ForAug: Mitigating Biases in Image Classification via Controlled Image Compositions – Supplementary Material –

Anonymous ECCV 2026 Submission

Paper ID #1741–Supplementary

Abstract. This is the supplementary material for the paper: *ForAug*: Mitigating Biases in Image Classification via Controlled Image Compositions.

Keywords: Data Augmentation · Vision Transformer · Robustness

A Training Setup

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

Augmentation Pipeline:	Basic	3-Augment [13]	RandAugment [12]
Image Resolution		224×224	
Epochs		300	
Learning Rate	S/B: 1e-3, L: 5e-4	3e-3	S/B: 1e-3, L: 5e-4
Learning Rate Schedule		cosine decay	
Batch Size	1024	2048	1024
GPUs		4× NVIDIA A100/H100/H200	
Warmup Schedule		linear	
Warmup Epochs		3	
Weight Decay	0.05	0.02	0.05
Label Smoothing		0.1	
Optimizer	AdamW	Lamb [15]	AdamW
Augmentations	RandomResizedCrop Horizontal Flip ColorJitter	Resize RandomCrop Horizontal Flip Grayscale Solarize Gaussian-Blur Color Jitter	RandomResizedCrop Horizontal Flip RandomErase [18] RandAugment [2] Color Jitter

On ImageNet, we test three different data augmentation pipelines and hyperparameter settings as shown in Table 1: A basic pipeline, a pipeline using

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

RandAugment based on the DeiT [12] setup and 3-Augment, as used in [7, 13]. When comparing different architectures, ViT, Swin, and ResNet are trained with the 3-Augment pipeline and DeiT is trained with the RandAugment pipeline. As our focus is on evaluating the changes in accuracy due to *ForAug*, like [7], 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. Our implementation is using PyTorch [8] and the *timm* library [14] for model architectures and basic functions.

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

Parameter	Value
GPU	4× NVIDIA A100/H100/H200
CPU	24 CPU cores (Intel Xenon) per GPU
Memory	up to 120 GB 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.

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 ± 2 ms to 534 ± 1 ms.

C Extended Bates Distribution

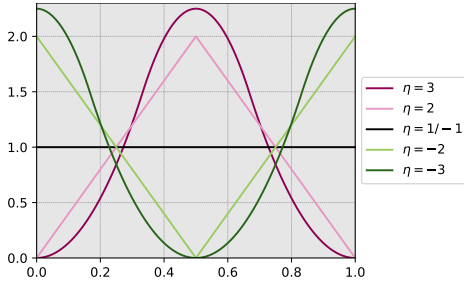


Fig. 1: Plot of the probability distribution function (PDF) of the extended Bates distribution for different parameters η. Higher values of η concentrate the distribution around the center.

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 η independent random variables drawn from a uniform distribution [5]. A larger η 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 $\eta \leq 1$.

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

[1] Link will go here.

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.

D Design Choices of *ForAug*

We start by ablating the design choices of *ForAug* on TinyImageNet [6], a subset of ImageNet containing 200 categories with 500 images each. Table 4 presents ablations for segmentation and Table 5 for recombination.

Table 4: Ablation of the design decisions in the segmentation phase of *ForAug* on TinyImageNet. The first line is our baseline, while the other lines are using *ForAug*. We use basic settings with the *same* background strategy during recombination for this experiment.

Detect. Prompt	Infill Model	TinyImageNet Accuracy [%]	
		ViT-Ti	ViT-S
TinyImageNet		66.1 ± 0.5	68.3 ± 0.7
specific	LaMa [11]	65.5 ± 0.4	71.2 ± 0.5
general	LaMa [11]	66.4 ± 0.6	72.9 ± 0.6
general	Att. Eraser [10]	67.5 ± 1.2	72.4 ± 0.5

Prompt. First, we evaluate the type of prompt used to detect the foreground object. Here, the *general* prompt, which contains the class and the more general object category, outperforms only having the class name (*specific*).

Inpainting. Among inpainting models, Attentive Eraser [10] produces slightly better results compared to LaMa [11] (+0.5 p.p. on average). For inpainting examples, see the supplementary material.

Foreground size significantly impacts performance. Employing a *range* of sizes during recombination, rather than a fixed *mean* size, boosts accuracy by approximately 1 p.p. This suggests that the added variability is beneficial.

Table 5: Ablation of the recombination phase of *ForAug* on TinyImageNet (top) and ImageNet (bottom). The first experiments use the initial segmentation settings with LaMa [11].

FG. size	Augment. Order	BG Strat.	BG. Prune	Original Mixing	Edge Smooth.	Accuracy [%] ViT-Ti	ViT-S
TinyImageNet						66.1 ± 0.5	68.3 ± 0.7
mean	crop→paste	same	-	-	-	64.6 ± 0.5	70.0 ± 0.6
range	crop→paste	same	-	-	-	65.5 ± 0.4	71.2 ± 0.5
range	crop→paste	same	-	-	-	67.5 ± 1.2	72.4 ± 0.5
range	paste→crop	same	-	-	-	67.1 ± 1.2	72.9 ± 0.5
range	paste→crop	same	1.0	-	-	67.0 ± 1.2	73.0 ± 0.3
range	paste→crop	same	0.8	-	-	67.2 ± 1.2	72.9 ± 0.8
range	paste→crop	same	0.6	-	-	67.5 ± 1.0	72.8 ± 0.7
range	paste→crop	same	0.8	$p = 0.2$	-	69.8 ± 0.5	75.0 ± 0.3
range	paste→crop	same	0.8	$p = 0.33$	-	69.5 ± 0.4	75.2 ± 1.0
range	paste→crop	same	0.8	$p = 0.5$	-	70.3 ± 1.0	74.2 ± 0.2
range	paste→crop	same	0.8	linear	-	70.1 ± 0.7	74.9 ± 0.8
range	paste→crop	same	0.8	reverse lin.	-	67.6 ± 0.2	73.2 ± 0.3
range	paste→crop	same	0.8	cos	-	71.3 ± 1.0	75.7 ± 0.8
range	paste→crop	same	0.8	cos	$\sigma_{\max} = 4.0$	70.0 ± 0.8	75.5 ± 0.7
range	paste→crop	orig.	0.8	cos	$\sigma_{\max} = 4.0$	67.2 ± 0.9	69.9 ± 1.0
range	paste→crop	all	0.8	cos	$\sigma_{\max} = 4.0$	70.1 ± 0.7	77.5 ± 0.6
ImageNet						-	79.1 ± 0.1
range	paste→crop	same	0.8	cos	-	-	80.5 ± 0.1
range	paste→crop	same	0.8	cos	$\sigma_{\max} = 4.0$	-	80.7 ± 0.1
range	paste→crop	all	0.8	cos	$\sigma_{\max} = 4.0$	-	81.4 ± 0.1

Order of data augmentation. Applying all augmentations after foreground-background recombination (*paste→crop→color*) improves ViT-S's performance compared to applying crop-related augmentations before pasting (*crop→paste→color*). ViT-Ti results are ambiguous.

Background pruning. When it comes to the backgrounds to use, we test different pruning thresholds (t_{prune}) to exclude backgrounds with large inpainting. A threshold of $t_{\text{prune}} = 1.0$ means that we use all backgrounds that are not fully infilled. Varying t_{prune} has minimal impact. We choose $t_{\text{prune}} = 0.8$ to exclude predominantly artificial backgrounds.

Mixing *ForAug*-augmented samples with the original ImageNet data proves crucial. While constant and linear mixing schedules improve performance over no mixing by $2-3$ p.p. compared to only augmented samples, the cosine annealing schedule proves optimal, boosting accuracy by $3-4$ p.p.

Edge smoothing. We evaluate the impact of using Gaussian blurring to smooth the edges of the foreground masks. For larger models, this gives us a slight performance boost on the full ImageNet (second to last line in Table 5).

Background strategy. Another point is the allowed choice of background image for each foreground object. We compare using the original background, a background from the same class, and any background. These strategies go from low diversity and high shared information content between the foreground and background to high diversity and low shared information content. For *ViT-Ti*, the latter two strategies perform comparably, while *ViT-S* benefits from the added diversity of using any background. The same is true when training on the full ImageNet.

Table 6: Accuracy of ViT-S on TinyImageNet (TIN) in percent using *ForAug* with different foreground position distributions by varying the Bates parameter η. The best performance is achieved when using the uniform distribution ($\eta = 1$) for training.

Bates Parameter during training	TIN w/o *ForAug*	TIN w/ *ForAug*				
		$\eta = -3$	-2	$1/-1$	2	3
Baseline	68.9	60.5	60.2	60.8	62.6	63.1
$\eta = -3$	71.3	79.3	79.5	79.1	79.3	79.1
$\eta = -2$	71.5	80.0	78.7	79.3	79.1	78.8
$\eta = 1/-1$	72.3	79.5	78.9	80.2	79.7	80.4
$\eta = 2$	71.3	78.2	77.8	79.1	79.6	79.9
$\eta = 3$	71.4	77.2	76.9	78.6	79.6	79.7

Foreground position. Finally, we analyze the foreground object's positioning in the image, using a generalization of the Bates distribution [1] with parameter $\eta \in \mathbb{Z}$ (see Appendix C). The Bates distribution presents an easy way to sample from a bounded domain with just one hyperparameter that controls its concentration. $\eta = 1/-1$ corresponds to the uniform distribution; $\eta > 1$ concentrates the distribution around the center; and for $\eta < -1$, the distribution is concentrated at the borders (see supplementary material for details). When sampling more towards the center of the image, the difficulty of the task is reduced, which reduces performance on TinyImageNet (Table 6). This is reflected in the performance when evaluating using *ForAug* with $\eta = 2$ and $\eta = 3$ compared to $\eta = -1/1$. We observe a similar reduction for $\eta < -1$.

After fixing the optimal design parameters in Tables 4 and 5 (last rows), we run *ForAug*'s segmentation step on the entire ImageNet dataset. Table 7 shows the resulting dataset statistics. The slightly reduced image count for *ForAug* is due to instances where Grounded SAM fails to produce valid segmentation masks.

E Robustness Evaluation on Corner-Cases

Table 8 reports accuracy on the corner-cases dataset [4] for models trained with and without *ForAug*. The dataset is constructed by pasting objects cropped by

Table 7: Dataset statistics for TinyImageNet and ImageNet with and without *ForAug*. For *ForAug* we report the number of foreground/background pairs.

Dataset	Classes	Training Images	Validation Images
TinyImageNet	200	100 000	10 000
TinyImageNet + *ForAug*	200	99 404	9915
ImageNet	1000	1 281 167	50 000
ImageNet + *ForAug*	1000	1 274 557	49 751

their full bounding boxes (which are available for the ImageNet validation set) onto 224×224 infilled backgrounds. The dataset has three factors: foreground size (56, 84, 112 pixels), spatial position (center, CeX, vs. corner, CoX), and background type (original image background, XxO, vs. a random background, XxR), yielding $3 \times 2 \times 2$ controlled configurations per model.

Across all architectures, training with *ForAug* consistently improves robustness to these composition shifts. For ViT-S/B/L, gains range from roughly +8 to over +27 percentage points, with the largest improvements occurring in the most challenging settings with foregrounds placed in corners on random backgrounds (e.g., CoR and CeR). Swin and ResNet models also benefit across all configurations, with increases typically between +3 and +10 points. DeiT-S shows small drops on some same-background center cases (CeO/CoO), but still improves notably on random-background conditions (XxR), while DeiT-B/L gain across nearly all settings.

Three trends are apparent. First, all baselines perform substantially worse when moving from original to random backgrounds and from centered to corner placements, indicating strong background and center biases. Second, *ForAug* reduces this sensitivity: the absolute gap between center and corner, and between original and random backgrounds, shrinks for almost all models and sizes. Third, the relative improvements are especially pronounced for smaller objects and off-center placements, suggesting that *ForAug* makes models more foreground-focused and less reliant on canonical object scale and position.

F *ForAug* Segmentation Samples

We show examples of the automatically generated segmentation masks for a diverse subset of object categories (“ant,” “busby,” “bell cote,” “pickelhaube,” “snorkel,” “stove,” “tennis “ball,” and “volleyball”). Note that “busby,” “bell cote,” “pickelhaube,” and “snorkel” are the four classes with the **worst** mean box precision and box-to-box IoU on the validation set. Figure 2 (right) illustrates masks from the evaluation split, while Figure 2 (left) shows examples from the training split. Across both sets, the masks accurately isolate foreground objects with clean boundaries, despite large variations in object scale, shape, and appearance, supporting their use for background removal and resampling in our training

Table 8: Evaluation on the Corner-Cases dataset. Objects cut from ImageNet evaluation bounding boxes are pasted onto infilled backgrounds. Objects have three sizes: 56px, 84px, and 112px. Objects are places in the center (CeX) or corner (CoX) of an image its original background (XxO) or a random background (XxR).

Model	w/ *ForAug*	Corner Cases Accuracy [%]											
		56				84				112			
		CeO	CoO	CeR	CoR	CeO	CoO	CeR	CoR	CeO	CoO	CeR	CoR
ViT-S	✗	40.5 ± 2.0	28.6 ± 0.8	10.3 ± 0.9	6.4 ± 0.2	56.8 ± 1.2	47.6 ± 1.0	31.3 ± 0.7	25.5 ± 0.5	70.9 ± 0.1	66.9 ± 1.6	55.2 ± 0.2	51.1 ± 0.8
ViT-S	✓	49.4 ± 0.6	39.9 ± 0.5	22.7 ± 0.4	17.6 ± 0.3	66.3 ± 0.3	60.0 ± 0.3	47.7 ± 0.7	43.2 ± 0.2	76.5 ± 0.2	74.9 ± 0.4	66.8 ± 0.6	64.9 ± 0.1
		+8.9	+11.3	+12.4	+11.2	+9.4	+12.4	+16.4	+17.7	+5.6	+8.0	+11.6	+13.7
ViT-B	✗	37.9 ± 1.4	29.3 ± 0.7	14.0 ± 1.7	11.9 ± 1.1	51.5 ± 0.7	45.0 ± 0.8	27.3 ± 0.8	26.3 ± 0.8	64.7 ± 0.3	61.8 ± 0.6	46.3 ± 0.3	45.5 ± 0.5
ViT-B	✓	50.4 ± 0.8	42.4 ± 0.6	26.5 ± 0.6	22.8 ± 0.8	65.3 ± 0.9	60.9 ± 0.6	47.6 ± 0.3	45.6 ± 0.1	75.7 ± 0.6	74.0 ± 0.6	65.7 ± 0.7	64.3 ± 0.5
		+12.5	+13.1	+12.4	+10.9	+13.8	+15.9	+20.2	+19.3	+11.0	+12.2	+19.3	+18.8
ViT-L	✗	32.8 ± 1.6	24.8 ± 1.1	14.8 ± 2.2	9.7 ± 1.2	42.7 ± 0.9	33.8 ± 0.7	21.3 ± 1.5	16.3 ± 1.0	55.7 ± 0.7	49.7 ± 0.7	36.0 ± 1.3	32.5 ± 0.9
ViT-L	✓	45.7 ± 0.6	39.0 ± 0.5	25.6 ± 0.6	24.1 ± 0.8	59.1 ± 0.3	55.2 ± 0.4	41.9 ± 1.0	42.7 ± 0.6	71.4 ± 0.3	69.0 ± 0.4	60.7 ± 1.0	60.3 ± 0.8
		+12.9	+14.2	+10.8	+14.4	+16.3	+21.5	+20.5	+26.4	+15.7	+19.3	+24.7	+27.8
DeiT-S	✗	46.3 ± 0.7	38.1 ± 0.3	13.1 ± 0.5	9.9 ± 0.1	62.8 ± 0.4	58.2 ± 0.2	37.1 ± 0.7	34.3 ± 0.5	73.3 ± 0.2	73.9 ± 0.4	58.8 ± 0.4	59.4 ± 0.6
DeiT-S	✓	44.7 ± 1.4	37.1 ± 1.4	15.6 ± 1.3	12.1 ± 0.9	62.1 ± 1.2	57.8 ± 1.1	41.6 ± 1.1	37.9 ± 1.2	73.2 ± 0.7	73.3 ± 0.4	62.3 ± 0.7	61.4 ± 0.9
		−1.6	−1.1	+2.4	+2.2	−0.7	−0.4	+4.4	+3.5	−0.1	−0.6	+3.5	+2.0
DeiT-B	✗	48.1 ± 0.9	40.4 ± 2.0	15.8 ± 0.2	12.9 ± 0.6	64.0 ± 0.9	59.5 ± 1.3	39.0 ± 0.9	37.2 ± 0.8	74.1 ± 0.7	74.8 ± 0.7	59.1 ± 0.8	60.0 ± 0.6
DeiT-B	✓	50.7 ± 0.1	44.0 ± 0.4	19.3 ± 0.2	16.3 ± 0.2	66.0 ± 0.2	62.0 ± 0.3	43.4 ± 0.3	40.9 ± 0.4	75.4 ± 0.1	76.4 ± 0.3	62.8 ± 0.2	63.9 ± 0.2
		+2.6	+3.6	+3.5	+3.5	+2.0	+2.5	+4.4	+3.8	+1.3	+1.6	+3.8	+3.9
DeiT-L	✗	39.2 ± 2.6	32.6 ± 1.5	10.5 ± 2.8	9.1 ± 2.3	55.7 ± 2.5	51.0 ± 2.7	30.3 ± 4.0	29.5 ± 3.9	68.5 ± 2.1	68.1 ± 1.7	51.7 ± 3.1	52.1 ± 2.7
DeiT-L	✓	51.9 ± 0.7	46.6 ± 0.5	21.5 ± 1.3	19.0 ± 1.2	66.6 ± 0.6	64.1 ± 0.7	45.3 ± 1.3	43.6 ± 1.1	75.6 ± 0.4	77.3 ± 0.4	63.8 ± 0.8	65.4 ± 0.6
		+12.8	+14.0	+11.0	+9.9	+11.0	+13.1	+15.0	+14.1	+7.1	+9.2	+12.1	+13.4
Swin-Ti	✗	41.2 ± 1.8	32.5 ± 0.3	17.4 ± 2.6	12.2 ± 0.2	60.0 ± 1.6	51.4 ± 0.2	39.6 ± 2.6	34.8 ± 0.9	71.7 ± 0.8	66.1 ± 0.7	58.2 ± 1.1	53.6 ± 1.2
Swin-Ti	✓	49.8 ± 0.6	42.8 ± 0.7	24.2 ± 0.7	21.4 ± 0.9	66.4 ± 0.6	60.5 ± 0.2	47.8 ± 0.5	44.6 ± 0.5	76.0 ± 0.3	72.7 ± 0.2	65.7 ± 0.5	62.1 ± 0.3
		+8.5	+10.3	+6.8	+9.2	+6.4	+9.2	+8.2	+9.8	+4.3	+6.5	+7.5	+8.5
Swin-S	✗	41.3 ± 0.6	33.0 ± 0.1	18.4 ± 0.7	13.3 ± 0.5	59.2 ± 0.1	51.2 ± 0.5	39.1 ± 0.2	35.9 ± 0.3	71.5 ± 0.2	65.6 ± 0.1	56.8 ± 0.5	53.2 ± 0.2
Swin-S	✓	48.6 ± 0.7	39.9 ± 1.6	22.2 ± 0.9	16.8 ± 1.1	64.4 ± 0.9	57.9 ± 1.5	43.8 ± 1.1	42.3 ± 1.0	75.7 ± 0.2	71.8 ± 0.8	63.2 ± 0.4	60.6 ± 0.6
		+7.3	+7.0	+3.8	+3.6	+5.1	+6.7	+4.7	+6.4	+4.2	+6.2	+6.4	+7.4
ResNet50	✗	48.6 ± 0.6	35.1 ± 0.4	23.0 ± 0.7	13.0 ± 0.3	65.8 ± 0.4	58.2 ± 0.3	44.4 ± 0.6	38.1 ± 0.5	73.2 ± 0.2	69.9 ± 0.2	56.9 ± 0.1	56.9 ± 0.1
ResNet50	✓	52.3 ± 0.6	39.5 ± 0.1	27.4 ± 0.6	17.6 ± 0.1	68.5 ± 0.3	61.9 ± 0.1	48.5 ± 0.4	43.7 ± 0.3	75.2 ± 0.1	72.4 ± 0.1	61.7 ± 0.3	61.7 ± 0.3
		+3.7	+4.4	+4.4	+4.6	+2.8	+3.8	+4.2	+5.5	+2.0	+2.5	+4.8	+4.8
ResNet101	✗	47.8 ± 0.7	37.2 ± 0.5	20.4 ± 1.2	14.2 ± 0.3	64.9 ± 0.2	58.6 ± 0.5	41.1 ± 0.5	38.3 ± 0.7	73.6 ± 0.3	70.5 ± 0.3	56.2 ± 0.4	57.0 ± 0.5
ResNet101	✓	52.3 ± 0.1	42.2 ± 0.1	24.7 ± 0.1	19.2 ± 0.4	68.8 ± 0.6	62.9 ± 0.3	46.4 ± 1.5	44.3 ± 0.9	76.0 ± 0.4	73.7 ± 0.3	61.0 ± 1.2	62.6 ± 0.5
		+4.4	+5.0	+4.3	+5.0	+3.9	+4.3	+5.3	+6.0	+2.4	+3.2	+4.7	+5.7

Fig. 2: ImageNet validation samples (left) and training samples (right) of our segmentation masks with annotated bounding boxes.

pipeline. We find that the main failure cases are: (*i*) When the ground-truth annotation corresponds to only a part of an object, the predicted mask often expands to cover the entire object rather than the annotated region. See for example "busby" or "bell cote". (*ii*) In images containing multiple instances, some objects may be missed, resulting in incomplete foreground coverage. This is especially visible for "busby" and "pickelhaube". However, note that especially for "pickelhaube" the training distribution is noticeably different from the validation distribution, showing many images with just the head instead of groups of people wearing it. (*iii*) In rare cases, the predicted mask degenerates and covers nearly the entire image, effectively eliminating the background. This happens in $< 10\%$ of all training images, and we do not use the resulting backgrounds for recombination (see Appendix I).

G *ForAug* Sample Images

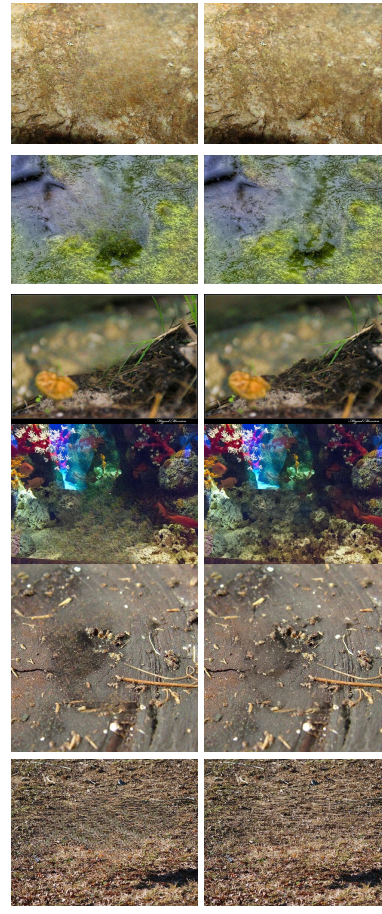
We show some example images of *ForAug*'s recombinations for 14 random classes of ImageNet [3] in Table 9. The recombined samples display substantial visual diversity, with each extracted foreground appearing in multiple, clearly different background contexts. Foreground objects remain sharp and well-preserved 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 9: 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									

H Infill Model Comparison

Table 10: Example infills of LaMa and Attentive Eraser.

LaMa	Att. Eraser	LaMa	Att. Eraser
			

We visualize example infilled images for both LaMa [11] and Attentive Eraser [10] in Table 10. 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.

I Image Infill Ratio

Table 11: Example infills with a large relative foreground area size that is infilled (infill ratio).

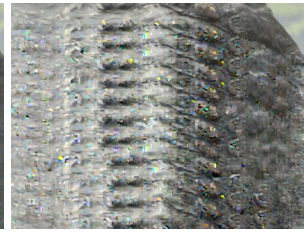
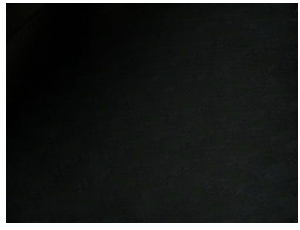
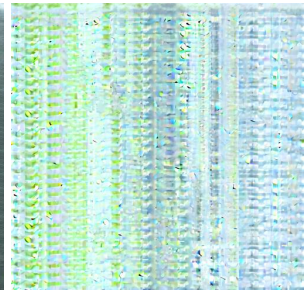
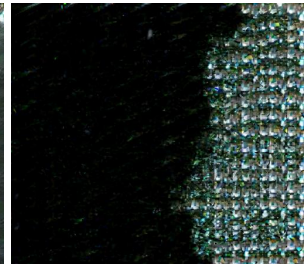
Infill Ratio	LaMa	Att. Eraser
83.7		
88.2		
93.7		
95.7		

Table 11 shows infills for images where Grounded SAM [9] marks a high percentile of the image as the foreground object (Infill Ratio), that has to be

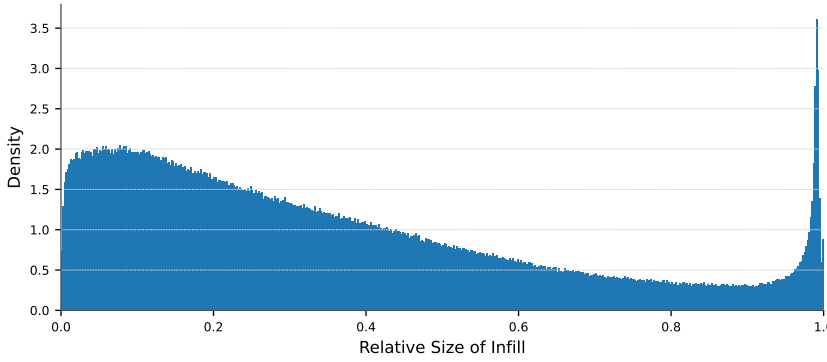


Fig. 3: We plot the distribution of the relative size of the detected foreground object that is infilled in our Segmentation step of ImageNet. While most images contain objects of smaller size, there is a peak where Grounded SAM [9] detects almost the whole image as the foreground object. For examples of such large infills, see Table 11.

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 10, 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 3 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. Bates, G.: 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)
2. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the 34th International Conference on Neural Information Processing Systems. NIPS '20, Curran Associates Inc., Red Hook, NY, USA (2020)
3. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition. IEEE (2009). `https://doi.org/10.1109/cvpr.2009.5206848`
4. Fatima, M., Jung, S., Keuper, M.: Corner cases: How size and position of objects challenge imagenet-trained models. Transactions on Machine Learning Research (2025), `https://openreview.net/forum?id=Yqf2BhqfyZ`

5. Jonhson, N.L., Kotz, S., Balakrishnan, N.: Continuous Univariate Distributions. Wiley series in probability and mathematical statistics., Wiley, 2 edn. (1995), wiley series in probability and mathematical statistics
6. Le, Y., Yang, X.: Tiny imagenet visual recognition challenge. CS 231N **7**(7), 3 (2015)
7. Nauen, T.C., Palacio, S., Raue, F., Dengel, A.: Which transformer to favor: A comparative analysis of efficiency in vision transformers. In: Proceedings of the Winter Conference on Applications of Computer Vision (WACV). pp. 6955–6966 (February 2025)
8. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: Advances in Neural Information Processing Systems 32, pp. 8024–8035. Curran Associates, Inc. (2019)
9. Ren, T., Liu, S., Zeng, A., Lin, J., Li, K., Cao, H., Chen, J., Huang, X., Chen, Y., Yan, F., Zeng, Z., Zhang, H., Li, F., Yang, J., Li, H., Jiang, Q., Zhang, L.: Grounded sam: Assembling open-world models for diverse visual tasks (2024). https://doi.org/10.48550/ARXIV.2401.14159
10. Sun, W., Cui, B., Dong, X.M., Tang, J.: Attentive eraser: Unleashing diffusion model's object removal potential via self-attention redirection guidance. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 39, pp. 20734–20742 (2025)
11. Suvorov, R., Logacheva, E., Mashikhin, A., Remizova, A., Ashukha, A., Silvestrov, A., Kong, N., Goka, H., Park, K., Lempitsky, V.: Resolution-robust large mask inpainting with fourier convolutions. In: 2022 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (2022). https://doi.org/10.1109/WACV51458.2022.00323
12. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jegou, H.: Training data-efficient image transformers & distillation through attention. In: Meila, M., Zhang, T. (eds.) Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 10347–10357. PMLR (7 2021), https://proceedings.mlr.press/v139/touvron21a.html
13. Touvron, H., Cord, M., Jégou, H.: Deit iii: Revenge of the vit. In: Avidan, S., Brostow, G., Cissé, M., Farinella, G.M., Hassner, T. (eds.) Computer Vision – ECCV 2022. pp. 516–533. Springer Nature Switzerland, Cham (2022)
14. Wightman, R.: Pytorch image models. https://github.com/rwightman/pytorch-image-models (2019). https://doi.org/10.5281/zenodo.4414861
15. You, Y., Li, J., Reddi, S., Hseu, J., Kumar, S., Bhojanapalli, S., Song, X., Demmel, J., Keutzer, K., Hsieh, C.J.: Large batch optimization for deep learning: Training bert in 76 minutes. In: International Conference on Learning Representations (2020). https://doi.org/10.48550/arxiv.1904.00962, https://openreview.net/forum?id=Syx4wnEtvH
16. Yun, S., Han, D., Chun, S., Oh, S.J., Yoo, Y., Choe, J.: CutMix: Regularization strategy to train strong classifiers with localizable features. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE (2019). https://doi.org/10.1109/iccv.2019.00612
17. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations (2018), https://openreview.net/forum?id=r1Ddp1-Rb

18. Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: Proceedings of the AAAI Conference on Artificial Intelligence (AAAI) (2020)