# *ForAug*: Mitigating Biases and Improving Vision Transformer Training by Recombining Foregrounds and Backgrounds – Supplementary Material –

Anonymous CVPR submission

Paper ID Supplementary

## Abstract

*This is the supplementary material for the paper: ForAug: Mitigating Biases and Improving Vision Transformer Training by Recombining Foregrounds and Backgrounds*

## A. Extended Bates Distribution

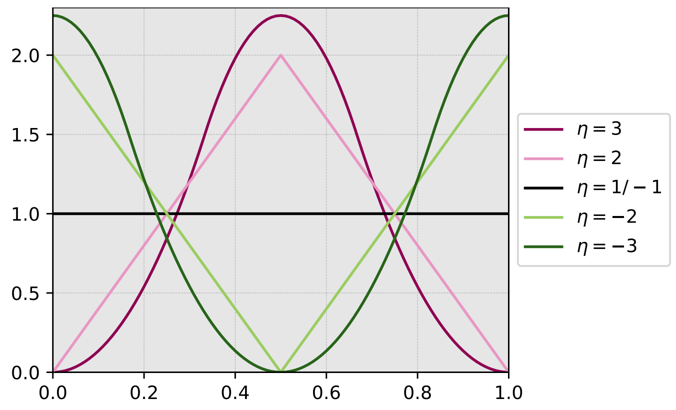


Figure 1. Plot of the probability distribution function (PDF) of the extended Bates distribution for different parameters $\eta$. Higher values of $\eta$ concentrate the distribution around the center.

We introduce an extension of the Bates distribution [1] to include negative parameters, enabling sampling of foreground object positions away from the image center. The standard Bates distribution, for $\eta \in \mathbb{N}$, is defined as the mean of $\eta$ independent random variables drawn from a uniform distribution [4]. A larger $\eta$ value increases the concentration of samples around the distribution's mean, which in this case is the image center.

To achieve an opposite effect–concentrating samples at the image borders–we extend the distribution to $\eta \leq 1$.

$$X \sim \text{Bates}(\eta) :\Leftrightarrow s(X) \sim \text{Bates}(-\eta)$$

This is accomplished by sampling from a standard Bates distribution with parameter $-\eta \geq 1$ and then applying a sawtooth function. The sawtooth function on the interval $[0, 1]$ is defined as

$$s(x) = \begin{cases} x + 0.5 & \text{if } 0 < x < 0.5 \\ x - 0.5 & \text{if } 0.5 \leq x \leq 1 \end{cases} \tag{1}$$

This function effectively maps the central portion of the interval to the edges and the edge portions to the center. For example, a value of 0.3 (central-left) is mapped to 0.8 (edge-right), while 0.8 (edge-right) is mapped to 0.3 (central-left). This transformation inverts the distribution's concentration, shifting the probability mass from the center to the borders. We visualize the distribution function of the extended Bates distribution in Figure 1. Both $\eta = 1$ and $\eta = -1$ result in a uniform distribution across the image.

## B. Resource Usage of *ForAug*

To utilize the proposed *ForAug*, specific computational resources are necessary, particularly for computing and storing for the output of the segmentation stage and for on-the-fly processing of the recombination stage.

**Segmentation.** *ForAug* involves a computationally expensive segmentation and infill stage, which is a one-time calculation per dataset. Once computed, the segmentation and infill results can be perpetually reused, amortizing the initial cost over all subsequent experiments and applications. On NVIDIA H100 GPUs, the segmentation stage will compute at a rate of $374.3\frac{\text{img}}{\text{GPU}\times\text{h}}$ when using Attentive Eraser or $5338.6\frac{\text{img}}{\text{GPU}\times\text{h}}$ for LaMa. For ImageNet this comes down to just under 9 days (Attentive Eraser) or 16 hours (LaMa) on two 8 GPU nodes. To facilitate immediate use and reproduction of results, we publicly provide the precalculated segmentation stage output for the ImageNet dataset for download[1]. The output of *ForAug*'s segmentation step on ImageNet dataset requires 73 GB of additional disk space for the segmentation output, which is separate from the base 147 GB ImageNet size.

**Recombination.** The recombination step of *ForAug* is implemented as a based data loader operation. It's thus offloaded to the CPU, where it can be heavily parallelized and thus only results in a very minor increase in the training step-time. For example, using a ViT-B model on an NVIDIA A100 GPU, the average update step-time increased by $1\%$, from $528 \pm 2$ ms to $534 \pm 1$ ms.

## C. Training Setup

Table 1. Training setup and hyperparameters for our ImageNet training.

| Parameter | ViT, Swin, ResNet | DeiT |
|---|---|---|
| Image Resolution | $224 \times 224$ | $224 \times 224$ |
| Epochs | 300 | 300 |
| Learning Rate | 3e-3 | S/B: 1e-3, L: 5e-4 |
| Learning Rate Schedule | cosine decay | cosine decay |
| Batch Size | 2048 | 1024 |
| GPUs | $4\times$ NVIDIA A100/H100/H200 | $4\times$ NVIDIA A100/H100/H200 |
| Warmup Schedule | linear | linear |
| Warmup Epochs | 3 | 3 |
| Weight Decay | 0.02 | 0.05 |
| Label Smoothing | 0.1 | 0.1 |
| Optimizer | Lamb [13] | AdamW |
| Data Augmentation Policy | **3-Augment [11]** | **DeiT [10]** |
| Augmentations | Resize<br>RandomCrop<br>HorizontalFlip<br>Grayscale<br>Solarize<br>GaussianBlur<br>ColorJitter<br>CutMix [14] | RandomResizedCrop<br>HorizontalFlip<br>RandomErase [16]<br>RandAugment [2]<br>ColorJitter<br>Mixup [15]<br>CutMix [14] |

[1]Link will go here.

Table 2. Training setup for finetuning on different downstream datasets. Other settings are the same as in Table 1. For finetuning, we always utilize 3-Augment and the related parameters from the *ViT, Swin, ResNet* column of Table 1

| Dataset | Batch Size | Epochs | Learning Rate | Num. GPUs |
|---|---|---|---|---|
| Aircraft | 512 | 500 | 3e-4 | 2 |
| Cars | 1024 | 500 | 3e-4 | 4 |
| Flowers | 256 | 500 | 3e-4 | 1 |
| Food | 2048 | 100 | 3e-4 | 4 |
| Pets | 512 | 500 | 3e-4 | 2 |

On ImageNet we use the same training setup as [5] and [11] without pretraining for ViT, Swin, and ResNet. For DeiT, we train the same ViT architecture but using the data augmentation scheme and hyperparameters from [10]. As our focus is on evaluating the changes in accuracy due to *ForAug*, like [5], we stick to one set of hyperparameters for all models. We list the settings used for training on ImageNet in Table 1 and the ones used for finetuning those weights on the downstream datasets in Table 2. Out implementation is using PyTorch [6] and the *timm* library [12] for model architectures and basic functions.

Table 3. Hardware and Software specifics used for both training and evaluation.

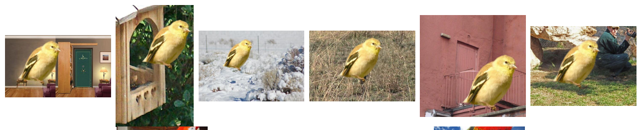
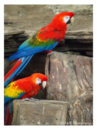
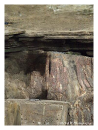
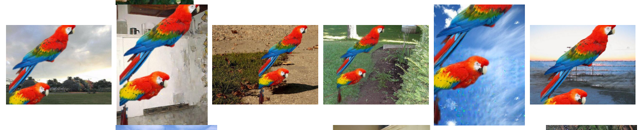
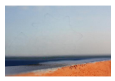
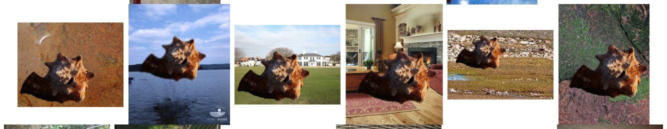
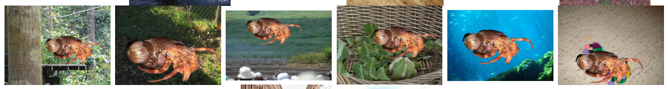
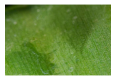
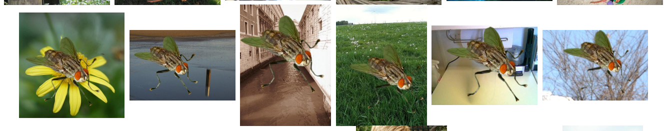
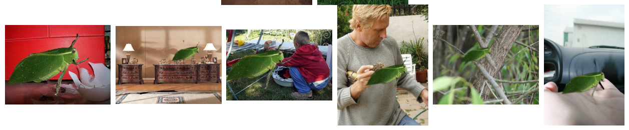
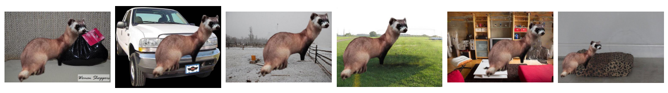
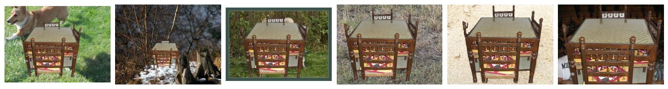
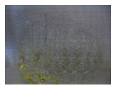
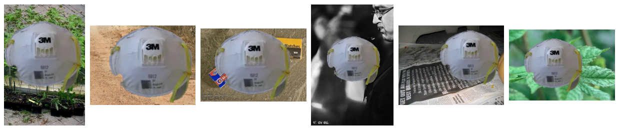
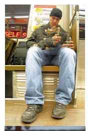
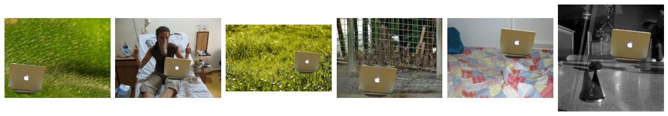
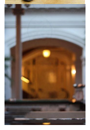
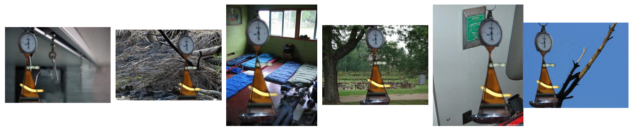
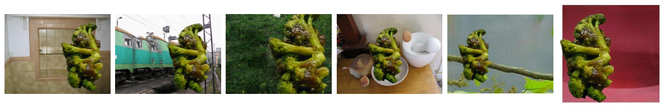
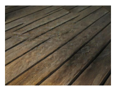
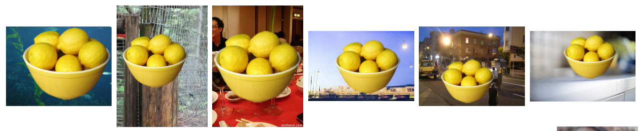
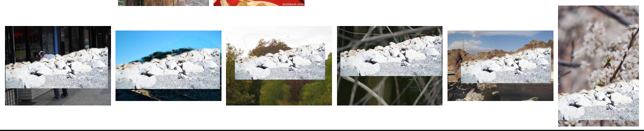
| Parameter | Value |
|---|---|
| GPU | NVIDIA A100/H100/H200 |
| CPU | 24 CPU cores (Intex Xenon) per GPU |
| Memory | up to 120GB per GPU |
| Operating System | Enroot container for SLURM based on Ubuntu 24.04 LTS |
| Python | 3.12.3 |
| PyTorch | 2.7.0 |
| TorchVision | 0.22.0 |
| Timm | 1.0.15 |

Table 3 lists the specific hardware we use, as well as versions of the relevant software packages.

## D. *ForAug* Sample Images

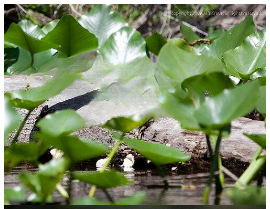
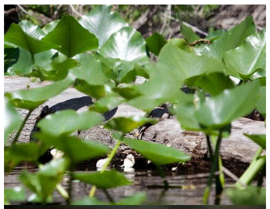
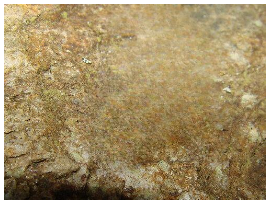
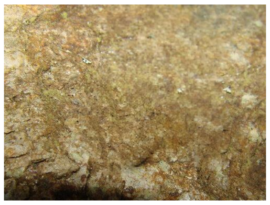
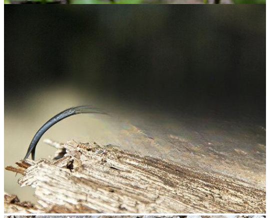
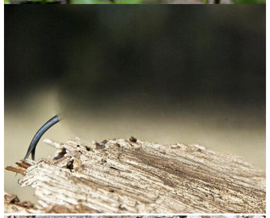
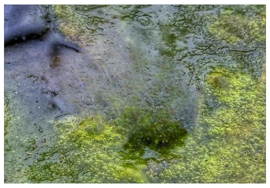
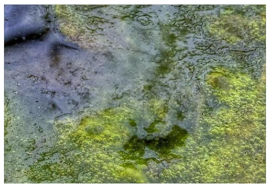
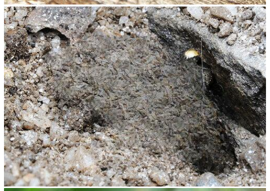
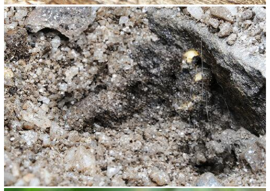
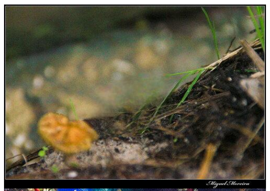
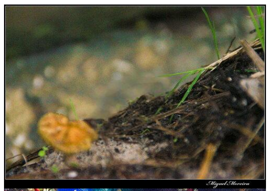
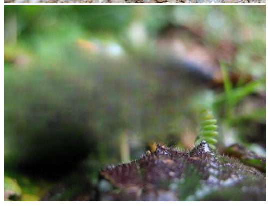
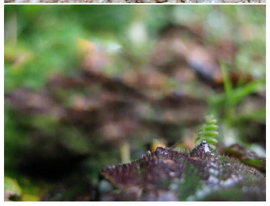
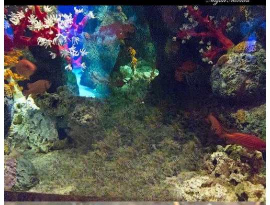
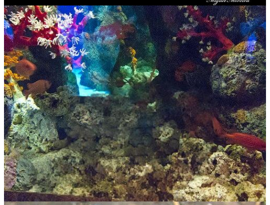
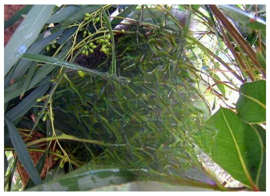
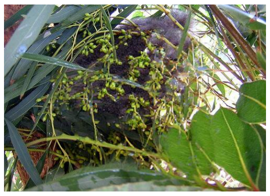
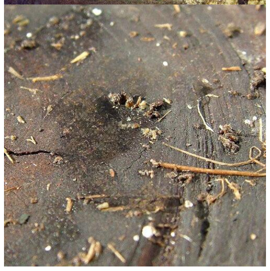
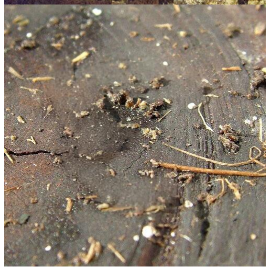
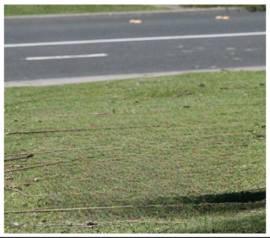
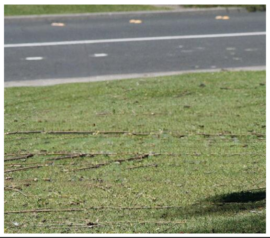
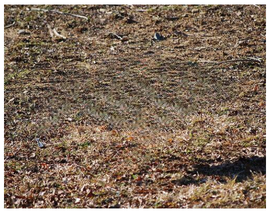
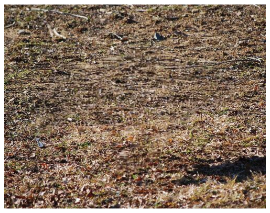
We show some example images of *ForAug*'s recombinations for 14 random classes of ImageNet [3] in Table 4. The recombined samples display substantial visual diversity, with each extracted foreground appearing in multiple, clearly different background contexts. Foreground objects remain sharp and well-preserved across recombinations, while backgrounds vary in texture, color, and scene type Images show a broad range of spatial placements and scales for the same object, resulting in noticeably different overall layouts.

Table 4. Sample Images from using *ForAug* on ImageNet.

| Class | Original Image | Extracted Foreground | Infilled Background | *ForAug*'s Recombinations |
|---|---|---|---|---|
| n01531178 Goldfinch | | | |  |
| n01818515 Macaw |  | |  |  |
| n01943899 Conch | | |  |  |
| n01986214 Hermit Crab | | | |  |
| n02190166 Fly | | |  |  |
| n02229544 Cricket | | | |  |
| n02443484 Black-Footed Ferret | | | |  |
| n03201208 Dining Table | | | |  |
| n03424325 Gasmask | | |  |  |
| n03642806 Laptop | | |  |  |
| n04141975 Scale | | |  |  |
| n07714990 Broccoli | | | |  |
| n07749582 Lemon | | |  |  |
| n09332890 Lakeside | | | |  |

## E. Infill Model Comparison

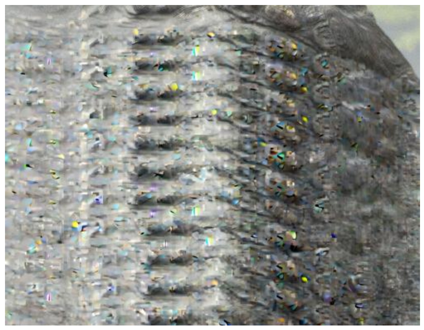
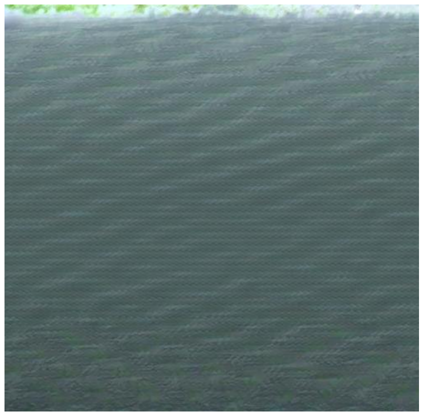
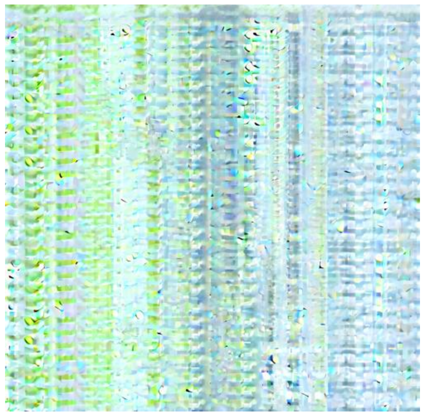
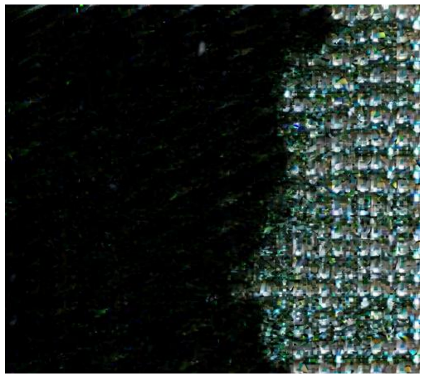
Table 5. Example infills of LaMa and Attentive Eraser.

| LaMa | Att. Eraser | LaMa | Att. Eraser |
|---|---|---|---|
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

We visualize example infilled images for both LaMa [9] and Attentive Eraser [8] in Table 5. The side-by-side examples show that both methods generally produce visually consistent infills, with many pairs appearing extremely similar at a glance. We qualitatively find that Attentive Eraser yields slightly sharper textures or more coherent local structure, while LaMa sometimes produces smoother or more homogenized regions. Across the table, fine-detail areas such as foliage, bark, and ground textures reveal the most noticeable differences between the two methods.

## F. Image Infill Ratio

Table 6. Example infills with a large relative foreground area size that is infilled (infill ratio).

| Infill Ratio | LaMa | Att. Eraser |
|---|---|---|
| 83.7 | |  |
| 88.2 | | |
| 93.7 |  |  |
| 95.7 | |  |

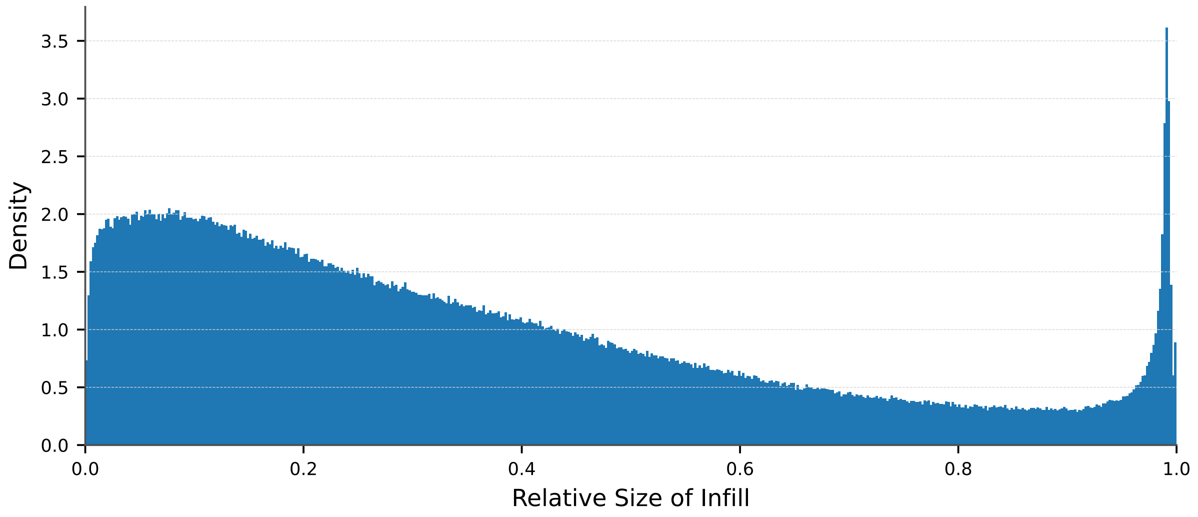


Figure 2. We plot the distribution of the relative size of the detected foreground object that is infilled in our Segmentation step of ImageNet. While most images contain objects of smaller size, there is a peak where Grounded SAM [7] detects almost the whole image as the foreground object. For examples of such large infills, see Table 6.

Table 6 shows infills for images where Grounded SAM [7] marks a high percentile of the image as the foreground object (Infill Ratio), that has to be erased by the infill models. The examples show that when the infilled region becomes large, both methods begin to lose coherent global structure, with outputs dominated by repetitive or texture-like patterns. LaMa tends to produce smoother, more uniform surfaces, like we saw in Table 5, while Attentive Eraser often generates denser, more regular texture patterns. Across the rows, increasing infill ratio corresponds to increasingly homogeneous results, with only faint hints of original scene cues remaining. Figure 2 plots the distribution of infill ratios in *ForAug*. While there is a smooth curve of the number of detections decreasing with the infill ratio until $\approx 90\%$, there is an additional peak at $\approx 100\%$ infill ratio. We hypothesize that this peak is made up of failure cases of Grounded SAM.

We filter out all backgrounds that have an infill ratio larger than our pruning threshold $t_{\text{prune}} = 0.8$, which translates to $10\%$ of backgrounds.

## References

[1] G.E. Bates. Joint distributions of time intervals for the occurrence of successive accidents in a generalized polya urn scheme. *Annals of Mathematical Statistics*, 26:705–720, 1955. 1

[2] Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space. 2019. 2

[3] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 2009. 3

[4] Norman L. Jonhson, Samuel Kotz, and N. Balakrishnan. *Continuous Univariate Distributions*. Wiley, 2 edition, 1995. Wiley series in probability and mathematical statistics. 1

[5] Tobias Christian Nauen, Sebastian Palacio, and Andreas Dengel. Which transformer to favor: A comparative analysis of efficiency in vision transformers. In *Proceedings of the Winter Conference on Applications of Computer Vision (WACV)*, pages 6955–6966, 2025. 3

[6] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. 3

[7] Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, Zhaoyang Zeng, Hao Zhang, Feng Li, Jie Yang, Hongyang Li, Qing Jiang, and Lei Zhang. Grounded sam: Assembling open-world models for diverse visual tasks. 2024. 7

[8] Wenhao Sun, Benlei Cui, Xue-Mei Dong, and Jingqun Tang. Attentive eraser: Unleashing diffusion model's object removal potential via self-attention redirection guidance. 2024. 5

[9] Roman Suvorov, Elizaveta Logacheva, Anton Mashikhin, Anastasia Remizova, Arsenii Ashukha, Aleksei Silvestrov, Naejin Kong, Harshith Goka, Kiwoong Park, and Victor Lempitsky. Resolution-robust large mask inpainting with fourier convolutions. 2021. 5

[10] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Herve Jegou. Training data-efficient image transformers & distillation through attention. In *Proceedings of the 38th International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021. 2, 3

[11] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *Computer Vision – ECCV 2022*, pages 516–533, Cham, 2022. Springer Nature Switzerland. 2, 3

[12] Ross Wightman. Pytorch image models. https://github.com/rwightman/pytorch-image-models, 2019. 3

[13] Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. Large batch optimization for deep learning: Training bert in 76 minutes. In *International Conference on Learning Representations*, 2020. 2

[14] Sangdoo Yun, Dongyoon Han, Sanghyuk Chun, Seong Joon Oh, Youngjoon Yoo, and Junsuk Choe. CutMix: Regularization strategy to train strong classifiers with localizable features. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*. IEEE, 2019. 2

[15] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 2

[16] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. 2017. 2