Masking meets Supervision: A Strong Learning Alliance

–Appendix–

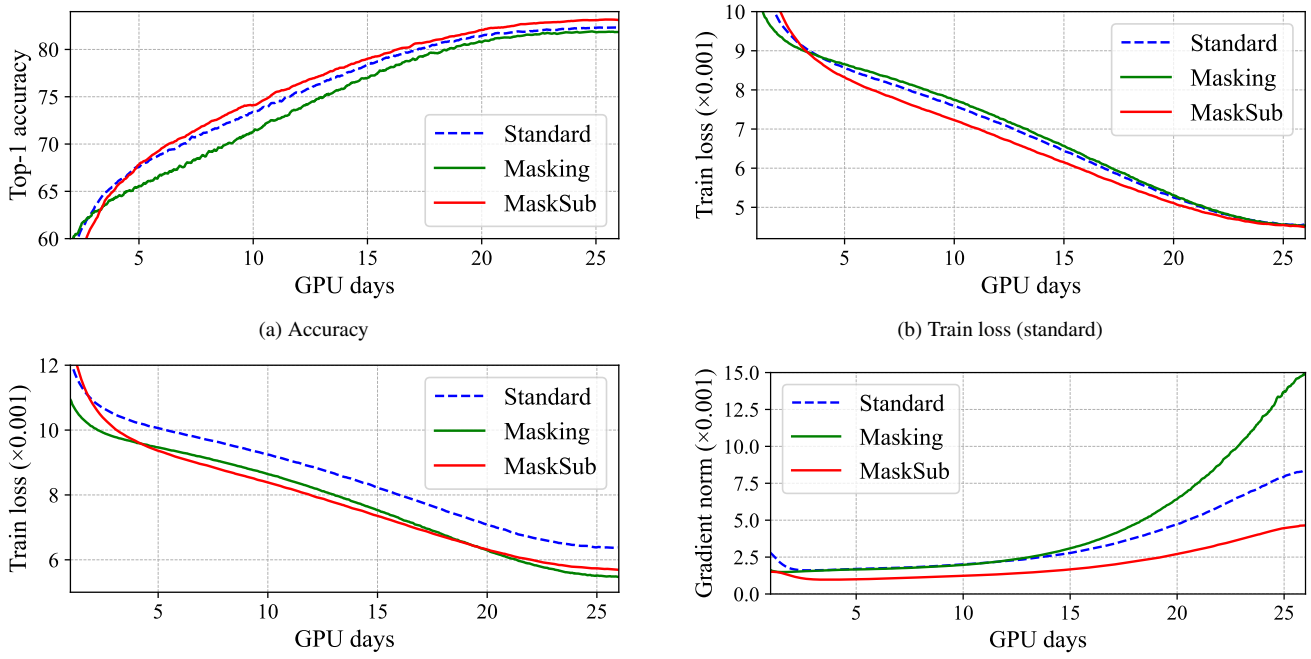
A. Experiments (cont'd)

A.1. Training budget

We report an extensive comparison of the training budget in addition to Table 2 and 8. Table A.1 shows the performance improvements of MaskSub compared to training recipes with increased epochs to have the same training costs as MaskSub. In most cases, MaskSub outperforms its counterparts even with the same training computation costs. The results imply that MaskSub is a better solution than increasing training epochs to improve model performance. Note that the performance of MAE finetuning is lower than that of the original epoch since it utilizes the benefits of early stop training to prevent overfitting. Table A.2 presents a comparison with other training methods using additional augmentations in MAE [9] fine-tuning task. Similar to Table 8, MaskSub shows the impressive trade-off between performance and computation costs, even in the fine-tuning task. Fig. A.1 shows training analysis in Fig. 2 with the same training budget setting. We use GPU days as an x-axis of analysis. MaskSub is effective for loss convergence and accuracy, even considering the additional training budget.

Table A.1. **Comparison under the equivalent training budget.** All trainings are conducted on NVIDIA V100 8 GPUs. GPU days refer to the number of days required for training when using a single V100 GPU.

	Architecture	Training recipe	+MaskSub	Epochs	GPU days	Accuracy
DeiT-III Training	ViT-S/16	DeiT-III	-	600	22	80.7
			✔	400	22	**81.2** (+0.5)
		DeiT-III	-	1200	45	81.6
			✔	800	44	**81.7** (+0.1)
	ViT-B/16	DeiT-III	-	600	26	83.7
			✔	400	25	**84.1** (+0.4)
		DeiT-III	-	1200	52	83.8
			✔	800	50	**84.2** (+0.4)
MAE Finetuning	ViT-B/16	MAE Finetune	-	150	9	83.5
			✔	100	9	**83.9** (+0.4)
	ViT-L/16	MAE Finetune	-	75	14	85.5
			✔	50	14	**86.1** (+0.6)
ResNet Training	ResNet50	RSB A1	-	600	22	80.4
		RSB A2	✔	300	14	**80.0** (-0.4)
	ResNet101	RSB A1	-	600	24	81.5
		RSB A2	✔	300	20	**82.1** (+0.6)
	ResNet152	RSB A1	-	600	32	82.0
		RSB A2	✔	300	29	**82.8** (+0.8)



(a) Accuracy

(b) Train loss (standard)

(c) Train loss (masking)

(d) Gradient norm

Figure A.1. **MaskSub training analysis for training budget.** The figures show training analysis experiments in Fig. 2 with the training budget (GPU days) as x-axis. Even considering its additional training budgets, MaskSub effectively improves convergence and accuracy.

Table A.2. **MAE finetuning comparison.** The table displays performances and GPU costs for MAE ViT-B fine-tuning.

Method	Accuracy	GPU days
Baseline [21]	83.6	6.0
DataAug [12]	83.1 (-0.5)	11.8 (+97%)
GradAug [26]	**84.0**(+0.4)	23.4 (+290%)
CoSub [22]	83.8 (+0.2)	11.0 (+83%)
MaskSub	83.9 (+0.3)	**8.6** (+43%)

A.2. Downstream tasks

Transfer learning performances are presented in Table 7 in the main paper.

Semantic segmentation results are included in Table 10 in the main paper.

Object detection and instance segmentation. We utilize Cascaded Mask R-CNN [2] with the ViT backbones [14] for MS COCO [15], which conducts object detection and instance segmentation simultaneously. ViTDet [14] is used as a training recipe for this experiment. Table A.3 shows the results. The metric AP^{box} quantifies the performance in object detection, while AP^{mask} provides performance in instance segmentation. In both measures, the backbone pretrained with MaskSub outperforms the DeiT-III backbone.

Table A.3. **Detection and instance segmentation on MS COCO.** Cascaded Mask R-CNN with ViT-B is used.

	AP^{box}	AP^{mask}
DeiT-III	50.7	43.6
+MaskSub	**50.9** (+0.2)	**43.9** (+0.3)

A.3. Robustness

We evaluate the impact of MaskSub in various robustness benchmarks. We use models trained for 800 epochs in Table 1. Table A.4 shows the results. ViT models trained with MaskSub demonstrate superior performance in in-distribution metrics and all out-of-distribution metrics. Specifically, MaskSub outperforms each baseline in natural adversarial examples (IN-A [11]), objects in different styles and textures (IN-R [10]), controls in rotation, background, and viewpoints (ObjNet [1]), and detecting spurious correlations with background [8] (SI-size, SI-loc, and SI-rot). The results demonstrate that the improvement of MaskSub is not limited to the ImageNet-1k validation and has been verified across various robustness metrics.

Table A.4. **Robustness benchmark.** Table shows the robustness benchmark for ViT pretrained with/without MaskSub. ↑ means higher score is better robustness, while ↓ indicates lower score is better.

Model	+MaskSub	IN-1k(↑)	IN-V2(↑)	IN-Real(↑)	IN-A(↑)	IN-R(↑)	IN-C(↓)	ObjNet(↑)	SI-size(↑)	SI-loc(↑)	SI-rot(↑)
ViT-S	-	81.4	70.1	87.0	23.4	46.4	44.8	32.6	55.0	39.8	37.8
	✔	**81.7**	**71.0**	**87.4**	**26.9**	**47.2**	**43.6**	**33.5**	**56.7**	**42.5**	**39.9**
ViT-B	-	83.8	73.4	88.2	36.8	54.1	38.1	35.7	58.0	42.7	41.5
	✔	**84.2**	**74.0**	**88.6**	**41.9**	**54.4**	**36.9**	**37.2**	**59.0**	**44.8**	**43.3**
ViT-L	-	84.9	74.8	88.8	45.3	57.4	33.9	38.8	59.8	46.5	45.0
	✔	**85.3**	**75.8**	**89.2**	**51.1**	**58.5**	**32.4**	**40.0**	**60.2**	**46.8**	**45.9**
ViT-H	-	85.2	75.7	89.2	51.9	58.8	32.8	40.1	61.9	49.0	46.8
	✔	**85.7**	**76.5**	**89.6**	**58.3**	**59.9**	**31.2**	**41.7**	**62.4**	**50.1**	**48.4**

A.4. Extending to drop regularizations

To generalize MaskSub to drop regularizations, we demonstrate MaskSub with drop-out [19] (DropSub) and drop-path [13] (PathSub) in Table 13. In this section, we present additional analysis to verify MaskSub variants. The experiment setup is the same as in Section 3.2. The results are shown in Table A.5. We validate three MaskSub methods for three kinds of regularizations: random masking [9], drop-out [19], and drop-path [13]. "Original" means the original training recipe trained with Eq. 1, and none of the drop regularizations is used. "Single model" shows the performance when the network is trained with Eq. 2. We compare those common practices with "MaskSub variants", models trained with Eq. 1 and 3. For analysis, we measured Eq. 1 "train loss (original)" and Eq. 2 "train loss (drop)" regardless of the loss used for training. It shows how losses changed by training setting. The results demonstrate that MaskSub improves training in all three cases. In all cases, MaskSub improves original and drop loss convergence, which is connected to superior accuracy compared to original training.

Table A.5. **Analysis with drop regularizations.** This table shows 100 epochs of the ViT-B performance trained with drop regularizations. Note that training loss scale 10^{-3} is omitted for simplicity. The table presents the average values over three separate runs, and the standard deviations are reported in Table A.9. We observe that MaskSub consistently enhances the baselines significantly and also decreases the training loss at the main branch. This suggests that our method effectively facilatates training with regularizations.

		Single model			MaskSub variants		
	Drop ratio	Accuracy	Train loss (original)	Train loss (drop)	Accuracy	Train loss (original)	Train loss (drop)
Original	-	77.4	6.42	-	-	-	-
Masking [9]	25%	76.3	6.60	6.96	79.0	5.89	6.38
	50%	73.8	7.02	7.77	79.4	5.81	6.89
	75%	67.3	8.08	9.27	79.2	5.84	8.15
Drop-out [19]	0.1	76.1	6.60	6.87	79.1	5.88	6.32
	0.2	74.1	6.95	7.34	79.1	5.82	6.57
	0.3	71.6	8.34	7.79	79.1	5.84	6.90
Drop-path [13]	0.1	77.4	6.42	6.42	78.4	6.11	6.11
	0.2	74.9	6.74	7.19	78.7	5.91	6.48
	0.3	71.6	7.31	8.04	78.8	5.87	7.02

A.5. Ablation studies

We conduct ablation studies of MaskSub using DeiT-III training settings to validate the role of KD loss and the effects of hyperparameters. Table A.6 shows the impacts of KD loss in MaskSub. Performance improvement by MaskSub is significantly limited when KD loss is not used. Table **??** and Table **??** show analysis for masking ratio and loss weights, respectively. Note that we train for 100 epochs with the DeiT-III setting, and the baseline without MaskSub is 77.4. In Table **??**, we test additional ways to improve MaskSub's performance: multiple sub-branch and photometric augmentation on sub-branch. We train ViT-B/16 with 400 epochs DeiT-III setting with MaskSub variants. Single sub-branch shows the best performance with the smallest computation budget. Color-jittering improves MaskSub with negligible cost increases, but Gaussian blur is ineffective.

Table A.6. **Effect of KD loss.** We analyze removing KD loss of the sub-model. We use the DeiT-III 400 epochs setting as in Table 1.

	Baseline	MaskSub	
		w/o KD loss	with KD loss
ViT-S/16	80.4	80.5 (+0.1)	**81.1** (+0.7)
ViT-B/16	83.5	83.6 (+0.1)	**84.1** (+0.6)

A.6. Reporting mean and standard deviation

We provide mean and standard deviation for experiments using different random seeds. The values presented in this section result from three independent runs with different seeds.
Mean and standard deviation values for transfer learning in Table 7 are shown in Table A.7. MaskSub demonstrates meaningful performance gains, which surpass the standard deviation of performance. Table A.8 presents short training (300 epochs) results. MaskSub shows substantial improvements when applied to pretraining and finetuning processes. Table A.10 shows 400 epochs training with DeiT-III [21], which is reported in Table 13 of the paper. MaskSub variants improve the performance of ViT training. Compared to the variants, MaskSub demonstrates the best performance. Table A.9 shows the mean and standard deviation values for Table A.5. Table A.9 shows the superiority of our MaskSub in additional regularization training.

Table A.7. **Mean and std for transfer learning to small scale datasets.** The table shows 'mean ± std' values for transfer learning performance with/without MaskSub. We measure the performance when MaskSub is applied to pretraining and finetuning.

Model	Pretraining + MaskSub	Finetuning + MaskSub	CIFAR10	CIFAR100	Flowers	Cars	iNat-18	iNat-19
ViT-S	-	-	98.83 ± 0.05	89.96 ± 0.15	94.54 ± 1.71	80.86 ± 0.71	70.12 ± 0.13	76.69 ± 0.56
	✔	-	98.88 ± 0.09	90.63 ± 0.09	95.19 ± 1.95	81.23 ± 0.73	70.82 ± 0.03	77.00 ± 0.21
	✔	✔	98.77 ± 0.05	89.87 ± 0.17	98.25 ± 0.51	92.17 ± 0.14	71.17 ± 0.21	77.12 ± 0.48
ViT-B	-	-	99.07 ± 0.05	91.69 ± 0.15	97.52 ± 0.51	90.05 ± 0.24	73.16 ± 0.05	78.49 ± 0.62
	✔	-	99.19 ± 0.03	91.89 ± 0.04	97.73 ± 0.30	90.18 ± 0.12	73.61 ± 0.08	78.77 ± 0.05
	✔	✔	98.82 ± 0.03	89.55 ± 0.05	98.68 ± 0.16	92.77 ± 0.09	73.88 ± 0.12	79.07 ± 0.55

Table A.8. **Transfer learning at short (300 epochs) training.** The table shows 'mean ± std' values for transfer learning at 300 epochs. We measure the performance when MaskSub is applied to pretraining and finetuning.

Model	Pretraining + MaskSub	Finetuning + MaskSub	CIFAR10	CIFAR100	Flowers	Cars
ViT-S	-	-	98.41 ± 0.12	87.27 ± 0.19	66.66 ± 2.52	46.04 ± 3.72
	✔	-	98.48 ± 0.06	87.54 ± 0.33	72.61 ± 1.20	45.46 ± 1.80
	✔	✔	**98.96** ± 0.06	**90.81** ± 0.09	**96.64** ± 0.23	**87.43** ± 0.43
ViT-B	-	-	98.97 ± 0.10	90.33 ± 0.14	90.92 ± 1.60	78.52 ± 0.59
	✔	-	99.06 ± 0.02	90.82 ± 0.21	92.45 ± 0.90	80.34 ± 0.56
	✔	✔	**99.15** ± 0.04	**91.52** ± 0.16	**98.44** ± 0.13	**92.22** ± 0.03

Table A.9. **Mean and std for analysis on drop regularization with/without MaskSub.** The table shows 'mean ± std' values for experiments in Table A.5 of the paper. Note that training loss scale 10^{-3} is omitted for simplicity.

		Single model			Sub-model training (MaskSub)		
	Drop ratio	Accuracy	Train loss (original)	Train loss (drop)	Accuracy	Train loss (original)	Train loss (drop)
Original	-	77.40 ± 0.20	6.42 ± 0.03	-	-	-	-
Masking [9]	25%	76.33 ± 0.28	6.60 ± 0.05	6.96 ± 0.05	79.02 ± 0.12	5.89 ± 0.03	6.38 ± 0.04
	50%	73.78 ± 0.08	7.02 ± 0.04	7.77 ± 0.03	79.36 ± 0.01	5.81 ± 0.01	6.89 ± 0.01
	75%	67.27 ± 0.25	8.08 ± 0.05	9.27 ± 0.04	79.16 ± 0.05	5.84 ± 0.01	8.15 ± 0.02
Drop-out [19]	0.1	76.09 ± 0.25	6.60 ± 0.07	6.87 ± 0.06	79.14 ± 0.15	5.88 ± 0.02	6.32 ± 0.02
	0.2	74.10 ± 0.22	6.95 ± 0.06	7.34 ± 0.06	79.10 ± 0.11	5.82 ± 0.04	6.57 ± 0.04
	0.3	71.62 ± 0.29	8.34 ± 0.03	7.79 ± 0.03	79.09 ± 0.15	5.84 ± 0.03	6.90 ± 0.03
Drop-path [13]	0.1	77.40 ± 0.20	6.42 ± 0.03	6.42 ± 0.03	78.36 ± 0.03	6.11 ± 0.01	6.11 ± 0.01
	0.2	74.92 ± 0.12	6.74 ± 0.04	7.19 ± 0.03	78.72 ± 0.12	5.91 ± 0.01	6.48 ± 0.01
	0.3	71.57 ± 0.10	7.31 ± 0.02	8.04 ± 0.02	78.80 ± 0.15	5.87 ± 0.02	7.02 ± 0.01

Table A.10. **Mean and std for three variants of MaskSub.** We report 'mean ± std' values for 400 epochs training with DeiT-III [21]. Note that we use the performance of the original paper [21] for baseline training.

Architecture	Baseline	DropSub	PathSub	MaskSub
ViT-S/16	80.40 ± 0.33	80.57 ± 0.12	80.78 ± 0.05	81.08 ± 0.12
ViT-B/16	83.46 ± 0.04	83.83 ± 0.11	83.80 ± 0.12	84.08 ± 0.02

B. Implementation details

Most experiments in the paper were performed on a machine with NVIDIA V100 8 GPUs. The exceptions were DeiT-III [21] experiments for ViT-L and ViT-H in Table 1 and object detection in Table A.3, conducted with NVIDIA A100 64 GPUs. Also, we use a single NVIDIA V100 for transfer learning, as shown in Table 7. We strictly follow original training recipes for experiments. We denote details of the training recipes to clarify our implementation details and assist in reproducing our results.

Image Classification. Table B.1 shows training recipes used for Table 1, 3, 5, 9, 13, and A.2. It demonstrates that our MaskSub is validated on various training recipes that cover diverse regularization and optimizer settings and achieves consistent improvement on all settings, which exhibits the general applicability of MaskSub. Note that MaskSub is applied to all recipes with the same masking ratio of 50%. Thus, MaskSub does not require hyper-parameter tuning specialized for each recipe. Model-specific training recipes of DeiT-III [21] are reported in Table B.2. DeiT-III achieves strong performance with sophistically tuned training parameters mainly focused on input size and drop-path rate. It makes improving DeiT-III more challenging than other recipes, yet our MaskSub accomplishes this with a reasonable performance gap.

Semantic segmentation. We use the BEiT v2's segmentation recipe [16] that utilizes MMCV [4] and MMSeg [5] library. Following the default setting, we replace the ViT backbone with the DeiT-III backbone, which includes layer-scale [20]. Then, we train the segmentation task for 160k iteration using DeiT-III and DeiT-III + MaskSub pretrained backbone.

Object detection and instance segmentation We use Detectron2 [25] on the COCO [15] dataset. Among various recipes in the Detectron2 library, we use ViTDet [14], which is a recent and strong recipe for a ViT-based detector. We train ViTDet Cascaded Mask-RCNN with DeiT-III and DeiT-III + MaskSub pretrained backbone and report performance after MSCOCO 100 epochs training.

Transfer learning. We use the AdamW training recipe on DeiT-III [21] transfer learning. We use lr 10^{-5}, weight-decay 0.05, batch size 768. Drop-path [13] and random erasing [27] are not used. Data augmentation is set to be the same as DeiT-III, and we train ViT for 1000 epochs with a cosine learning rate decay. For CIFAR datasets, we resize 32×32 image to 224×224 to use the ImageNet pretrained backbones. In the case of the iNaturalist datasets [23], we use AdamW with lr 7.5×10^{-5}, weight-decay 0.05, batch size 768. Drop-path and random erasing ratios are set to 0.1, and ViT is trained for 360 epochs with cosine learning rate decay.

Text classificaiton. We use the HuggingFace open-source code for the GLUE benchmark [24]. Random masking is implemented using the masking part of the HuggingFace BERT [7], which replaces random tokens with a mask token. Following the BERT pretraining, we set a random masking ratio to 15%, and only replacing tokens to the mask token is applied to randomly selected tokens.

CLIP pretraining. In pretraining, CLIP [17] encodes N image-text pairs of a batch and builds an $N \times N$ similarity matrix. Using contrastive learning loss, CLIP is trained to have high similarity for matched image-text pairs and low similarity for unmatched pairs. Implementing MaskSub upon CLIP is straightforward since the contrastive learning loss also utilizes *cross-entropy loss* over the similarity matrix. In the experiment, we only mask images, not texts. The difference between the "main-model" and "sub-model" is that the "sub-model" receives 50% masked images, and the "sub-model" is trained to mimic the similarity matrix of the "main-model" for the same batch. We use OPEN-CLIP[1] codebase for the CLIP pretraining. We follow the codebase's default training recipes. We train CLIP ViT-B/32 with a batch size of 1,024 using 8 NVIDIA V100 GPUs. Training datasets are CC3M [18], CC12M [3], and RedCaps [6]. The number of seen samples during training is 128M.

[1] https://github.com/mlfoundations/open_clip

Table B.1. **Details of various training recipes used for experiments.** Our MaskSub achieves consistent improvement in all training recipes, which covers diverse regularization and optimizer settings.

Training recipe	DeiT-III	RSB A2	Swin	MAE	BEiT v2	FT-CLIP
Fine-tuning	✗	✗	✗	✔	✔	✔
Epoch	400 / 800	300	300	100, 50	100, 50	50, 30
Batch size	2048	2048	1024	1024	1024	2048
Optimizer	LAMB	LAMB	AdamW	AdamW	AdamW	AdamW
LR	3×10^{-3}	5×10^{-3}	1×10^{-3}	$(2, 4) \times 10^{-3}$	$(5, 2) \times 10^{-4}$	$(6, 4) \times 10^{-4}$
LR decay	cosine	cosine	cosine	cosine	cosine	cosine
Layer LR decay	-	-	-	0.65, 0.75	0.6, 0.8	0.6, 0.65
Weight decay	0.03 / 0.05	0.01	0.05	0.05	0.05	0.05
Warmup epochs	5	5	20	5	20, 5	10, 5
Loss	BCE	BCE	CE	CE	CE	CE
Label smoothing	-	-	0.1	0.1	0.1	0.1
Dropout	-	-	-	-	-	-
Drop-path	Table B.2	0.05	0.1	0.1, 0.2, 0.3	0.2	-
Repeated aug	✔	✔	-	-	-	-
Gradient clip	1.0	-	5.0	-	-	-
RandAugment	Three Aug.	7 / 0.5	9 / 0.5	9 / 0.5	9 / 0.5	9 / 0.5
Mixup alpha	0.8	0.1	0.8	0.8	0.8	-
CutMix alpha	1.0	1.0	1.0	1.0	1.0	-
Random erasing	-	-	0.25	0.25	0.25	0.25
Color jitter	0.3	-	0.4	-	0.4	0.4
EMA	-	-	-	-	-	0.9998
Train image size	Table B.2	224×224	224×224	224×224	224×224	224×224
Test image size	224×224	224×224	224×224	224×224	224×224	224×224
Test crop ratio	1.0	0.95	0.875	0.875	0.875	1.0

Table B.2. **Model specific recipes of DeiT-III [21].** The table shows the model size and training length-specific training arguments used for the DeiT-III recipe. In addition to Table B.1, DeiT-III utilizes drop-path and image size to adjust the recipe.

		400 epochs				800 epochs			
		ViT-S	ViT-B	ViT-L	ViT-H	ViT-S	ViT-B	ViT-L	ViT-H
Pretraining	Image size	224	192	192	160	224	192	192	160
	Drop-path	0.0	0.1	0.4	0.5	0.05	0.2	0.45	0.6
	LR	0.004	0.003			0.004	0.003		
	Weight decay	0.03				0.05			
Resolution Finetuning	Drop-path	-	0.2	0.45	0.55	-	0.2	0.45	0.55
	Epochs	-	20			-	20		
	Image size	-	224 x 224			-	224 x 224		
	Optimizer	-	AdamW			-	AdamW		
	LR	-	1e-5			-	1e-5		

References

[1] Andrei Barbu, David Mayo, Julian Alverio, William Luo, Christopher Wang, Dan Gutfreund, Josh Tenenbaum, and Boris Katz. Objectnet: A large-scale bias-controlled dataset for pushing the limits of object recognition models. *Advances in neural information processing systems*, 32, 2019. 3

[2] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: high quality object detection and instance segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 43(5):1483–1498, 2019. 2

[3] Soravit Changpinyo, Piyush Sharma, Nan Ding, and Radu Soricut. Conceptual 12M: Pushing web-scale image-text pre-training to recognize long-tail visual concepts. In *CVPR*, 2021. 7

[4] MMCV Contributors. MMCV: OpenMMLab computer vision foundation. https://github.com/open-mmlab/mmcv, 2018. 7

[5] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 7

[6] Karan Desai, Gaurav Kaul, Zubin Aysola, and Justin Johnson. RedCaps: Web-curated image-text data created by the people, for the people. In *NeurIPS Datasets and Benchmarks*, 2021. 7

[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. 7

[8] Josip Djolonga, Jessica Yung, Michael Tschannen, Rob Romijnders, Lucas Beyer, Alexander Kolesnikov, Joan Puigcerver, Matthias Minderer, Alexander D'Amour, Dan Moldovan, et al. On robustness and transferability of convolutional neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16458–16468, 2021. 3

[9] 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, 4, 6

[10] Dan Hendrycks, Steven Basart, Norman Mu, Saurav Kadavath, Frank Wang, Evan Dorundo, Rahul Desai, Tyler Zhu, Samyak Parajuli, Mike Guo, et al. The many faces of robustness: A critical analysis of out-of-distribution generalization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 8340–8349, 2021. 3

[11] Dan Hendrycks, Kevin Zhao, Steven Basart, Jacob Steinhardt, and Dawn Song. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15262–15271, 2021. 3

[12] 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. 2

[13] 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. 4, 6, 7

[14] Yanghao Li, Hanzi Mao, Ross Girshick, and Kaiming He. Exploring plain vision transformer backbones for object detection. In *Proceedings of the European Conference on Computer Vision*, pages 280–296. Springer, 2022. 2, 7

[15] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Proceedings of the European Conference on Computer Vision*, pages 740–755. Springer, 2014. 2, 7

[16] 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. 7

[17] 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. 7

[18] Piyush Sharma, Nan Ding, Sebastian Goodman, and Radu Soricut. Conceptual captions: A cleaned, hypernymed, image alt-text dataset for automatic image captioning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2556–2565, 2018. 7

[19] 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. 4, 6

[20] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021. 7

[21] 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. 2, 5, 6, 7, 8

[22] 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

[23] 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

[24] 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. 7

[25] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 7

[26] 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. 2

[27] 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. 7