Masking meets Supervision: A Strong Learning Alliance

Byeongho Heo Taekyung Kim Sangdoo Yun Dongyoon Han
NAVER AI Lab

Abstract

Pre-training with random masked inputs has emerged as a novel trend in self-supervised training. However, supervised learning still faces a challenge in adopting masking augmentations, primarily due to unstable training. In this paper, we propose a novel way to involve masking augmentations dubbed Masked Sub-branch (MaskSub). MaskSub consists of the main-branch and sub-branch, the latter being a part of the former. The main-branch undergoes conventional training recipes, while the sub-branch merits intensive masking augmentations, during training. MaskSub tackles the challenge by mitigating adverse effects through a relaxed loss function similar to a self-distillation loss. Our analysis shows that MaskSub improves performance, with the training loss converging faster than in standard training, which suggests our method stabilizes the training process. We further validate MaskSub across diverse training scenarios and models, including DeiT-III training, MAE finetuning, CLIP finetuning, BERT training, and hierarchical architectures (ResNet and Swin Transformer). Our results show that MaskSub consistently achieves impressive performance gains across all the cases. MaskSub provides a practical and effective solution for introducing additional regularization under various training recipes. Code available at https://github.com/naver-ai/augsub

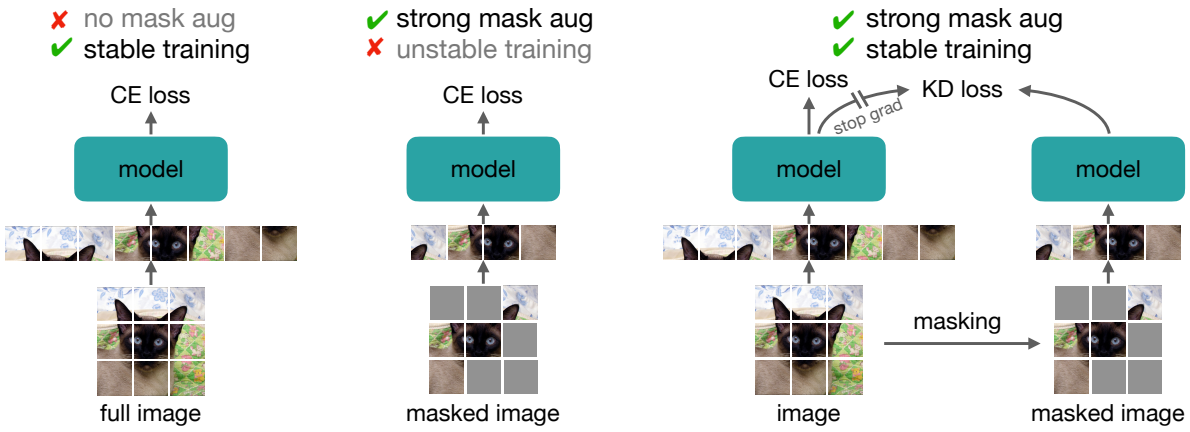
1. Introduction

Supervised learning is the most basic and effective way to train a network to achieve high performance on a target task. To improve supervised learning, diverse regularizations are developed and used as training recipes [36, 37, 44], which represent a group of sophisticatedly tuned regularizations to maximize learning performance. Supervised learning has always held an advantage over self-supervised learning [2, 4] based on the benefit of supervision. However, emergence of Vision Transformer (ViT) [9] and Masked Image Modeling (MIM) [1, 15, 30] is changing this trends. ViT, which lacks inductive bias compared to convolution networks, poses many challenges to generalization performance for supervised learning. On the other hand, MIMs such as MAE [15] rise as an alternative pretraining method for ViT by achieving competitive performance with supervised learning recipes. Although a recent study [37] shows that new supervised learning outperforms MIMs, the gap is insignificant. Thus, MIMs are still a strong competitor of supervised learning methods.

MIM masks random areas of an input image and forces the network to infer the masked area using the remaining area. A representative part of MIM is high mask ratios over 50%. Although MIM also works at small mask ratios, it shows remarkable performance when trained with a high mask ratio. The high mask ratio is a major difference between MIM and supervised learning since this high mask ratio, over 50%, is not beneficial in supervised learning. Supervised learning also has utilized random masking as an augmentation [11, 56], but it significantly degrades performance when the masking ratio is high. In other words, supervised learning is not applicable for strong masking augmentation. We conjecture that it is a major problem of the current supervised learning recipe, and there is room for improvement by enabling strong masking.

Our goal is to improve supervised learning with strong mask augmentation over 50%. To this end, we introduce a novel learning framework using a "sub-branch" alongside the main-branch; throughout this paper, we use the term "sub-branch" to describe a model with dropped inputs. The main-branch uses standard training recipes [37, 44], while the sub-branch utilizes mask augmentation. We name our method as Masked Sub-branch (MaskSub).

We visualize the overview of MaskSub in Figure 1. We consider a high masking ratio over 50% as similar in MAE [15]. Figure 1 (b) shows that applying the strong random masking on the main-branch may lead to degraded performance. In contrast, as in Figure 1 (c), MaskSub leverages the sub-branch for random masking, and the sub-branch receives the training signal from the main-branch similar to the self-distillation [32, 52, 58]. While the random masking technique amplifies the difficulty of the training process, this is counterbalanced by self-distillation loss since the outputs of the main-branch are relaxed and easy-to-learn objective than the ground-truth label. In summary, MaskSub ap-



(a) Standard training (b) Mask augmentation (c) **MaskSub** training

Figure 1. **Overview of Masked Sub-branch (MaskSub).** (a) standard supervised training; (b) masking augmentation training. The masking is applied to the main model, which degrades performance; (c) our **MaskSub** training, which separates the masking from the main model using the sub-branch and relaxes loss with self-distillation. MaskSub substantially improves the state-of-the-art training recipes [37, 44].

plies a mask augmentation separated from the main-branch, utilizing a relaxed loss form.

We analyze MaskSub using 100 epochs training on ImageNet [6]. Without MaskSub, loss convergence speed and corresponding accuracy are significantly degraded when mask augmentation is applied. Conversely, MaskSub mitigates potential harmful effects from additional regularization, leading to a network training process that is even more efficient than standard training procedures. Also, MaskSub is not limited to mask augmentation and can be used for general drop regularizations. As a result, MaskSub is expanded to any random drop regularization without disrupting the convergence of original train loss; we employ three in-network drop-based options to show the applicability: masking [1, 15], dropout [35], and drop-path [10, 19]. Corresponding to each respective regularization strategy, we denote them MaskSub, DropSub, and PathSub. Among the three variants, MaskSub notably exhibits a remarkable performance enhancement, demonstrating the necessity of mask augmentation in supervised learning.

We extensively validate the performance of MaskSub. MaskSub is applied on various state-of-the-art supervised learnings including DeiT-III training [37], MAE finetuning [15], BEiTv2 finetuning [30], CLIP finetuning [8], BERT training [7], ResNet-RSB [44], and Swin transformer [27]. MaskSub demonstrates remarkable performance improvement in all benchmarks. We argue that MaskSub can be regarded as a novel way to utilize regularization for visual recognition.

2. Related Work

Training recipe has been considered an important ingredient in building a high-performance network. He et al. [16] demonstrate that the training recipe significantly influences the network performance. RSB [44] is a representative and high-performance recipe for ResNet. With the emergence of ViT [9], the training recipe for ViT has gained the attention of the field. DeiT [36] shows that ViT can be trained to display strong performance with only ImageNet-1k [6]. DeiT-III [37] is an improved version of DeiT, which applies findings from RSB to DeiT instead of distillation from CNN teacher. It is challenging to implement stronger or additional regularization in existing training recipes. To address this issue, we propose our MaskSub employing sub-branchs.

CoSub [38] introduces a similar concept to ours, utilizing sub-branchs. However, the sub-branch objective differs: while MaskSub aims to stabilize training through additional regularization, CoSub aims to train the sub-branchs by co-training [55]. We regard MaskSub as a more generalized framework since CoSub only considers the drop-path method to employ sub-branchs, whereas MaskSub can cover various drop-based techniques, including masking.

Self-distillation utilizes supervision from a network itself instead of using a teacher. ONE [58] uses a multi-branch ensemble to build superior output for the network and distill ensemble outputs as supervision for each branch. Some studies [32, 52] utilize the early-exit network for self-distillation. Those studies improve performance by using an entire network as a teacher and an early exit network as a student. MaskedKD [34] utilizes masking to reduce computation for knowledge distillation. From a self-distillation perspective, MaskSub presents a new insight into constructing the student model (i.e., sub-branch) from the teacher model (i.e., main-branch) utilizing drop-based techniques. Note that most self-distillation studies are not compatible with recent training recipes [37, 44]. Thus, the general applicability of MaskSub is a notable contribution.

Self- and semi-supervised learning share components

with MaskSub. Contrastive learning incorporates two models with self-distillation loss [3, 12]. Want et al. [41] introduce a double tower with weak and strong augmentation for each model. MAE [15] uses masked image reconstruction as self-supervision, and supervised MAE [26] introduces supervised learning as an additional task for MAE. MAE and supMAE aim to reconstruct masked images using MAE training recipe, rather than supervised learning. In contrast, MaskSub only relies on label-related loss with a supervised learning recipe. In semi-supervised learning, UDA [47] introduces a two-branch framework, similar to the main- and sub-branch in MaskSub. However, MaskSub is more computationally efficient by using masking [15] and removing label-consistency checks for unlabeled data. Also, MaskSub extends the two-branch framework to supervised learning via distillation loss, in contrast to UDA's consistency loss. While these studies share the fundamental concept with MaskSub and inspired our work, the training techniques for supervised learning differ from those in semi- and self-supervised learning. Thus, we argue that MaskSub retains its originality and novelty compared to these studies.

3. Method

We propose our method Masked Sub-branch (MaskSub) with formulation and pseudo-code in Section 3.1. Section 3.2 presents analyses of MaskSub with loss convergence, accuracy, and gradient. In Section 3.3, we introduce variants of MaskSub: DropSub, and PathSub.

3.1. Masked Sub-branch (MaskSub)

The cross-entropy loss with the softmax $\sigma(\mathbf{z}) = e^{z_i}/\sum_j e^{z_j}$ for images $\mathbf{x}_i$ and one-hot labels $\mathbf{y}_i (i \in [1, 2, ..., N])$ in a mini-batch with size N is denoted as

$$-\frac{1}{N}\sum_{i}^{N}\mathbf{y}_i \log\left(\sigma(f_\theta(\mathbf{x}_i|r_{\text{mask}}=0))\right), \tag{1}$$

where f_θ represents the network used for training. r_{mask} means a ratio of masked patches in an input image. Since the masking ratio can be easily changed, we denote it as a condition for network function. Based on the value of r_{mask}, certain network features are dropped with probability r_{mask}. Note that we set the default masking ratio to zero for convenience. Then, loss for masking ratio $r \in [0, 1]$ is

$$-\frac{1}{N}\sum_{i}^{N}\mathbf{y}_i \log\left(\sigma(f_\theta(\mathbf{x}_i|r_{\text{mask}}=r))\right). \tag{2}$$

Typically, a network with mask augmentation is trained with Eq. (2). But, we conjecture that training using Eq. (2) with a high masking ratio (*i.e.* $r \geq 0.5$) may interfere with loss convergence and induce instability in training. To ensure training stability, we utilize the model output of equation Eq. (1), $f_\theta(\mathbf{x}_i|r_{\text{mask}}=0)$, as guidance for masking augmentation $f_\theta(\mathbf{x}_i|r_{\text{mask}}=r)$ instead of $\mathbf{y}_i$. In other words, Eq. (2) is changed as

$$-\frac{1}{N}\sum_{i}^{N}\sigma(f_\theta(\mathbf{x}_i|r_{\text{mask}}=0)) \log\left(\sigma(f_\theta(\mathbf{x}_i|r_{\text{mask}}=r))\right). \tag{3}$$

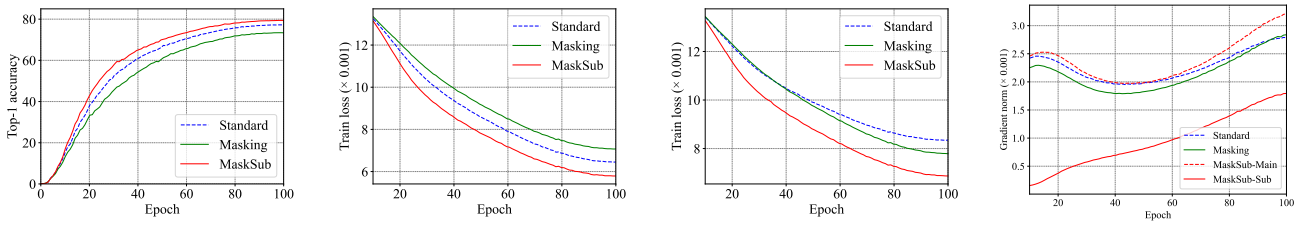
In our Masked Sub-branch (MaskSub), the average of Eq. (1) and Eq. (3) is used as a loss function for the network. We designate $f_\theta(\mathbf{x}_i|r_{\text{mask}}=0)$ as the main-branch and $f_\theta(\mathbf{x}_i|r_{\text{mask}}=r)$ as the sub-branch. This naming convention is employed because a network with masked inputs appears to be a subset of the entire network. In Eq. (3), the main-branch output $f_\theta(\mathbf{x}_i|r_{\text{mask}}=0)$ is used with stop-gradient. Thus, the sub-branch is trained to mimic the main model, but we want the gradient for the main-branch to be independent of the sub-branch. This can be interpreted as self-distillation, where knowledge is transferred from the teacher (main-branch) to the student (sub-branch). Note that MaskSub can easily be expanded to binary cross-entropy loss by replacing the softmax function with the sigmoid function, which is used for recent training recipes [37, 44].

Algorithm 1 MaskSub in PyTorch-style pseudo-code

```
for (x, label) in data_loader:
    o1 = f(x)          # main
    o2 = f(mask(x,r)) # sub (mask ratio: r)
    loss1 = CE(o1, label) / 2
    loss2 = CE(o2, softmax(o1.detach())) / 2
    (loss1+loss2).backward()
    optimizer.step()
```

Algorithm 1 describes PyTorch-style pseudo-code of training with MaskSub. The masking ratio is put into the network input. The gradients are calculated on the main and sub-branch average losses. Note that MaskSub does not use any additional data augmentation, optimizer steps, and network parameters for the sub-branch. We use MAE-style random masking [15], removing masked tokens to reduce computation costs by default. It significantly reduces the training cost of MaskSub. Approximately, MaskSub with 50% masking requires $\times 1.5$ computation to standard training. In practical implementation, we separate the main and sub-branch backward passes utilizing the gradient accumulation of PyTorch. So, VRAM for the main-branch can be released before the sub-branch computation, which eliminates the need for additional VRAM for MaskSub training. We will show the impact of MaskSub on diverse cases in Section 4, including computation analysis in Section 4.5.

MaskSub automatically controls the difficulty of the sub-branch. If the main-branch is close to the ground-truth label, the sub-branch loss aims to attain the ground-truth label under masking. Conversely, if the main-branch fails to converge, the sub-branch loss becomes easy. This difficulty design is inspired by distillation studies [5, 20, 28]. The



(a) Accuracy (b) Train loss (standard) (c) Train loss (masking) (d) Gradient norm

Figure 2. **MaskSub training analysis**. We use 50%-random masking to compare three training settings: standard Eq. (1), masking Eq. (2), and MaskSub. We visualize (a) validation accuracy; (b) train loss without masking; (c) train loss with masking; (d) gradient norm.

distillation becomes difficult when a high-performance network is used as a teacher [5, 28]. An early-stage network is easy, and an end-stage network is challenging [20]. Thus, MaskSub can be considered as a sample-wise masking augmentation that is exclusively applied to images that produce successful output in the main-branch.

3.2. Analysis

We analyze MaskSub with ViT-B [9] for 100 epochs training on ImageNet-1k [6]. Based on DeiT-III [37], we shorten the epoch to 100 epochs and use image resolution 224×224. We compare three settings: standard, masking, and MaskSub. The standard uses Eq. (1) as the training loss, and masking augmentation is not used. For the masking setting, the network is trained with Eq. (2). Note that it is a common practice to use a regularization or an augmentation in supervised learning. We compare those two settings with MaskSub. For analysis, we measured Eq. (1) 'train loss - standard' and Eq. (2) 'train loss - masking'. It shows how losses changed by training setting.

Figure 2 shows loss and accuracy trends in random masking 50% (*i.e.*, $r_{mask} = 0.5$) case. When random masking is applied to training (green), the masking loss (Figure 2c) converges better than the standard (blue). However, it significantly degrades the standard train loss (Figure 2b), resulting in a drop in accuracy (Figure 2a). Regularization over a certain strength often causes malicious effects on standard train loss, which decreases accuracy. As shown in Figure 2b and 2c, MaskSub improves the loss convergence for both losses, original and masking, which brings an improvement in accuracy.

Figure 2d explains the learning pattern between main-branch and sub-branch of MaskSub (Eq. (3)) in the aspect of gradients magnitude for training with random masking 50%. The gradient magnitude from the main-branch (MaskSub-Main) is similar to that of other training. In contrast, gradients from the sub-branch (MaskSub-Sub) have a small magnitude at the early stage. As the learning progresses, the gradients from the sub-branch increase. It shows that MaskSub trains the network following our intention: automatic difficulty control. During the early stage of training, the gradients from the main-branch lead the training. Following the progress of the main-branch training, the sub-branch adaptively increases its gradient magnitude and produces a reasonable amount of gradients at the end of training. In other words, the model training is relaxed from challenging masked inputs at the early stage, while it starts to learn masked input when the original inputs are sufficiently trained. We claim that the automatic difficulty control of MaskSub could be a general solution to introduce strong augmentation for supervised learning.

3.3. Expand to drop regularizations

We design MaskSub for masking augmentation. Due to its simplicity, it can be expanded to drop-based regularizations [11, 19, 35]. In this section, we introduce two variants of MaskSub: DropSub for dropout [35] and PathSub for drop-path [19]. Since the drop-based regularizations easily adjust their strength by controlling drop probability, MaskSub enables the model to learn dropped features without degrading performance at a standard loss, similar to masking augmentation. The performance of MaskSub variants is shown in Section 4.8. Note that detailed experiments with loss convergence for various drop rates are reported in Table A.5 in the Appendix.

DropSub. Dropout [35] is a fundamental drop regularization. Dropout drops random elements of network features with a fixed probability. Since dropout is unrelated to feature structure, every feature element has independent drop probability p_{drop}. DropSub is simply implemented by changing r_{mask} to the dropout probability p_{drop}. Thus, the sub-branch uses strong dropout, while the main-branch follows a standard training recipe. Due to stability issues, dropout is not preferred in recent training recipes [36, 37]. However, DropSub enables strong dropout in ViT training and achieves performance improvement.

PathSub. Drop-path [10, 19] randomly drops a total feature of the network block with a probability p_{path}. PathSub is also implemented by changing r_{mask} to the drop-path probability p_{path}. Drop-path widely used in training recipes [36, 37, 44] to adjust the regularization strength [37]. Thus, unlike previous cases, the main-branch uses the drop-path following the training recipe, and the sub-branch uses a higher drop probability than the main-branch.

4. Experiments

We validate the effectiveness of our Masked Sub-branch (MaskSub) by applying it to diverse training scenarios. We claim MaskSub is an easy plug-in solution for various training recipes. Thus, we strictly follow the original training recipe, including optimizer parameters, learning rate and weight-decay, and regularization parameters. The only difference between baseline and MaskSub is the masking augmentation for the sub-branch. We simply set the masking ratio of MaskSub to 50% across all experiments. In short, MaskSub does not have a hyper-parameter that varies depending on training scenarios.

4.1. Training from scratch (pretraining)

The training recipe in ViTs is a key factor enabling ViT to surpass CNN; thus, the ViT training recipe is an important and active research topic. We use a state-of-the-art ViT training recipe, DeiT-III [37], as our baseline. Enhancing DeiT-III by integrating additional techniques is challenging, so we believe improvements made over DeiT-III would represent a new state-of-the-art in ViT training.

Network	400 epochs		800 epochs	
	DeiT-III	+ MaskSub	DeiT-III	+ MaskSub
ViT-S/16	80.4	**81.1** (+0.7)	81.4	**81.7** (+0.3)
ViT-B/16	83.5	**84.1** (+0.6)	83.8	**84.2** (+0.4)
ViT-L/16	84.5	**85.2** (+0.7)	84.9	**85.3** (+0.4)
ViT-H/14	85.1	**85.7** (+0.6)	85.2	**85.7** (+0.5)

Table 1. **Training from scratch with ViT using the DeiT-III.** MaskSub (50%) is applied to the ViT training [37] on ImageNet-1k. Note that the training settings are identical to the original ones.

Network	Method	Epochs	Top-1 acc.	Cost
ViT-S	DeiT [36]	300	79.8	-
	MAE [15]†	1600	81.4	-
	DeiT-III [37]	800	81.4	×1.0
	CoSub [38]	800	81.5	×2.0
	MaskSub	400	81.1	**×0.75**
	MaskSub	800	**81.7**	×1.5
ViT-B	DeiT [36]	300	81.8	-
	MAE [15]	1600	83.6	-
	SupMAE [26]	400	83.6	-
	DeiT-III [37]	800	83.8	×1.0
	CoSub [38]	800	**84.2**	×2.0
	MaskSub	400	84.1	**×0.75**
	MaskSub	800	**84.2**	×1.5
ViT-L	DeiT-III [37]	800	84.9	×1.0
	CoSub [38]	800	**85.3**	×2.0
	MaskSub	400	85.2	**×0.75**
	MaskSub	800	**85.3**	×1.5
ViT-H	DeiT-III [37]	800	85.2	×1.0
	CoSub [38]	800	**85.7**	×2.0
	MaskSub	400	**85.7**	**×0.75**

Table 2. **Pre-training methods comparison.** We compare DeiT-III [37] + MaskSub with various pre-training methods. MaskSub shows remarkable performances compared to its training cost.

	Epochs	Network	Baseline	+MaskSub
MAE [15] finetuning	100	ViT-B/16	83.6	**83.9** (+0.3)
	50	ViT-L/16	85.9	**86.1** (+0.2)
	50	ViT-H/14	86.9	**87.2** (+0.3)
BEiTv2 [30] finetuning	100	ViT-B/16	85.5	**85.6** (+0.1)
	50	ViT-L/16	87.3	**87.4** (+0.1)
CLIP [33] finetuning	50	ViT-B/16	84.8	**85.2** (+0.4)
	30	ViT-L/14	87.5	**87.8** (+0.3)

Table 3. **ImageNet-1k finetuning.** We report finetuning performance of MAE [15], BEiT v2 [30] and CLIP finetuning [8] with MaskSub (50%). Official weights are used.

ViTs are trained with MaskSub (50%) on 400 and 800 epochs training. The results are shown in Table 1. MaskSub improves performance across all settings. For 400-epochs training, MaskSub improves DeiT-III with substantial margins, which even outperforms 800-epochs trained DeiT-III except for ViT-S/16. MaskSub also demonstrates superior performance when training for 800 epochs. The impact of MaskSub is impressively consistent with larger models like ViT-L/16 and ViT-H/16. It is worth noting that ViT-H + MaskSub (400 epochs) outperforms ViT-H/16 (800 epochs) with +0.5pp gain, even with half the training epochs. Thus, MaskSub is an effective way to improve ViT training.

Table 2 shows the performance and computation cost of MaskSub compared to other pretrainings. In ViT-S and ViT-B, MaskSub outperforms MAE [15] with a reasonable performance gap. Compared to SupMAE [26], MaskSup outperforms under the same epochs. CoSub [38] has comparable performance with MaskSub; however, MaskSub requires less computation costs than CoSub. Thus, we argue that MaskSub outperforms CoSub. More comparisons with CoSub are included in Section 4.5.

4.2. Finetuning

Following the emergence of self-supervised learning [15] and visual-language modeling [33], the significance of finetuning has notably increased. Generally, self-supervised learning, such as MAE [15] and BEiT [1, 30], does not use supervised labels at pretraining, which makes MaskSub inapplicable for pretraining. However, a standard is to evaluate the model's capability using supervised finetuning after pretraining. Thus, we apply our MaskSub (50%) to the finetuning stage to verify the effect of MaskSub on finetuning. Note that we strictly follow the original recipes mentioned below and apply MaskSub (50%) based on it. All finetuning is conducted using officially released pretrained weights.

We utilize three finetuning recipes: MAE [15], BEiT

Network	Method	Top-1 acc.
CLIP-B	Linear probing [33]	80.2
	Finetuning [8]	84.8
	Finetuning [8] + MaskSub	**85.2**
ViT-B	FD-CLIP [43]	84.9
	MaskDistill [31]	85.5
	MVP [42]	84.4
	MILAN [18]	85.4
	CAEv2 [54]	85.5
	BEiTv2 [30]	85.5
	BEiTv2 [30] + MaskSub	**85.6**

Table 4. **Comparison with CLIP-based training on ImageNet-1k.** Our finetuning experiment is close to the state-of-the-art of ViT-B training. MaskSub applied to BEiTv2 [30] fine-tuning outperforms cutting-edge studies on CLIP-based training.

v2 [30], and Finetune CLIP [8]. MAE [15] is a representative method of masked image models (MIM). Since our random masking is motivated by MAE, MaskSub is seamlessly integrated into the MAE finetuning process. BEiT v2 [30] utilizes the pretrained CLIP for MIM and achieves superior performance compared to MAE. Following the masking strategy of BEiT v2 using mask-token, we adjust MaskSub to masking using mask-token from the pretrained weight instead of MAE-style masking. Finetune CLIP [8] is a finetuning recipe for CLIP [33] pretrained weights. MaskSub is applied to finetuning CLIP without change.

Table 3 shows the finetuning results. MaskSub improves the performance of all finetune practices, including large-scale ViT models. This is notable as it shows substantial improvement with a short finetuning phase of fewer than 100 epochs compared to the pretraining period of 1600 epochs. In MAE finetuning, MaskSub improves 0.2 - 0.3pp in all model sizes. MaskSub is also effective on BEiT v2, which utilizes Relative Position Encoding (RPE) [1, 27] and block-masking strategy with mask-tokens. CLIP finetuning also displays that MaskSub achieves substantial improvements. In finetuning CLIP, we report performance at the last epoch rather than selecting the best performance in early epochs. The best performance of finetuning CLIP with MaskSub is the same as the baseline. Table 4 demonstrates the impacts of MaskSub compared to cutting-edge CLIP-based training recipes. It shows that MaskSub improves the performance of state-of-the-art training recipes.

4.3. Hierarchical architecture

We extend experiments to architectures with hierarchical spatial dimensions: ResNet [13] and Swin Transformer [27]. Unlike ViT, which maintains spatial token length for all layers, those networks change the spatial size of features in the middle of layers, requiring a change in masking strategy. We apply MaskSub (50%) to ResNet and

Recipe	Epochs	Network	Baseline	+ MaskSub
RSB A2 [44]	300	ResNet50	79.7	**80.0** (+0.3)
		ResNet101	81.4	**82.1** (+0.7)
		ResNet152	81.8	**82.8** (+1.0)
Swin [27]	300	Swin-T	81.3	**81.4** (+0.1)
		Swin-S	83.0	**83.4** (+0.4)
		Swin-B	83.5	**83.9** (+0.4)

Table 5. **ImageNet-1k with hierarchical architecture.** We show the performance of ResNet [13] and Swin Transformer [27] trained from scratch with MaskSub (50%).

Network	Method	Top-1 acc.
ResNet50	Baseline [14]	76.1
	ResNeXt50 [48] + ONE [58]	78.2
	BYOT [52]	75.2
	Self-Distillation [53]	78.3
	MixSKD [49]	78.8
	RSB [44] + MaskSub	**80.0**
ResNet101	Baseline [14]	77.4
	Self-Distillation [53]	78.9
	RSB [44] + SD-dropout [25]	81.2
	RSB [44] + PS-KD [21]	81.7
	RSB [44] + MaskSub	**82.1**
ResNet152	Baseline [14]	78.3
	PS-KD [21]	79.2
	Self-Distillation [53]	80.6
	SD-dropout [25]	75.5
	RSB [44] + SD-dropout [25]	81.8
	RSB [44] + PS-KD [21]	82.3
	RSB [44] + MaskSub	**82.8**

Table 6. **Comparison with self-distillation methods.** Based on ResNet, we compare MaskSub with self-distillation methods.

Swin Transformer. We simply fill out masked regions with zero pixels for ResNets and replace masked regions with mask-tokens for Swin Transformer. It maintains the spatial structure and enables spatial size reduction of hierarchical architecture. Following the literature [45], we use random masking with the patch size of 32×32. Note that the computation reduction in MAE-style masking does not apply here; therefore, MaskSub costs double the training budget. For ResNet, we use an effective training recipe [44] with 300 epochs. The recipe in the original paper [27] is used for the Swin Transformer training. We strictly follow the training recipes and apply MaskSub without tuning them.

Results are shown in Table 5. MaskSub achieves impressive performance gains with ResNet and Swin Transformer as well. ResNet and Swin are substantially different architectures from ViT. Thus, the result implies that the effectiveness of MaskSub is not limited to ViT architectures and is applicable to hierarchical architectures.

Model	Pretraining + MaskSub	Finetuning + MaskSub	CIFAR100 [24]	CIFAR100 [24]	Flowers [29]	Cars [23]	iNat-18 [39]	iNat-19 [39]
ViT-S/16	-	-	98.8	90.0	94.5	80.9	70.1	76.7
	✔	-	**98.9**	**90.6**	95.2	81.2	70.8	77.0
	✔	✔	98.8	89.9	**98.3**	**92.2**	**71.2**	**77.1**
ViT-B/16	-	-	99.1	91.7	97.5	90.0	73.2	78.5
	✔	-	**99.2**	**91.9**	97.7	90.2	73.6	78.8
	✔	✔	98.8	89.6	**98.7**	**92.8**	**73.9**	**79.1**

Table 7. **Transfer learning.** Table shows transfer learning performance with/without MaskSub. We measure the performance when MaskSub is applied to pretraining and finetuning. The standard deviations over three runs are reported in Appendix.

Architecture	+MaskSub	Epochs	GPU days	Accuracy
ViT-S/16	-	600	22	80.7
	✔	400	22	**81.2** (+0.5)
ViT-B/16	-	600	26	83.7
	✔	400	25	**84.1** (+0.4)
ResNet101	-	600	24	81.5
	✔	300	20	**82.1** (+0.6)
ResNet152	-	600	32	82.0
	✔	300	29	**82.8** (+0.8)

Table 8. **Comparison in the same training budget.** Training has been conducted with NVIDIA V100 8 GPUs. GPU days refer to the number of days required for training when using a V100 GPU.

Method	Accuracy	GPU days
Baseline [37]	83.5	17.3
DataAug [17]	83.5 (+0.0)	36.8 (+113%)
GradAug [50]	83.2 (-0.3)	39.7 (+129%)
CoSub [38]	83.9 (+0.4)	35.3 (+104%)
MaskSub	**84.1**(+0.6)	**25.1** (+45%)

Table 9. **ImageNet-1k Comparison.** The table shows performance and computational costs for ViT-B's 400 epoch training.

	Single-scale mIoU		Multi-scale mIoU	
	DeiT-III	+ MaskSub	DeiT-III	+ MaskSub
ViT-B	48.8	**49.4** (+0.6)	49.7	**50.2** (+0.5)
ViT-L	51.7	**52.2** (+0.5)	52.3	**52.7** (+0.4)

Table 10. **Semantic segmentation on ADE-20k.** UpperNet for ViT backbone is trained with the BEiTv2 segmentation recipe.

4.4. Self-distillation

We compare MaskSub with self-distillation methods. As shown in Table 6, most self-distillations report their performance based on weak and old recipes. Thus, they are less effective with cutting-edge recipes (RSB [44]) or architectures (ViT). Otherwise, MaskSub can be plugged into strong training recipes and achieves state-of-the-art with self-distillation loss. Thus, MaskSub has contributed to practical self-distillation with its broad applicability.

4.5. Training budget

We have shown that MaskSub effectively improves the performance of various architectures. However, MaskSub requires additional computation costs for the sub-branch, which increases training costs. Thus, we analyze MaskSub regarding its training costs to determine if MaskSub could be an effective solution within a limited training budget. We compare MaskSub with training recipes with increased epochs to align with the training budget. The training budget is quantified regarding required GPU days when only a single NVIDIA V100 GPU is used for training. Table 8 shows the results. In ViT training, MaskSub outperforms baseline with $\times 1.5$ epochs setting. Thus, MaskSub is superior to the long epoch training to spend computation costs for training ViT. For ResNet, we compare 300 epochs MaskSub with 600 epochs training recipe RSB [44] A1. MaskSub outperforms 600 epochs training recipes in ResNet101 and ResNet152. Consequently, the results show that MaskSub is an effective way to improve training, even considering computation costs for the sub-branch.

We compare MaskSub with other training methods: DataAugment [17], GradAug [50], and CoSub [38]. DataAugment [17] uses doubled data augmentations for the same image, which is similar to contrastive learning [2, 4]. GradAug [50] utilizes a network pruning [51] to build sub-network. CoSub introduces a sub-network based on drop-path [19] and uses the sub-network as mutual learning [55]. Table 9 shows the 400-epochs training from scratch result. Note that GradAug in Table 9 is a 200-epochs training result to adjust computation cost similar to other methods. All augmentation methods require additional computation costs. In particular, GradAug spends almost 300% of additional training costs compared to original training. On the other hand, our MaskSub only requires a small amount of extra costs (below 50%), which is a remarkable advantage in training. With the smallest computation, our MaskSub achieves substantial performance improvements. MaskSub performs superior to CoSub in all cases.

Model	+MaskSub	MNLI	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
BERT [7]	-	84.1	87.5	91.0	91.6	54.7	**87.0**	88.5	62.8	80.9
base	✔	**84.5**	**87.7**	**91.3**	**91.9**	**58.3**	86.8	**89.2**	**63.2**	**81.6**
BERT [7]	-	86.8	88.2	92.3	93.8	63.3	**89.3**	**92.0**	**69.7**	84.4
large	✔	**87.1**	**89.0**	**92.7**	**94.0**	**65.2**	88.6	91.5	69.3	**84.7**

Table 11. **GLUE [40] benchmark with BERT [7].** We apply MaskSub on GLUE benchmark to validate the effect of MaskSub on language model fine-tuning. MaskSub effectively improves BERT finetuning performance.

Training method	ImageNet-1k Zero-shot acc.
CLIP [33]	33.5
CLIP [33] + Masking	29.8 (-3.7)
CLIP [33] + MaskSub	**37.6** (+4.1)

Table 12. **MaskSub on CLIP pretraining** with ViT-B/32. We apply MaskSub to CLIP, vision and language, pre-training process. MaskSub is effective for CLIP pre-training.

4.6. Transfer learning

Improvement in pretraining can boost the performance of downstream tasks [22]. We measure the transfer learning performance of MaskSub using 800 epochs pretrained weight from Table 1. CIFAR-10 [24], CIFAR-100 [24], Oxford Flowers-102 [29], Stanford Cars [23] and iNaturalist [39] are used for finetuning datasets. We use the AdamW training recipe [37] and also evaluate performance when MaskSub (50%) is applied to the finetuning process. Table 7 shows the results. The backbone pretrained with MaskSub consistently outperforms the DeiT-III backbone across all cases. Moreover, when MaskSub is applied to the finetuning, it further boosts performance except CIFAR [24].

We verify transfer learning to semantic segmentation task on ADE-20k [57]. We train UperNet [46] training recipe [30] and utilize pretrained weight from Table 1. Table 10 shows the segmentation results of single-scale and multi-scale evaluations. On both evaluations, the backbone pretrained with MaskSub demonstrates superior performance, consistent for ViT-B and ViT-L.

4.7. Beyond vision domain

MaskSub can be extended to domains beyond images, as long as the masking is applicable. Thus, we apply MaskSub to two additional tasks beyond the image domain: GLUE [40] benchmark and CLIP [33] pretraining. The first task is a text-classification benchmark GLUE [40]. We use BERT [7] as a pretrained model and apply MaskSub with 15% masking following the masking ratio of BERT. As shown in Table 11, MaskSub improves text-classification performance. MaskSub is also applied to CLIP [33] pretraining. Table 12 shows the results. CLIP trained with MaskSub (50%) shows improved zero-shot performance.

Architecture	Baseline	MaskSub	DropSub	PathSub
ViT-S/16	80.4	**81.1** (+0.7)	80.6 (+0.2)	80.8 (+0.4)
ViT-B/16	83.5	**84.1** (+0.6)	83.8 (+0.3)	83.8 (+0.3)
Computation	×1.0	×1.5	×2.0	×2.0

Table 13. **Comparison of MaskSub variants.** We validate drop-based variants of MaskSub. The sub-branch training improves performance with other drop-methods. But, MaskSub shows the best improvement with the smallest computations.

Experimental details are in Appendix. These results verify that MaskSub has remarkable impacts not only on the vision but also on the language and vision&language domain.

4.8. Extending to drop regularizations

In Section 3.3, we expand MaskSub with drop regularizations [19, 35]. We validate the performance of MaskSub variants on a 400 epochs training with Deit-III. We use masking [15] (50%), dropout [35] (0.2), and drop-path [19] (baseline + 0.1) for MaskSub, DropSub, and PathSub, respectively. Table 13 shows the results. Variants of MaskSub outperform the baseline. Among the three, MaskSub shows the best performance. Also, MaskSub has the lowest computation costs due to MAE [15]-style computation reduction. Thus, we conclude that MaskSub (50%) is the best in practice compared to variants with drop regularizations. Note that Table A.5 in Appendix includes more results.

5. Conclusion

In this work, we have presented a new way to introduce masking augmentation to supervised learning. Our method, Masked Sub-branch (MaskSub), is designed to leverage masking augmentation within a sub-branch, which is separated from main training and uses a relaxed loss function. Our extensive analysis reveals that MaskSub effectively mitigates malicious effects of heavy masking while accelerating the convergence, yielding superior performance. We verify MaskSub on various training recipes with diverse architecture. Notably, MaskSub demonstrates impressive performance improvements across various scenarios. We claim that MaskSub is a substantial advancement in training recipes and contributes to using augmentations.

References

[1] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 1, 2, 5, 6

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9650–9660, 2021. 1, 7

[3] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15750–15758, 2021. 3

[4] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9620–9629, 2021. 1, 7

[5] Jang Hyun Cho and Bharath Hariharan. On the efficacy of knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4794–4802, 2019. 3, 4

[6] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 248–255. Ieee, 2009. 2, 4

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 2, 8

[8] Xiaoyi Dong, Jianmin Bao, Ting Zhang, Dongdong Chen, Shuyang Gu, Weiming Zhang, Lu Yuan, Dong Chen, Fang Wen, and Nenghai Yu. Clip itself is a strong fine-tuner: Achieving 85.7% and 88.0% top-1 accuracy with vit-b and vit-l on imagenet. *arXiv preprint arXiv:2212.06138*, 2022. 2, 5, 6

[9] 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. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2, 4

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

[11] Golnaz Ghiasi, Tsung-Yi Lin, and Quoc V Le. Dropblock: A regularization method for convolutional networks. *Advances in neural information processing systems*, 31, 2018. 1, 4

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 3

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[15] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 1, 2, 3, 5, 6, 8

[16] Tong He, Zhi Zhang, Hang Zhang, Zhongyue Zhang, Junyuan Xie, and Mu Li. Bag of tricks for image classification with convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 558–567, 2019. 2

[17] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8129–8138, 2020. 7

[18] Zejiang Hou, Fei Sun, Yen-Kuang Chen, Yuan Xie, and Sun-Yuan Kung. Milan: Masked image pretraining on language assisted representation. *arXiv preprint arXiv:2208.06049*, 2022. 6

[19] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *Proceedings of the European Conference on Computer Vision*, pages 646–661. Springer, 2016. 2, 4, 7, 8

[20] Xiao Jin, Baoyun Peng, Yichao Wu, Yu Liu, Jiaheng Liu, Ding Liang, Junjie Yan, and Xiaolin Hu. Knowledge distillation via route constrained optimization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1345–1354, 2019. 3, 4

[21] Kyungyul Kim, ByeongMoon Ji, Doyoung Yoon, and Sangheum Hwang. Self-knowledge distillation with progressive refinement of targets. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6567–6576, 2021. 6

[22] Simon Kornblith, Jonathon Shlens, and Quoc V Le. Do better imagenet models transfer better? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2661–2671, 2019. 8

[23] Jonathan Krause, Michael Stark, Jia Deng, and Li Fei-Fei. 3d object representations for fine-grained categorization. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 554–561, 2013. 7, 8

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 7, 8

[25] Hyoje Lee, Yeachan Park, Hyun Seo, and Myungjoo Kang. Self-knowledge distillation via dropout. *Computer Vision and Image Understanding*, 233:103720, 2023. 6

[26] Feng Liang, Yangguang Li, and Diana Marculescu. Supmae: Supervised masked autoencoders are efficient vision learners. *arXiv preprint arXiv:2205.14540*, 2022. 3, 5

[27] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer:

Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 2, 6

[28] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Ang Li, Nir Levine, Akihiro Matsukawa, and Hassan Ghasemzadeh. Improved knowledge distillation via teacher assistant. In *Proceedings of the AAAI conference on artificial intelligence*, pages 5191–5198, 2020. 3, 4

[29] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729. IEEE, 2008. 7, 8

[30] Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. Beit v2: Masked image modeling with vector-quantized visual tokenizers. *arXiv preprint arXiv:2208.06366*, 2022. 1, 2, 5, 6, 8

[31] Zhiliang Peng, Li Dong, Hangbo Bao, Qixiang Ye, and Furu Wei. A unified view of masked image modeling. *arXiv preprint arXiv:2210.10615*, 2022. 6

[32] Mary Phuong and Christoph H Lampert. Distillation-based training for multi-exit architectures. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1355–1364, 2019. 1, 2

[33] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, pages 8748–8763. PMLR, 2021. 5, 6, 8

[34] Seungwoo Son, Namhoon Lee, and Jaeho Lee. Maskedkd: Efficient distillation of vision transformers with masked images. *arXiv preprint arXiv:2302.10494*, 2023. 2

[35] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014. 2, 4, 8

[36] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 1, 2, 4, 5

[37] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *Proceedings of the European Conference on Computer Vision*, pages 516–533. Springer, 2022. 1, 2, 3, 4, 5, 7, 8

[38] Hugo Touvron, Matthieu Cord, Maxime Oquab, Piotr Bojanowski, Jakob Verbeek, and Hervé Jégou. Co-training 2^L submodels for visual recognition. *arXiv preprint arXiv:2212.04884*, 2022. 2, 5, 7

[39] Grant Van Horn, Oisin Mac Aodha, Yang Song, Yin Cui, Chen Sun, Alex Shepard, Hartwig Adam, Pietro Perona, and Serge Belongie. The inaturalist species classification and detection dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 8769–8778, 2018. 7, 8

[40] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 8

[41] Xiao Wang, Haoqi Fan, Yuandong Tian, Daisuke Kihara, and Xinlei Chen. On the importance of asymmetry for siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16570–16579, 2022. 3

[42] Longhui Wei, Lingxi Xie, Wengang Zhou, Houqiang Li, and Qi Tian. Mvp: Multimodality-guided visual pre-training. In *European Conference on Computer Vision*, pages 337–353. Springer, 2022. 6

[43] Yixuan Wei, Han Hu, Zhenda Xie, Zheng Zhang, Yue Cao, Jianmin Bao, Dong Chen, and Baining Guo. Contrastive learning rivals masked image modeling in fine-tuning via feature distillation. *arXiv preprint arXiv:2205.14141*, 2022. 6

[44] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 1, 2, 3, 4, 6, 7

[45] Sanghyun Woo, Shoubhik Debnath, Ronghang Hu, Xinlei Chen, Zhuang Liu, In So Kweon, and Saining Xie. Convnext v2: Co-designing and scaling convnets with masked autoencoders. *arXiv preprint arXiv:2301.00808*, 2023. 6

[46] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 418–434, 2018. 8

[47] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *Advances in neural information processing systems*, 33:6256–6268, 2020. 3

[48] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017. 6

[49] Chuanguang Yang, Zhulin An, Helong Zhou, Linhang Cai, Xiang Zhi, Jiwen Wu, Yongjun Xu, and Qian Zhang. Mixskd: Self-knowledge distillation from mixup for image recognition. In *European Conference on Computer Vision*, pages 534–551. Springer, 2022. 6

[50] Taojiannan Yang, Sijie Zhu, and Chen Chen. Gradaug: A new regularization method for deep neural networks. *Advances in Neural Information Processing Systems*, 33: 14207–14218, 2020. 7

[51] Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. *arXiv preprint arXiv:1812.08928*, 2018. 7

[52] Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722, 2019. 1, 2, 6

[53] Linfeng Zhang, Chenglong Bao, and Kaisheng Ma. Self-distillation: Towards efficient and compact neural networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(8):4388–4403, 2021. 6

[54] Xinyu Zhang, Jiahui Chen, Junkun Yuan, Qiang Chen, Jian Wang, Xiaodi Wang, Shumin Han, Xiaokang Chen, Jimin Pi, Kun Yao, et al. Cae v2: Context autoencoder with clip latent alignment. *Transactions on Machine Learning Research*, 2023. 6

[55] Ying Zhang, Tao Xiang, Timothy M Hospedales, and Huchuan Lu. Deep mutual learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4320–4328, 2018. 2, 7

[56] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13001–13008, 2020. 1

[57] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 633–641, 2017. 8

[58] Xiatian Zhu, Shaogang Gong, et al. Knowledge distillation by on-the-fly native ensemble. *Advances in neural information processing systems*, 31, 2018. 1, 2, 6