	4096	27.0 GB	7 min 50 s

	Model	Method	#pretraining epochs	#finetune epochs	ImageNet val	ImageNet Real	ImageNet V2
Imnet-1k pre-/training	ViT-B	BeiT	300	$100^{(1k)}$	82.9	-	-
			800	$100^{(1k)}$	83.2	-	-
		MAE*	1600	$100^{(1k)}$	83.6	88.1	73.2
		Ours	$400^{(1k)}$	$20^{(1k)}$	83.8	88.6	73.5
			$800^{(1k)}$	$20^{(1k)}$	**84.2**	**88.5**	**74.2**
	ViT-L	BeiT	800	$30^{(1k)}$	85.2	-	-
		MAE	400	$50^{(1k)}$	84.3	-	-
			800	$50^{(1k)}$	84.9	-	-
			1600	$50^{(1k)}$	85.1	-	-
		MAE*	1600	$50^{(1k)}$	**85.9**	**89.4**	**76.5**
		Ours	$400^{(1k)}$	$20^{(1k)}$	85.0	89.4	75.5
			$800^{(1k)}$	$20^{(1k)}$	85.3	89.2	75.5
Imnet-21k pre-training	ViT-B	BeiT	150	$50^{(1k)}$	83.7	88.2	73.1
			$150 + 90^{(21k)}$	$50^{(1k)}$	85.2	89.4	75.4
		Ours	$90^{(21k)}$	$50^{(1k)}$	86.0	**89.8**	**77.0**
			$270^{(21k)}$	$50^{k)}$	**86.3**	89.7	**77.0**
	ViT-L	BeiT	150	$50^{k)}$	86.0	89.6	76.7
			$150 + 90^{1k)}$	$50^{k)}$	**87.5**	90.1	78.8
		Ours	$90^{1k)}$	$50^{k)}$	**87.5**	**90.3**	**79.1**

Table B.2. Comparison of self-supervised pre-training with our supervised approach. All models are evaluated at resolution 224×224. We report image classification results on ImageNet val, real and v2 in order to evaluate overfitting. $^{(21k)}$ indicate a finetuning with labels on ImageNet-21k and $^{1k)}$ indicate a finetuning with labels on ImageNet-1k. * indicates the improved setting of MAE using pixel (w/ norm) loss. MAE training is more efficient for a given number of epochs, thanks to its masking strategy.

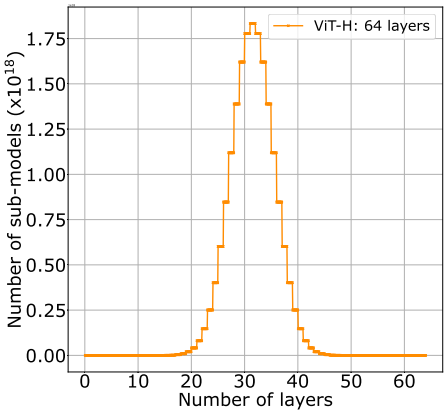

Figure D.1. Number of submodels with a given number of layers for ViT-H (32 blocks). $\tau = 0.5$ gives the same probability to instantiate each.

D. Submodel analysis & using more submodels

Number of layers with stochastic depth. In Figure D.1, we show the number of submodels exist for a given number of layers. This corresponds to a binomial distribution, which attains its maximum with 32 layers. Setting $\tau = 0.5$ gives the same probability to instantiate each of those, hence we can se that it stochastic depth will draw with very high probability a model that contains about 32 layers.

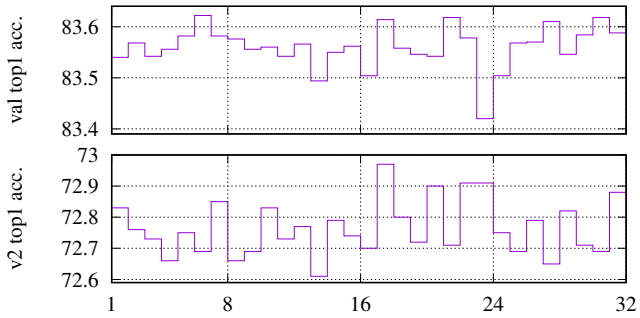

Figure D.2. Layer ablation: we trim one block (i.e., a multi-head self-attention and its corresponding "FFN") of a fixed ViT-H network 126×126 learned with submodel co-training, and evaluate the performance of the L corresponding subnetworks. Variations are overall small and there is no strong correlation between Imagenet-val and -v2 accuracy.

Layer ablation. Inspired by experiments from Fan et al. [20], we measure the performance of the submodels produced when dropping exactly one block (Multi-head soft-attention and corresponding Feedforward) from a trained ViT model, in this case a ViT-H trained at resolution 126×126 on Imagenet1k. Our objective is to measure whether some layers are more important than others. The performance of the submodels are reported in Figure D.2. We observe that almost most of the submodels have almost an identical performance on Imagenet1k-val, around 83.55% in top-1 accuracy, close to the performance (83.6%) of the full 32-blocks/64-layers model.

More submodels. In our paper, all the experiments have been carried out by considering 2 submodels, as depicted in Figure 1. Following a suggestion by a reviewer[2], we have also considered an extension with 3 submodels. In this case the number of distillation terms is more important: the extension of Equation 2 gives 6 distillation terms, versus 2 in with two submodels. We ensure that the relative importance of the actual labels remains the same by providing it a relative weight of 0.5. All hyper-parameters are kept identical in this variant. Our small experiments on 112×112 images trained during 400 epochs seem to indicate a gain:

	Vit-S@112	Vit-B@112	Vit-L@112	Vit-H@112
baseline	72.63	77.03	79.46	80.94
cosub – 2 submodels	73.26	77.75	80.69	82.27
cosub – 3 submodels	73.17	78.22	81.29	82.83

However in settings with high-resolution images and longer training schedules, and similarly on Imagenet21k, we did not observe any clear advantage of extracting 3 submodels instead of 2. Since the complexity is further increased by a factor 1.5, in these settings there is no clear benefit to go beyond 2 submodels.

[2]We thank all the reviewers for their constructive comments, their corrections and suggestions.