This CVPR paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Co-training 2^L Submodels for Visual Recognition

Hugo Touvron[⋆,†] Matthieu Cord[†] Maxime Oquab[⋆]
Piotr Bojanowski[⋆] Jakob Verbeek[⋆] Hervé Jégou[⋆]
[⋆]Meta AI / FAIR Paris [†]Sorbonne University

Abstract

We introduce submodel co-training, a regularization method related to co-training, self-distillation and stochastic depth. Given a neural network to be trained, for each sample we implicitly instantiate two altered networks, "submodels", with stochastic depth: we activate only a subset of the layers. Each network serves as a soft teacher to the other, by providing a loss that complements the regular loss provided by the one-hot label. Our approach, dubbed "cosub", uses a single set of weights, and does not involve a pre-trained external model or temporal averaging.

Experimentally, we show that submodel co-training is effective to train backbones for recognition tasks such as image classification and semantic segmentation. Our approach is compatible with multiple architectures, including RegNet, ViT, PiT, XCiT, Swin and ConvNext. Our training strategy improves their results in comparable settings. For instance, a ViT-B pretrained with cosub on ImageNet-21k obtains 87.4% top-1 acc. @448 on ImageNet-val.

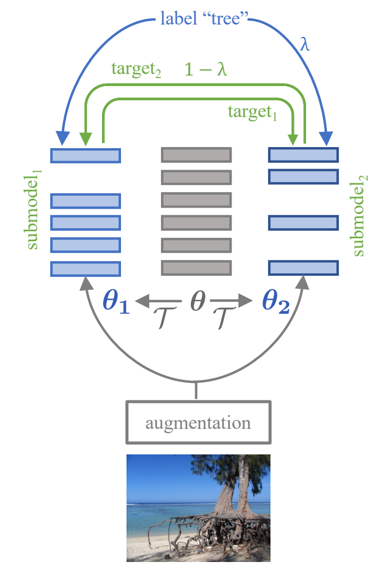


Figure 1. **Co-training of submodels (cosub):** for each image, two submodels are sampled by randomly dropping layers of the full model. The training signal for each submodel mixes the cross-entropy loss from the image label with a self-distillation loss obtained from the other submodel.

1. Introduction

Although the fundamental ideas of deep trainable neural networks have been around for decades, only recently have barriers been removed to allow breakthroughs in successfully training deep neural architectures in practice. Many of these barriers are related to non-convex optimization in one way or another, which is central to the success of modern neural networks. The optimization challenges have been addressed from multiple angles in the literature. First, modern architectures are designed to facilitate the optimization of very deep networks. An exceptionally successful design principle is using residual connections [24, 25]. Although this does not change the expressiveness of the functions that the network can implement, the improved gradient flow alleviates, to some extent, the difficulties of optimizing very deep networks. Another key element to the optimization is the importance of data, revealed by the step-change in visual recognition performance resulting from the ImageNet dataset [11], and the popularization of transfer learning with pre-training on large datasets [39, 58].

However, even when (pre-)trained with millions of images, recent deep networks with millions if not billions of parameters, are still heavily overparameterized. Traditional regularization like weight decay, dropout [46], or label smoothing [47] are limited in their ability to address this issue. Data-augmentation strategies, including those mixing different images like Mixup [61] and CutMix [60], have proven to provide a complementary data-driven form of regularization. More recently, multiple works propose to resort to self-supervised pre-training. These approaches rely on a proxy objective that generally provides more supervision signal than the one available from labels, like in recent (masked) auto-encoders [5, 16, 22], which were popular in the early deep learning literature [7, 19, 27]. Similarly, contrastive approaches [23] or self-distillation [9] provide a richer supervision less prone to supervision collapse [12]. Overall, self-supervised learning makes it possible to learn larger models with less data, possibly reducing the need of a pre-training stage [15].

Distillation is a complementary approach to improve optimization. Distillation techniques were originally developed to transfer knowledge from a teacher model to a student model [4, 28], allowing the student to improve over learning from the data directly. In contrast to traditional distillation, co-distillation does not require pre-training a (strong) teacher. Instead, a pool of models supervise each other. Practically, it faces several limitations, including the difficulty of jointly training more than two students for complexity reasons, as it involves duplicating the weights.

In this paper, we propose a practical way to enable co-training for a very large number of students. We consider a single target model to be trained, and we instantiate two submodels *on-the-fly*, simply by layerwise dropout [20, 31]. This gives us two neural networks through which we can backpropagate to the shared parameters of the target model. In addition to the regular training loss, each submodel serves as a teacher to the other, which provides an additional supervision signal ensuring the consistency across the submodels. Our approach is illustrated in Figure 1: the parameter λ controls the importance of the co-training loss compared to the label loss, and our experiments show that it significantly increases the final model accuracy.

This co-training across different submodels, which we refer to as *cosub*, can be regarded as a massive co-training between 2^L models that share a common set of parameters, where L is the number of layers in the target architecture. The target model can be interpreted as the expectation of all models. With a layer drop-rate set to 0.5, for instance for a ViT-H model, all submodels are equiprobable, and then it amounts to averaging the weights of $2^{2\times 32}$ models.

Our contributions can be summarized as follows:

- We introduce a novel training approach for deep neural networks: we co-train submodels. This significantly improves the training of most models, establishing the new state of the art in multiple cases. For instance, after pre-training ViT-B on Imagenet-21k and fine-tuning it at resolution 448, we obtain 87.4% top-1 accuracy on Imagenet-val.
- We provide an efficient implementation to subsample models on the fly. It is a simple yet effective variation of stochastic depth [31] to drop residual blocks.
- We provide multiple analyses and ablations. Noticeably, we show that our submodels are effective models by themselves even with significant trimming, similar to LayerDrop [20] in natural language processing.
- We validate our approach on multiple architectures (like ViT, ResNet, RegNet, PiT, XCiT, Swin, ConvNext), both for image classification –trained from scratch or with transfer–, and semantic segmentation.
- We will share models/code for reproducibility in the DeiT repository.

2. Related work

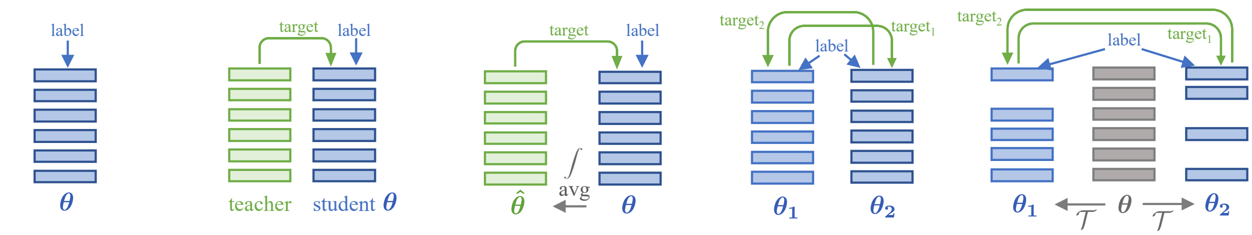
Knowledge distillation. Originally, distillation was introduced as a way to train a model such that it reproduces the performance of another model [4, 28]. The typical use-case is to improve the quality of a relatively small model by leveraging a strong teacher, whose complexity may be prohibitive for a practical deployment. The teacher's soft labels have a similar effect as label smoothing [59]. As shown by Wei *et al*. [53], the teacher's supervision takes into account the effects of the data augmentation, which sometimes causes a misalignment between the real label and the image. Knowledge distillation can transfer inductive biases [1] in a soft way in a student model when using a teacher model these biases are enforced in a hard way. Touvron *et al*. [50] proposed a variant of distillation adapted to Vision Transformer (ViT), showing the effectiveness of using a ConvNet teacher for a ViT student.

Mixture models. Ensembling has a long history that we can trace back to the origins of statistics [37] and the possibility of improving the precision of measurements with multiple observations. In machine learning, bagging [8] combines multiple weak classifiers to produce a strong one. This idea is naturally extended to neural networks, where it can offer improved stability or accuracy, or other properties like anytime inference [44]. A mixture model can also be seen as a larger model with an internal parallel structure.

Co-distillation. At the interface of mixture models and distillation, the concept of co-distillation [62, 64] does not require a prior teacher: the mixture serves as the teacher to all the networks in the mixture. The models are jointly optimized, which leads to improved accuracy of the individual models [2]. This can be regarded as a form of collaborative learning between the different elements of the mixture [45]. Compared to training only one model, co-distillation involves training two or more models, each requiring storage for weights and activation, and computing backward and forward passes.

Exponential moving average teacher. A special form of teacher is a model obtained from the student model itself. Tarvainen and Valpila [49] show that such a model obtained by temporal averaging of intermediate checkpoints during the training is an effective teacher, that can be obtained with an exponential moving average (EMA). This idea has also been adopted in self-supervised training with learning schemes like DINO [9]. This method involves storing a copy of the weights corresponding to the temporal averaging carried out by the EMA model.

Dropout and model populations. Dropout [46] is an effective way to regularize models. With residual architectures, one effective way to train deeper architecture is stochastic depth [31]. It reduces the size of a network at



(a) supervised (b) knowledge distillation (c) self-distillation (d) co-training (e) submodel co-training

Figure 2. Brief summary of related works and of submodel co-training (cosub). (a) supervised baseline: the supervision is provided as a one-hot label. (b) Knowledge distillation [28]: a teacher provides a (soft) target. A simple yet effective variant [50] combines the predicted label of the teacher with the image label. KD requires a pre-existing teacher that must be stored for inference. (c) In self-distillation, the teacher is obtained from the model itself, typically by model averaging. (d) Co-training involves two distinct models that serves as teacher to each other. (e) Submodel co-training generates on-the-fly two distinct models (amongst 2^L) for each sample, which serves a teacher to each other. We only need to store a single set of weights for the model and optimizer because the random selections $\theta_1 = \mathcal{T}(\theta)$ (resp., θ_2) of submodels allow us to back-propagate on θ.

method	model params weights	model params optimizer	compute*
(a) supervised	×1	×1	×1
(b) KD†	×2	×1	×1.33
(c) mean teacher	×2	×1	×1.33
(d) co-training	×2	×2	×2
(e) Cosub (ours)	×1	×1	×2

*backward pass assumed 2× the complexity of forward
†requires a trained teacher model

training time by dropping residual blocks during training. The initial goal of this approach was to improve the training of deep network. However, there are other implications: in natural language processing, Fan *et al*. show that the counterpart of stochastic depth, namely LayerDrop [20], is effective to reduce the transformer depth on demand. Interestingly, submodels extracted from a model trained with LayerDrop are stronger than those trained with the full target depth with a proper selection strategy for layers.

In our paper, we regard stochastic depth as a regularization technique, but at the same time we adopt the point of view of LayerDrop, *i.e*., an effective way to train a population of submodels that share parameters, or entire layers in our case. From that viewpoint, stochastic depth amounts to training 2^L distinct models. This is different from population training as involved in network space design [41], whose goal is related to network architecture search [18], which aims at optimizing the architecture itself.

More recently, model soups [56] are models obtained by averaging the weights of multiple models finetuned with different hyperparameter configurations. The authors show that it often improves accuracy and robustness. We point out that stochastic depth can be regarded as a special form of model soup over the entire population of submodels that we can instantiate with stochastic depth. We further discuss this relationship in the next section.

3. Submodel co-training

In this section, we present our *cosub* approach, for co-training submodels. We consider a neural network f_θ parameterized by learnable parameters θ.

Submodel instantiation: model augmentation operator. We first define a model augmentation operator $\mathcal{T}$. For a given neural network, it provides a set of parameters $\theta' = \mathcal{T}(\theta, R)$ that allow us to define variations $f_{\theta'}$ of the function f_θ by drawing a random variable R. The model augmentation is such that f_θ and $f_{\theta'}$ share parameters, hence any update on θ' modifies θ and therefore f_θ accordingly. A simple way to define $\mathcal{T}$ is to replace some parameters by zeros, which corresponds to dropout [46]. In this paper, we focus on stochastic depth, which has interesting connections with model averaging [56].

Overview. For a given training sample x within a batch, the training is as follows (see also Figure 1):

1. We first data-augment the image, producing $\hat{x}$.
2. The image is duplicated with the batch, effectively doubling the batch size, which hence contains two identical copies of each augmented image[1].
3. We determine the stochastic depth pattern for each sample according to the target drop-rate τ, which amounts to producing two functions f_{θ_1} and f_{θ_2}. See Section 4 for the details of this procedure.
4. The forward pass proceeds as usual: we compute the soft output labels $y_1 = f_{\theta_1}(\hat{x})$ and $y_2 = f_{\theta_2}(\hat{x})$.
5. We compute the losses and the backwards pass accordingly. Note that the gradients on θ_1 and θ_2 are directly used to update θ, as they are just subsets of θ.

Loss. Each submodel is trained using a weighted average of (i) the standard binary cross-entropy loss obtained from the image label y, and (ii) a binary cross-entropy loss w.r.t. the soft-labels computed for the the same image by the other submodel. The respective weight of the standard binary cross-entropy loss $\mathcal{L}_{\text{label}}$ versus the cosub loss $\mathcal{L}_{\text{cosub}}$ is controlled by the hyper-parameter λ, as

$$\mathcal{L}_{\text{tot}} = \lambda\, \mathcal{L}_{\text{label}} + (1-\lambda)\, \mathcal{L}_{\text{cosub}}. \quad (1)$$

[1]Different data-augmentations did not provide any advantage in our preliminary experiments. We simply duplicate the batch for simplicity.

In details, the loss writes as

$$\mathcal{L}_{\text{tot}} = \lambda \left(\frac{\mathcal{L}(y_1, y) + \mathcal{L}(y_2, y)}{2} \right) + (1-\lambda) \left(\frac{\mathcal{L}(y_1, \mathrm{s_g}(y_2)) + \mathcal{L}(y_2, \mathrm{s_g}(y_1))}{2} \right), \quad (2)$$

where $\mathcal{L}(y, y')$ is either the binary cross-entropy (BCE) or a cross-entropy (CE) loss. Importantly, when applying this loss, we do not back-propagate on the second term y'. This is indicated by the stop-gradient operator $\mathrm{s_g}(\cdot)$.

Discussion. The submodel instantiation provides a model that is, by itself, a valid neural network model. Co-training submodels is a way to enforce that all such submodels produce a consistent output. Amongst these submodels, a very special case is when we retain all residual blocks, which is the model that we primarily intend to use at inference time. Note that Fan *et al*. [20] show, in an NLP context, that submodels extracted from a deeper model are superior, for a given target depth. With proper rescaling of the residual branches, this specific submodel with all blocks activated can be seen as the expectation over all models. This can be shown as follows: let us consider an extra scalar parameter s_l associated with a given residual block $r_l(x)$, which we use as a multiplicative factor on output of each residual: $s_l = 1$ if the residual block r_l is included in the submodel, $s_l = 0$ otherwise. Each submodel is fully parameterized by a binary vector $s = (s_1, \ldots, s_L)$, where L is the total number of layers. Therefore all submodels have the same parameters θ, and only differ by s indicating the zeroed residual blocks. The weight expectation $\bar{\theta}$ is hence

$$\begin{aligned} \bar{\theta} &= \mathbb{E}_{s\in[0,1]^L}[\theta] = \theta, \\ \bar{s} &= \mathbb{E}_{s\in[0,1]^L}[s] = [1-\tau, \ldots, 1-\tau]^\top, \end{aligned} \quad (3)$$

where $\bar{s}$ is the scaling factor used in stochastic depth. Under a uniform distribution of submodels that is obtained when the stochastic drop rate $\tau = 0.5$, the inference-time model is exactly the average of all 2^L models. While stochastic depth is often regarded as a regularization technique, this averaging interpretation relates it to the recent model soup [56].

4. Efficient stochastic depth

We revisit the original stochastic depth [31] formulation in order enable an efficient implementation, which we will release for PyTorch [40]. Instantiating a submodel for a sample amounts to selecting a subset of residual layers of the model and performing training on these. This can be implemented using the stochastic depth (SD) approach, whose objective was initially to improve the training of very deep networks. In stochastic depth, for each sample and each layers of the network, we select whether the layers will be dropped or not with a certain probability τ.

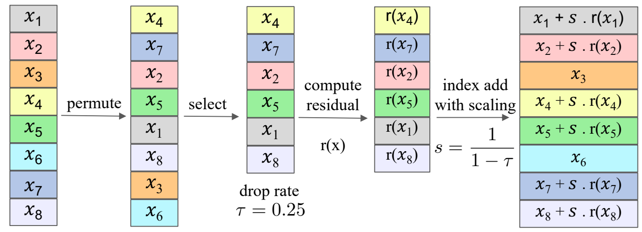


Figure 3. Efficient implementation of stochastic depth, using the permute-select approach. In this example, we drop the residuals for samples 3 and 6, corresponding to a drop rate $\tau = 0.25$. For a drop rate higher than 0.1, the overhead of the approach is negligible in terms of memory and compute.

In practice, *e.g*. in the timm library [54], stochastic depth is implemented by masking with zeros the residuals added for a given sample of a batch. This is not efficient: this naive approach performs the computations for a residual then throws it away, wasting computation.

Our efficient stochastic depth (ESD) approach addresses this issue, saving both memory and compute, and is illustrated in Figure 3. For each layer, given a batch size B and a drop rate τ, we apply the layer to $B_{\text{keep}} = \text{round}(B \times (1-\tau))$ samples in the batch. In contrast to the original version of SD, our efficient version drops a fixed number of samples at each layer, where d is adjusted such that B_{keep} is an integer. In our experiments, this did not have an effect on the final accuracy of the models. Our efficient implementation proceeds as follows: (i) we apply a random permutation of the B samples, then (ii) we select the first B_{keep} samples. We then compute the residual function for the selected subset, then add the result onto the full batch using the built-in `index_add` function and scaling the result by $\frac{1}{1-\tau}$ to have the correct scaling as in the original SD formulation.

Discussion: progressive vs uniform rate. In the original paper [31], stochastic depth drops a layer with a probability that is linearly increasing during the forward pass: layers closer to the output has a higher chance to be dropped. However this strategy is limited with high drop rates, as later layers are excessively dropped. Touvron *et al*. [52] adopt a uniform rate drop per layer. It is as effective as the progressive rate with vision transformers, but it makes it possible to target a drop-rate greater than 0.5 on average. For this reason we adopt an uniform drop rate everywhere.

Our efficient stochastic depth implementation variant works both with uniform and progressive drop-rate. In the case of progressive, the effective batch size is decreasing during to the forward pass. One limitation is that our technique implies a quantization of the drop-rate, which can be problematic for small batch sizes. Therefore one must take care to properly verify that the drop-rate is not too severely quantized, in particular in the progressive case for which the drop-rate can be very high in the last layers. In the supplemental material (Appendix C), we discuss with more details the drop-rate quantization effect.

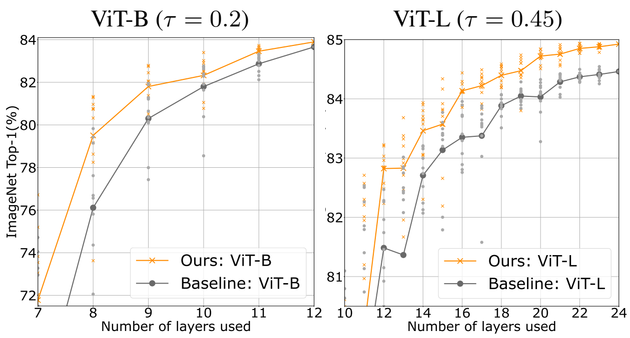


Figure 4. Cosub as population training: each submodel extracted by a transformation $\mathcal{T}$ is a valid neural network. Our cosub strategy can hence be regarded as co-training a large number of subnetworks. We plot the accuracy of the submodels as a function of the number of layers that we preserve. We drop layers with probability $\tau = 0.2$ for ViT-B and $\tau = 0.45$ for ViT-L. On average our method provides a significant boost in performance for the whole population of submodels extracted from the main model.

Model	ViT-S	ViT-M	ViT-B	ViT-L	ViT-H
τ	(0.05)	(0.1)	(0.2)	(0.45)	(0.6)
baseline	81.4	82.5	83.7	84.5	84.9
baseline + cosub	81.5	82.8	83.9	84.9	85.5

Table 1. Comparison of top-1 accuracy with ViT architecture trained with/without cosub at resolution 224 (800 epochs) on Imagenet-1k. The improvement is not significant for the smaller architecture ViT-S. Cosub is gradually more effective when we increase the model size and the SD rate.

5. Experiments

We evaluate our approach on residual architectures [24, 25], since they are readily compatible with the stochastic dropout. We take as our main reference the vanilla Vision Transformer introduced by Dosovitskiy *et al*. [14]. However, as shown in our experiments, our approach is effective with all the residual architectures that we have considered.

5.1. Baseline and training settings

We adopt the state-of-the-art training procedure proposed in the DeiT III paper [51] as baseline for transformers architectures and the training procedure from Wightman *et al*. [55] for the convnets. All hyper-parameters are identical except on Imagenet-21k, where the hyper-parameter τ are adjusted depending on the training setting. We recapitulate the training hyper-parameters in Appendix A.

We additionally adopt and measure the impact of LayerDecay for the fine-tuning when transferring from Imagenet-21k to Imagenet-1k. This method slightly boosts the performance, as discussed later in this section, and in Table 10. This method was adopted in multiple recent papers and in particular for fine-tuning of self-supervised approaches [5, 22], yet the contribution of this fine-tuning ingredient was not quantitatively measured.

Model	τ	Original	Baseline	+cosub	Δ
		Transformers			
ViT-L [14]	0.45	76.5	84.5	84.9	+0.4
CaiT-L24 [52]	0.45	_	83.8	84.4	+0.6
PiT-B [26]	0.25	82.0	83.8	84.1	+0.3
XCiT-L12 [17]	0.20	_	82.6	83.0	+0.4
Swin-B [34]	0.20	83.5	82.9	83.3	+0.4
Swin-L [34]	0.45	_	80.8	84.0	+3.2
		Convnets			
ResNet-50 [24]	0.10	76.2	80.2	80.3	+0.1
ResNet-101 [24]	0.20	77.4	81.8	82.1	+0.4
ResNet-152 [24]	0.30	78.3	82.4	83.1	+0.7
RegNet-16GF [42]	0.30	80.4	82.9	83.8	+0.9

Table 2. Benefit of cosub for different architectures trained from scratch on Imagenet-1k at resolution 224. We report top-1 acc. for the supervised baseline and cosub, as well as results reported in the corresponding papers when available (trained with different settings). We have adjusted the stochastic-depth drop-rate (SD) hyper-parameter for each architecture.

Loss	BCE-soft	BCE-hard	CE-hard
Imagenet-val top-1 accuracy	83.5	83.5	81.9

Table 3. Ablation on the loss for cosub with ViT-H trained at resolution 126×126 on Imagenet-1k during 800 epochs. The training is inherited from DeiT-III, which also uses BCE when training with Imagenet-1k only.

5.2. Empirical analysis of cosub

We perform various ablations on Imagenet-1k to analyse the impact of our training method on the learned networks.

Performance for different model sizes and architectures. Table 1 provides the results obtained by the baseline and cosub when we vary the model size of vision transformers. The stochastic depth coefficient was optimized for the baseline and we keep it unchanged with cosub. As to be expected, our method is almost neutral for small models like ViT-S: +0.1% top-1 accuracy, which is about the standard deviation of measurements. The improvement is increasingly important for larger models, up to a significant improvement of +0.6% top-1 accuracy for ViT-H models.

In Table 2, we show that our approach is beneficial with all architectures that we have tried. We report the results of the original paper, evaluate the performance with our baseline training, and measure the improvement brought by cosub. For almost all architectures, we observe a significant boost in performance. The exception is the ResNet-50, for which cosub improves the top-1 accuracy by only +0.1%, similar to our observation with ViT-S. In Table B.1 in the appendix we present improved results obtained for multiple architectures pre-trained with cosub on Imagenet-21k.

Analysis of submodel performance. With cosub, we sample different subnetworks during training to perform the co-training. We analyse the impact of cosub on the accuracy of the sub networks themselves. In Figure 4 we consider the accuracy of submodels of different size of ViT-B and ViT-L

	Method	ViT-L	ViT-H
(a)	supervised baseline [51]	84.5	84.9
(b)	KD	85.3	85.3
(c)	mean teacher	84.4	83.4
(d)	co-training	82.6	83.1
(e)	cosub	84.9	85.5
(b) + (e)	KD + cosub	85.3	85.7

Table 4. Training strategies with distillation. We compare on Imagenet-1k at resolution 224×224 different approaches involving co- or self-training with distillation. KD, mean teacher and co-training use the same $\lambda = 0.5$ and same hyper-parameters as in Deit-III. Unlike cosub, the mean teacher (c) requires other hyper-parameters for EMA. We perform a small grid-search to adjust this parameters. Note (last row): our approach is complementary with KD, assuming a pre-trained teacher is available beforehand.

λ	0.1	0.3	**0.5**	0.7	0.9	1.0
top1 accuracy	79.05	83.27	**83.55**	83.20	83.04	82.91

Table 5. Ablation of the parameter λ controlling the weight of the co-distillation loss across submodels (Imagenet1k-val, top1-acc). Model ViT-H trained at resolution 126×126 on Imagenet-1k during 800 epochs. $\lambda = 1.0$ corresponds a supervised baseline and $\lambda = 0.5$ corresponds to cosub.

models. Cosub improves the accuracy of the whole population of sub-networks and, in particular, the target network.

Loss formulation. In Table 3 we experiment with different losses for cosub. With Imagenet1k, DeiT-III training uses BCE instead of CE for the main loss. With cosub, BCE is more compatible with the loss of the baseline and, as to be expected, we also observe a better performance with BCE. We have done ablations using hard and soft targets for the cosub loss. The results are similar, therefore by default we keep soft-targets for the cosub loss.

Alternative teacher/student. In Table 4 we report the results obtained with the different distillation or co-training approaches depicted in Figure 2. Other approaches are not effective off-the-shelf, except KD that requires a pre-trained teacher. Our approach is on par with KD (lower for ViT-L, better or ViT-H), and in the last row we show that it even provides a slight boost to combine KD with cosub.

Impact of the cosub loss. The hyper-parameter λ controls the tradeoff between the co-distillation loss and the cross-entropy classification loss. Setting $\lambda = 1$ means that we have a regular supervised training setting, except that (i) we double the batch size by duplicating the image after data augmentation, and (ii) stochastic depth selects different layers for each image copy.

In Table 5, we measure the impact of λ, with all other hyper-parameters being fixed, for ViT-H trained at resolution 126×126 on Imagenet-1k. We observe that the best ratio is to use an equal weighting of the cosub loss and the classic training loss. Using the cosub loss increases the performance by 0.6% Top-1 accuracy on Imagenet-val, which is the typical improvement that we observe for large models.

Model	τ	epochs	Baseline		+cosub	
			val	v2	val	v2
CaiT-M12	0.20	400	83.2	72.9	83.7	73.5
		800	82.9	72.6	83.6	73.1
PiT-B	0.25	400	83.8	73.6	84.1	74.1
		800	82.4	71.9	83.1	72.8
ViT-B	0.20	400	83.1	72.6	83.2	73.1
		800	83.7	73.1	83.9	73.5
		1200	83.3	72.8	_	_
		1600	83.3	73.4	_	_
ViT-H	0.60	400	84.8	75.3	85.0	75.8
		800	84.9	75.6	85.5	76.3

Table 6. We compare ViT models trained with and without cosub on ImageNet-1k only with different number of epochs at resolution 224×224. One can see that cosub is more effective for larger models yielding higher values of the SD hyper-parameter τ. It avoids the early saturation or overfitting of the performance that we typically observe with the baseline when we increase the training time without re-adjusting hyper-parameters. See also Table 5 for a direct comparison with and without the co-distillation loss, and Table 7 for the corresponding training times per epoch.

Training method	model	GPUs used	Memory peak (GB)	Time (min) by epoch
DeiT-III	ViT-L	32	21.4	8
	ViT-H	64	27.6	12
DeiT-III + ESD	ViT-L	32	15.1	9
	ViT-H	64	15.2	11
cosub (with ESD)	ViT-L	32	26.9	16
	ViT-H	64	25.0	17

Table 7. Training times of different models trained at resolution 224×224 with batch size 2048 on Imagenet-1k with DeiT-III and our approach. cosub uses our efficient stochastic depth (ESD), which amortizes the extra memory needed by cosub, especially for the largest models with high stochastic depth values (0.45 for ViT-L, and 0.6 for ViT-H). Timings are indicative and not representative of an optimized selection of the batch size, see Appendix C for measurements with adjusted parameters.

Number of training epochs. In Table 6 we compare results on Imagenet-1k-val and Imagenet-v2 different for architectures trained with and without cosub on Imagenet-1k only at resolution 224×224 with different number of epochs. We observe less overfitting with cosub and longer training schedule. In particular, with bigger architecture like ViT-H, we observe continuous improvement with a longer schedule where the baseline saturates.

Training time. In Table 7 we compare the training costs of cosub and DeiT-III. Thanks to our efficient stochastic depth formulation we maintain a similar memory peak during training. For bigger architectures the gap in training speed between cosub and the baseline is decreasing.

Resolution. In Table 8 we compare different ViT architectures trained with and without cosub at different resolutions on Imagenet-1k. We fine-tune during 20 epochs at resolution 224×224 before evaluation at this resolution. We observe that cosub gives significant improvements across the different resolutions and models.

Model	Resolution	Deit-III Baseline		+cosub	
		val	v2	val	v2
ViT-B	128 × 128	83.5	73.4	83.8	74.0
	192 × 192	83.8	73.6	84.1	74.0
	224 × 224	83.7	73.1	83.9	73.5
ViT-L	128 × 128	84.5	74.7	85.1	75.5
	192 × 192	84.9	75.1	85.2	75.7
	224 × 224	84.5	75.0	84.9	75.6
ViT-H	126 × 126	85.1	75.6	85.6	76.4
	182 × 182	85.1	75.9	85.7	76.6
	224 × 224	84.9	75.6	85.5	76.3

Table 8. Imagenet-1k val and v2 top-1 accuracy of ViT models trained with and without cosub for 800 epochs on Imagenet-1k at different resolutions, followed by finetuning for 20 epochs at resolution 224×224.

resol. →	112	224	336	448	112	224	336	448
↓ model	Imagenet-val				Imagenet-v2			
ViT-S	78.0	83.1	84.6	85.2	66.6	73.7	75.1	76.3
ViT-M	80.6	85.0	86.0	86.3	69.6	76.0	76.8	77.2
ViT-B	82.8	86.3	86.9	87.4	72.1	77.0	77.9	78.3
ViT-L	85.4	87.5	88.1	88.3	75.7	79.1	79.8	80.0
ViT-H	86.2	88.0	-	-	76.9	79.6	-	-
ViT-g	86.5	-	-	-	77.3	-	-	-

Table 9. **Performance of models at different resolutions**. We report the results obtained on Imagenet-val by models of different sizes pre-trained with cosub on Imagenet-21k and fine-tuned on Imagenet-1k. Training schedule: 270 epochs except ViT-g (90 epochs). Except for ViT-S, the results at resolution 336 and 448 were pre-trained on Imagenet-21k at resolution 224 for efficiency reasons. We have not fine-tuned ViT-H and ViT-g at large resolutions since these models are computationally expensive.

Method	Long Training	Layer Decay	Model ViT-S	ViT-B	ViT-L
baseline	✗	✗	82.6	85.2	86.8
cosub	✗	✗	82.5	85.8	87.4
	✗	✓	82.7	86.0	87.5
	✓	✗	82.8	86.0	87.4
	✓	✓	83.1	86.3	87.5

Table 10. We measure the impact of layer-decay during the finetuning on Imagenet-1k for models pre-trained on Imagenet-21k with cosub during 90 epochs (default) and 270 epochs (long training).

Imagenet-21k impact of layer-decay. In Table 10 we compare different number of epochs for the pre-training on Imagenet-21k and the finetuning on Imagenet-1k with and without layer-decay. We observe that both layer-decay and long training bring improvements with cosub.

5.3. Comparisons with the state of the art

Imagenet-1k. In Table 11 we compare architectures trained with cosub with state-of-the-art results from the literature for this architecture. We observe that architectures trained with cosub are very competitive. For instance, for ResNet-152, RegNetY-16GF, PiT-B we improve the best results reported in the literature by more than 0.9% top-1 accuracy on Imagenet-1k-val.

Model	nb params ($\times 10^6$)	throughput (im/s)	FLOPs ($\times 10^9$)	Peak Mem (MB)	Top-1 Acc.	v2 Acc.
"Traditional" ConvNets						
ResNet-50 [24, 55]	25.6	2587	4.1	2182	80.4	68.7
ResNet-101 [24, 55]	44.5	1586	7.9	2269	81.5	70.3
ResNet-101 – cosub	44.5	1586	7.9	2269	82.1	70.8
ResNet-152 [24, 55]	60.2	1122	11.6	2359	82.0	70.6
ResNet-152 – cosub	60.2	1122	11.6	2359	83.1	72.1
RegNetY-8GF [42, 55]	39.2	1158	8.0	3939	82.2	71.1
RegNetY-16GF [42, 50]	83.6	714	16.0	5204	82.9	72.4
RegNetY-16GF – cosub	83.6	714	16.0	5204	83.8	72.8
Vision Transformers derivatives						
Swin-T [34]	28.3	1109	4.5	3345	81.3	69.5
Swin-B [34]	87.8	532	15.4	4695	83.5	_
PiT-S [26]	23.5	1809	2.9	3293	80.9	_
PiT-S – cosub	23.5	1809	2.9	3293	81.3	69.7
PiT-B [26]	73.8	615	12.5	7564	82.0	_
PiT-B – cosub	73.8	615	12.5	7564	84.1	74.1
Vanilla Vision Transformers						
ViT-S [50]	22.0	1891	4.6	987	79.8	68.1
ViT-S – Deit III [51]	22.0	1891	4.6	987	81.4	70.5
ViT-S – cosub	22.0	1891	4.6	987	81.5	70.8
ViT-B [14]	86.6	831	17.5	2078	77.9	–
ViT-B – DeiT [50]	86.6	831	17.5	2078	81.8	71.5
ViT-B – DeiT/distilled	86.6	831	17.5	2078	83.4	73.2
ViT-B – Deit III [51]	86.6	831	17.5	2078	83.8	73.6
ViT-B – cosub	86.6	831	17.5	2078	84.2	74.2
ViT-L – Deit III [51]	304.4	277	61.6	3789	84.9	75.1
ViT-L – cosub	304.4	277	61.6	3789	85.3	75.5
ViT-H – Deit III [51]	632.1	112	167.4	6984	85.2	75.9
ViT-H – cosub	632.1	112	167.4	6984	85.7	76.6

Table 11. **Classification with Imagenet1k training.** We compare with models trained on Imagenet-1k only at resolution 224×224 without self-supervised pre-training (see supp. material for a comparison). We report Top-1 accuracy on Imagenet-1k-val and Imagenet-v2 with different measures of complexity: throughput, FLOPs, number of parameters and peak memory usage. The throughput and peak memory are measured on a single V100-32GB GPU with batch size fixed to 256 and mixed precision.

Imagenet-21k. In Table 12 we compare ViT models pre-trained with cosub on Imagenet-21k and finetuned with cosub on Imagenet-1k with other architectures and our baseline DeiT-III. Our results with vanilla ViT outperform the baseline and are competitive with recent architectures.

Overfitting analysis. As recommended in prior works [51, 55] we perform an overfitting analysis. We evaluate our models trained with codist on Imagenet-1k val and Imagenet-v2 [43]. The results are reported in Figure 5. For ViT, we observe that cosub does not overfit more than the DeiT-III baseline [51]. Our results with other architectures in Table 12 concur with that observation: our results are comparatively stronger on Imagenet-v2 than those reported in the literature for the exact same models.

5.4. Downstream tasks

Semantic segmentation. First we evaluate our ViT models pre-trained on Imagenet with cosub for semantic segmentation on the ADE20k dataset [63]. ADE20k consists of 20k training and 5k validation images with labels over 150 categories. We adopt the training schedule from Swin: 160k iterations with UperNet [57]. At test time we evaluate with

Architecture	nb params ($\times 10^6$)	throughput (im/s)	FLOPs ($\times 10^9$)	peak mem (MB)	top1 acc. val	v2
Convnets						
EfficientNetV2-M [48]	54.1	312	25.0	7127	86.2	75.9
EfficientNetV2-L [48]	118.5	179	53.0	9540	86.8	76.9
ResNet-152 [24, 55]	60.2	1122	11.6	2359	82.0	70.6
ResNet-152 – cosub	60.2	1122	11.6	2359	83.1	73.1
RegnetY16GF – cosub	83.6	714	16.0	5204	84.2	74.7
ConvNeXt-S– cosub	50.2	783	8.7	2218	85.2	76.0
ConvNeXt-B [35]	88.6	563	15.4	3029	85.8	75.6
ConvNeXt-B – cosub	88.6	563	15.4	3029	85.8	76.9
ConvNeXt-L [35]	197.8	344	34.4	4865	86.6	76.6
ConvNeXt-XL [35]	350.2	241	60.9	6951	87.0	77.0
Transformers variations						
Swin-B [34]	87.8	532	15.4	4695	85.2	74.6
Swin-B – cosub	87.8	532	15.4	4695	86.2	77.2
Swin-L [34]	196.5	337	34.5	7350	86.3	76.3
Swin-L – cosub	196.5	337	34.5	7350	87.1	78.1
PiT-B – cosub [26]	73.8	615	12.5	7564	85.8	76.8
XCiT-S12 – cosub [17]	26.2	1373	4.9	1330	84.2	74.9
XCiT-M24 – cosub [17]	84.4	553	16.2	2010	86.5	78.0
XCiT-L24 – cosub [17]	189.0	334	36.1	3315	87.2	77.8
Vanilla Vision Transformers [14, 51]						
ViT-S – cosub	22.0	1891	4.6	987	83.1	73.7
ViT-M – cosub	39.0	1307	8.0	1322	85.0	76.0
ViT-B – DeiT-III	86.6	831	17.6	2078	85.7	76.5
ViT-B – cosub	86.6	831	17.6	2078	86.3	77.0
ViT-L – DeiT-III	304.4	277	61.6	3789	87.0	78.6
ViT-L – cosub	304.4	277	61.6	3789	87.5	79.1
ViT-H – DeiT-III	632.1	112	167.4	6984	87.2	79.2
ViT-H – cosub	632.1	112	167.4	6984	88.0	80.0

Table 12. **Classification with ImageNet-21k pretraining.** We report top-1 accuracy on the validation set of Imagenet1k and Imagenet-V2 with different measures of complexity. The peak memory usage is measured on a single V100-32GB GPU with batch size fixed to 256 and mixed precision. For Swin-L the memory peak is an estimation since we decreased the batch size to 128 to fit in memory. All models are evaluated at resolution 224 except EfficientNetV2 that use resolution 480. ViT are pretrained wit a $\times 3$ schedule, comparable to that used in the best DeiT-III baseline (270 vs. 240 epochs). All other cosub models are pretrained during 90 epochs on Imagenet21k, with 50 epochs of fine-tuning. The τ hyper-parameter is set per model based on prior choices or best guess based on model size.

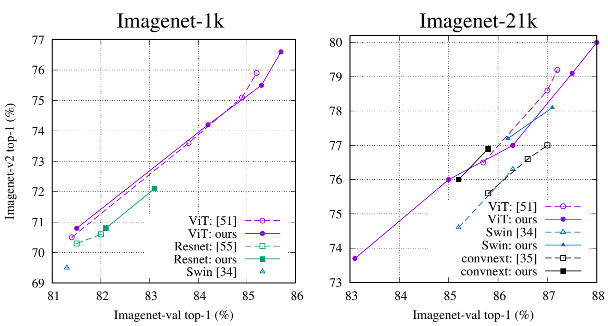


Figure 5. Overfitting measurement: top-1 accuracy on Imagenet-val vs. Imagenet-v2 for models in Tables 11 and 12 pre-trained on Imagenet-1k and Imagenet-21k, respectively. Our cosub ViTs (plain lines and points) do not overfit more than DeiT-III [51] overall. Our Imagenet-21k results for Swin and Convnext generalize much better on v2 than the original models.

a single scale and multi-scale. Our UperNet implementation is the same as in DeiT III [51]. Our results are reported in Table 13. We observe that vanilla ViTs trained with cosub outperform our baseline DeiT-III but also have better FLOPs-accuracy trade-offs than recent architectures.

Backbone	#params ($\times 10^6$)	FLOPs ($\times 10^9$)	Single-scale mIoU	Multi-scale mIoU
DeiT-S	52.0	1099	_	44.0
Swin-T	59.9	945	44.5	46.1
ViT-S – DeiT-III	41.7	588	45.6	46.8
ViT-S – cosub	41.7	588	**47.0**	**48.0**
PatchConvNet-B60	140.6	1258	48.1	48.6
PatchConvNet-B120	229.8	1550	49.4	50.3
MAE ViT-B	127.7	1283	48.1	_
Swin-B	121.0	1188	48.1	49.7
ViT-B – DeiT-III	127.7	1283	49.3	50.2
ViT-B – cosub	127.7	1283	49.3	49.9
ViT-L – DeiT-III	353.6	2231	51.5	52.0
ViT-L – cosub	353.6	2231	**52.5**	**53.1**
PatchConvNet–B60†	140.6	1258	50.5	51.1
Swin-B† (640×640)	121.0	1841	50.0	51.6
ViT-B – DeiT-III †	127.7	1283	53.4	54.1
ViT-B – cosub†	127.7	1283	53.7	54.7
PatchConvNet-L120†	383.7	2086	52.2	52.9
Swin-L† (640×640)	234.0	3230	_	53.5
ViT-L – DeiT-III †	353.6	2231	54.6	55.6
ViT-L – cosub†	353.6	2231	**55.7**	**56.3**

Table 13. **ADE20k semantic segmentation** performance using UperNet [57], in comparable setting as prior works [13, 17, 34]. All models are pre-trained on Imagenet-1k, except bottom models identified with †, which are pre-trained on Imagenet-21k. By default the finetuning resolution on ADE20k is 512×512 except when mentioned otherwise (for Swin).

Model	Cifar-10	Cifar-100	Flowers	Cars	iNat-18	iNat-19
ViT-S – DeiT-III	98.9	90.6	96.4	89.9	67.1	72.7
ViT-B – DeiT-III	99.3	92.5	98.6	93.4	73.6	78.0
ViT-L – DeiT-III	99.3	93.4	**98.9**	**94.5**	75.6	79.3
ViT-S – cosub	99.1	91.7	97.4	93.0	70.1	75.6
ViT-B – cosub	99.1	92.6	98.4	93.5	74.1	78.1
ViT-L – cosub	**99.4**	**93.5**	98.8	**94.5**	**76.2**	**80.1**

Table 14. ViT models pre-trained with cosub or DeiT-III on Imagenet-1k and finetuned on six different target datasets. We note that for small datasets (CIFAR, Flowers, and Cars) our approach is useful for small models but neutral for larger models. The gains are more significant when transferring to the larger iNaturalist-18 and iNaturalist-19 datasets.

Transfer learning. We now measure how the performance improvements observed with cosub translate to other classification problems. For this purpose, we performed transfer learning on the six different datasets used in DeiT-III. Our results are reported Table 14. Our pre-trained and fine-tuned models with cosub generally improve the baseline. The gains are overall more significant on the more challenging datasets like iNaturalist 2018 and iNaturalist 2019.

6. Conclusion

Co-training submodels (cosub) is an effective way to improve existing deep residual networks. It is straightforward to implement, just involving a few lines of code. It does not need a pre-trained teacher, and it only maintains a single set of weights for the model. Extensive experimental results on image classification, transfer learning and semantic segmentation show that cosub is overall extremely effective. It works off-the-shelf and improves the state of the art for various network architectures, including convnets like Regnet.

References

[1] Samira Abnar, Mostafa Dehghani, and Willem Zuidema. Transferring inductive biases through knowledge distillation. *arXiv preprint arXiv:2006.00555*, 2020. 2

[2] Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*, 2018. 2

[3] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Florian Bordes, Pascal Vincent, Armand Joulin, Mike Rabbat, and Nicolas Ballas. Masked siamese networks for label-efficient learning. In *European Conference on Computer Vision*, pages 456–473. Springer, 2022. I

[4] L. Ba and R. Caruana. Do deep nets really need to be deep? In *NeurIPS*, 2014. 2

[5] Hangbo Bao, Li Dong, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 1, 5, I, II

[6] Irwan Bello, W. Fedus, Xianzhi Du, E. D. Cubuk, A. Srinivas, Tsung-Yi Lin, Jonathon Shlens, and Barret Zoph. Revisiting ResNets: Improved training and scaling strategies. *arXiv preprint arXiv:2103.07579*, 2021. II

[7] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59:291–294, 1988. 1

[8] Leo Breiman. Bagging predictors. *Machine learning*, 24(2):123–140, 1996. 2

[9] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *arXiv preprint arXiv:2104.14294*, 2021. 1, 2, II

[10] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020. I

[11] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. 1

[12] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. *arXiv preprint arXiv:2007.11498*, 2020. 1

[13] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Weiming Zhang, Nenghai Yu, Lu Yuan, Dong Chen, and Baining Guo. Cswin transformer: A general vision transformer backbone with cross-shaped windows. *arXiv preprint arXiv:2107.00652*, 2021. 8

[14] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. 5, 7, 8

[15] Alaaeldin El-Nouby, Gautier Izacard, Hugo Touvron, Ivan Laptev, Hervé Jegou, and Edouard Grave. Are large-scale datasets necessary for self-supervised pre-training? *arXiv preprint arXiv:2112.10740*, 2021. 1

[16] Alaaeldin El-Nouby, Natalia Neverova, Ivan Laptev, and Hervé Jégou. Training vision transformers for image retrieval. *arXiv preprint arXiv:2102.05644*, 2021. 1

[17] Alaaeldin El-Nouby, Hugo Touvron, Mathilde Caron, Piotr Bojanowski, Matthijs Douze, Armand Joulin, Ivan Laptev, Natalia Neverova, Gabriel Synnaeve, Jakob Verbeek, et al. Xcit: Cross-covariance image transformers. *arXiv preprint arXiv:2106.09681*, 2021. 5, 8, II

[18] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Neural architecture search: A survey. *Journal of Machine Learning Research*, 20(1):1997–2017, 2019. 3

[19] Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *AISTATS*, 2010. 1

[20] Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*, 2019. ICLR 2020. 2, 3, 4, IV

[21] Rohit Girdhar, Alaaeldin El-Nouby, Mannat Singh, Kalyan Vasudev Alwala, Armand Joulin, and Ishan Misra. Omnimae: Single model masked pretraining on images and videos. *arXiv preprint arXiv:2206.08356*, 2022. I

[22] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Doll'ar, and Ross B. Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021. 1, 5, I, II

[23] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. *arXiv preprint arXiv:1911.05722*, 2019. 1

[24] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Conference on Computer Vision and Pattern Recognition*, 2016. 1, 5, 7, 8, II

[25] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. *arXiv preprint arXiv:1603.05027*, 2016. 1, 5

[26] Byeongho Heo, Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Junsuk Choe, and Seong Joon Oh. Rethinking spatial dimensions of vision transformers. *arXiv preprint arXiv:2103.16302*, 2021. 5, 7, 8, II

[27] G. Hinton and R. Zemel. Autoencoders, minimum description length and Helmholtz free energy. In *NeurIPS*, 1993. 1

[28] Geoffrey E. Hinton, Oriol Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 2, 3

[29] Grant Van Horn, Oisin Mac Aodha, Yang Song, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The iNaturalist species classification and detection dataset. *arXiv preprint arXiv:1707.06642*, 2017. II

[30] Grant Van Horn, Oisin Mac Aodha, Yang Song, Alexander Shepard, Hartwig Adam, Pietro Perona, and Serge J. Belongie. The inaturalist challenge 2018 dataset. *arXiv preprint arXiv:1707.06642*, 2018. II

[31] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q. Weinberger. Deep networks with stochastic depth. In *European Conference on Computer Vision*, 2016. 2, 4, I

[32] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *IEEE Workshop on 3D Representation and Recognition*, 2013. II

[33] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, CIFAR, 2009. II

[34] 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. *arXiv preprint arXiv:2103.14030*, 2021. 5, 7, 8, II

[35] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. *arXiv preprint arXiv:2201.03545*, 2022. 8, II

[36] I. Loshchilov and F. Hutter. Fixing weight decay regularization in adam. *arXiv preprint arXiv:1711.05101*, 2017. I

[37] G. J. McLachlan and D. Peel. *Finite Mixture Models*. Wiley Series in Probability and Statistics. John Wiley & Sons, 2000. 2

[38] M-E. Nilsback and A. Zisserman. Automated flower classification over a large number of classes. In *Proceedings of the Indian Conference on Computer Vision, Graphics and Image Processing*, 2008. II

[39] Maxime Oquab, Leon Bottou, Ivan Laptev, and Josef Sivic. Learning and transferring mid-level image representations using convolutional neural networks. In *Conference on Computer Vision and Pattern Recognition*, 2014. 1

[40] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019. 4

[41] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Conference on Computer Vision and Pattern Recognition*, 2020. 3

[42] Ilija Radosavovic, Raj Prateek Kosaraju, Ross B. Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. *Conference on Computer Vision and Pattern Recognition*, 2020. 5, 7, II

[43] B. Recht, Rebecca Roelofs, L. Schmidt, and V. Shankar. Do ImageNet classifiers generalize to ImageNet? In *International Conference on Machine Learning*, 2019. 7

[44] Adria Ruiz and Jakob Verbeek. Anytime inference with distilled hierarchical neural ensembles. In *AAAI*, 2021. 2

[45] Guocong Song and Wei Chai. Collaborative learning for deep neural networks. *NeurIPS*, 31, 2018. 2

[46] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. In *Journal of Machine Learning Research*, 2014. 1, 2, 3

[47] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. *Conference on Computer Vision and Pattern Recognition*, 2016. 1

[48] Mingxing Tan and Quoc V. Le. Efficientnetv2: Smaller models and faster training. In *International Conference on Machine Learning*, 2021. 8

[49] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Neurips*, 30, 2017. 2

[50] Hugo Touvron, M. Cord, M. Douze, F. Massa, Alexandre Sablayrolles, and H. Jégou. Training data-efficient image transformers & distillation through attention. *International Conference on Machine Learning*, 2021. 2, 3, 7

[51] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit III: Revenge of the vit. *arXiv preprint arXiv:2204.07118*, 2022. 5, 6, 7, 8, I, II

[52] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. *International Conference on Computer Vision*, 2021. 4, 5

[53] Longhui Wei, An Xiao, Lingxi Xie, X. Chen, Xiaopeng Zhang, and Q. Tian. Circumventing outliers of autoaugment with knowledge distillation. In *European Conference on Computer Vision*, 2020. 2

[54] Ross Wightman. Pytorch image models. `https://github.com/rwightman/pytorch-image-models`, 2019. 4

[55] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 5, 7, 8, II

[56] Mitchell Wortsman, Gabriel Ilharco, Samir Ya Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, et al. Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time. In *International Conference on Machine Learning*, 2022. 3, 4

[57] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *European Conference on Computer Vision*, 2018. 7, 8

[58] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? *arXiv preprint arXiv:1411.1792*, 2014. 1

[59] L. Yuan, F. Tay, G. Li, T. Wang, and Jiashi Feng. Revisit knowledge distillation: a teacher-free framework. *Conference on Computer Vision and Pattern Recognition*, 2020. 2

[60] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. *arXiv preprint arXiv:1905.04899*, 2019. 1

[61] Hongyi Zhang, Moustapha Cissé, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017. 1

[62] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Conference on Computer Vision and Pattern Recognition*, pages 4320–4328, 2018. 2

[63] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. *Conference on Computer Vision and Pattern Recognition*, 2017. 7

[64] Xiatian Zhu, Shaogang Gong, et al. Knowledge distillation by on-the-fly native ensemble. *Neurips*, 31, 2018. 2