SAGE: Saliency-Guided Mixup with Optimal Rearrangements

Avery Ma[1,2*]
ama@cs.toronto.edu
Nikita Dvornik[3]
n.dvornik@samsung.com
Ran Zhang[3]
ran.zhang@samsung.com
Leila Pishdad[4†]
leila.pishdad@mail.mcgill.ca
Konstantinos G. Derpanis[2,3,5]
kosta@yorku.ca
Afsaneh Fazly[3]
a.fazly@samsung.com

[1] University of Toronto
[2] Vector Institute
[3] Samsung AI Centre Toronto
[4] Borealis AI
[5] York University

A Summary of the Supplementary Material

The supplementary material is organized as follows. In Sec. B, we describe the exact optimization schedule and the hyperparameters used to train with SAGE and other baseline DA frameworks. In Sec. C and Sec. D, we provide detailed results to bolster our claim on SAGE's improvement on OOD generalization (Sec. 4.2) and its low computation overhead (Sec. 4.3). Pseudocode to augment data with SAGE is included in Sec. E. In Sec. F, we show examples of augmentations using SAGE w/o SM and SAGE w/o OR (Sec. 4.4). Furthermore, we provide additional ablation studies to verify the design choices of SAGE in Sec. G.

B Optimization schedule and hyper-parameters

Optimization schedule: Following previous work [3, 4], all models are trained using stochastic gradient descent (SGD) for 300 epochs with an initial learning rate of 0.2. The learning rate decreases by a factor of 0.1 at epoch 100 and 200. We use a momentum of 0.9 and a weight decay of 0.0001. The above optimization schedule is used to train both CIFAR-10 and CIFAR-100 for all models, except for Co-Mixup [4] on CIFAR-10. We notice that training with Co-Mixup on CIFAR-10 with an initial learning rate of 0.2 results in divergence at the beginning of the training. We find training becomes stable with an initial learning rate of 0.12.

* Work done during an internship at Samsung AI Centre Toronto
† Work done while at Samsung AI Centre Toronto

Training with baseline DA: We follow the hyperparameter settings used in previous work [3, 4]. To train with Mixup [9], CutMix [8], Puzzle Mix [3] and Co-Mixup [4], we use $\lambda \sim \text{Beta}(\alpha, \alpha)$ with $\alpha = 1.0$, and use $\alpha = 2.0$ for Manifold Mixup [7]. For SaliencyMix [1], Puzzle Mix[2] and Co-Mixup[3], we use the parameter settings described in author's public repository: $(\beta, \mathbb{P}_{\text{mix}}) = (1.0, 0.5)$, $(\beta, \gamma, \eta, \varepsilon) = (1.2, 0.5, 0.2, 0.8)$ and $(\beta, \gamma, \eta, \tau, \omega) = (0.32, 1.0, 0.05, 0.83, 0.001)$.

Training with SAGE: For all models and datasets, we use 1% of all possible rearrangements (Sec. 3.3) and a smoothing parameter of $\sigma^2 = 1.0$ (Sec. 3.2). Here we use u to denote the truncation factor (Sec. G) and use η to denote the gradient update ratio (Sec. 3.4). On CIFAR-10 with ResNet18, we $(u, \eta) = (0.6, 0.7)$. On CIFAR-100 with ResNet18, we $(u, \eta) = (0.5, 0.7)$. On CIFAR-100 with WRN16, we $(u, \eta) = (0.6, 0.7)$. On CIFAR-100 with ResNext29, we $(u, \eta) = (0.7, 0.5)$.

C Robustness Evaluation on CIFAR-10 and CIFAR-100

We evaluate the robustness of models trained with various baseline DA methods. In particular, we measure the classification accuracy of models on test data perturbed using Gaussian noise ($\sigma^2 = 0.01$) and adversarial attacks. To craft adversarial perturbations, we use $\varepsilon = \frac{8}{255}$ for ℓ_∞ bounded FGSM [2] and $\varepsilon = 0.5$ for ℓ_2 bounded FGM [2]. Results are based on models trained with ResNet18. In Figure 1, we notice that models trained with SAGE achieve improved classification accuracy on both clean and noise-perturbed test data. However, method such as SaliencyMix, Puzzle Mix, Co-Mixup and CutMix improves generalization performance on the test data at the cost of decreased robustness.

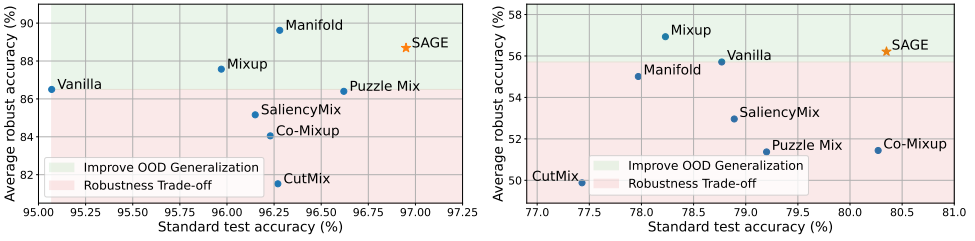
Perturbations	Vanilla	Mixup	CutMix	Manifold	SaliencyMix	Puzzle Mix	Co-Mixup	SAGE
Rank	4	3	8	1	6	5	7	2
FGSM (ℓ_∞)	79.96	80.93	79.57	**85.79**	80.62	81.96	78.78	<u>83.75</u>
FGM (ℓ_2)	89.67	89.22	87.81	**90.86**	88.86	89.64	88.11	<u>90.64</u>
Gaussian	89.88	**92.56**	77.2	<u>92.21</u>	85.99	87.60	85.25	91.67

Table 1: Classification accuracy on noise perturbed CIFAR-10 test data.

Perturbations	Vanilla	Mixup	CutMix	Manifold	SaliencyMix	Puzzle Mix	Co-Mixup	SAGE
Rank	3	1	8	4	5	7	6	2
FGSM (ℓ_∞)	49.24	**50.51**	44.2	48.89	46.52	44.58	44.32	<u>50.18</u>
FGM (ℓ_2)	62.19	**63.36**	55.57	61.14	59.4	58.16	58.56	<u>62.23</u>
Gaussian	52.68	**60.76**	28.06	<u>55.47</u>	38.21	43.96	34.46	47.68

Table 2: Classification accuracy on noise perturbed CIFAR-100 test data.

[1] https://github.com/afm-shahab-uddin/SaliencyMix
[2] https://github.com/snu-mllab/PuzzleMix
[3] https://github.com/snu-mllab/Co-Mixup



CIFAR-10 CIFAR-100

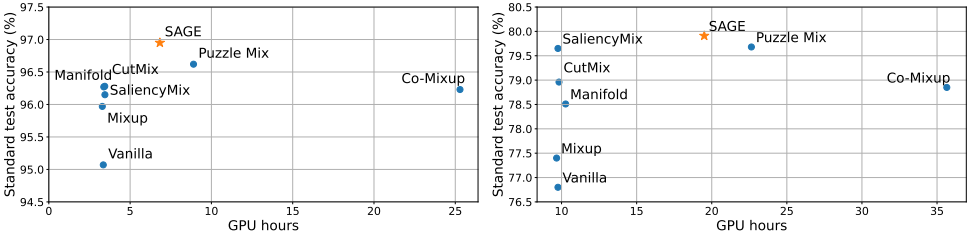
Figure 1: Visualization of the standard generalization performance vs. generalization in the OOD setting. We notice that SaliencyMix, CutMix, Co-Mixup and Puzzle Mix improves standard test accuracy over vanilla but at a cost of decreased robustness.

D Runtime Comparison

To estimate the computation cost of various baseline DA methods, we measure the total GPU hours required to train CIFAR-10 and CIFAR-100 using a single NVIDIA Tesla T4. Notice training with SAGE approximately doubles the time of vanilla training due to the computation of the saliency map; however, unlike Puzzle Mix and Co-Mixup, there is no additional overhead in finding the optimal rearrangements to maximize the total saliency. SaliencyMix stands apart from the other saliency-based augmentation techniques. This follows because it utilizes an external trained saliency detector based on a shallow pre-deep learning method [5], that is fast but considerably less capable than the deep saliency methods [6] used for the other augmentation techniques. Consequently, SaliencyMix introduces minimal overhead; however, its improvement on classification accuracy is limited.

Dataset	Model	Vanilla	Mixup	CutMix	Manifold	SaliencyMix	Puzzle Mix	Co-Mixup	SAGE
CIFAR10	PreActResNet18	3.35	3.29	3.38	3.44	3.45	8.9	25.29	**6.83**
CIFAR100	ResNext29	9.76	9.67	9.83	10.27	10.18	22.64	35.65	**19.5**

Table 3: GPU hours comparison of SAGE and other baselines.



CIFAR-10 CIFAR-100

Figure 2: Compared to other saliency-guided methods, SAGE achieves better standard test accuracy on both datasets with low computation overhead.

E Full SAGE Algorithm

Algorithm 1 shows the exact procedure of SAGE. We discuss saliency-guided mixing with optimal rearrangement (Ln 3) in Sec. 3.2, and the rest of the algorithm is covered in Sec. 3.1.

Algorithm 1: Data Augmentation based on SMART Mixup

Input : Pairs of training samples: (x_0, y_0) and (x_1, y_1), a classifier $f_\theta(\cdot)$, a loss function ℓ, a randomly sampled mix ratio λ, a Gaussian smoothing parameter σ^2 and $\mathcal{O}$ is the space of all possible image translations

Output : A new data-label pair: (x', y')

1 $s_0 = |\nabla_x \ell(f_\theta(x_0), y_0)|_{l_{2,3}}, s_1 = |\nabla_x \ell(f_\theta(x_1), y_1)|_{l_{2,3}}$

2 $\tilde{s}_0 = \lambda * \text{Smoothing}(s_0, \sigma^2)$, $\tilde{s}_1 = (1-\lambda) * \text{Smoothing}(s_1, \sigma^2)$

3 $\tau^* = \underset{\tau \in \mathcal{O}}{\text{argmax}}\ \nu(\tau)$, where $\nu(\tau)$ is defined in Eq. 4

4 $M^{\tau^*} = \frac{\tilde{s}_0}{\tilde{s}_0 + \mathcal{T}(\tilde{s}_1; \tau^*) + \zeta}$

5 $\gamma = \frac{1}{d^2} \sum_{i,j=1}^{d} M_{ij}^{\tau^*}$

6 $x' = M^{\tau^*} \odot x_0 + (1 - M^{\tau^*}) \odot \mathcal{T}(x_1; \tau^*)$

7 $y' = \gamma \cdot y_0 + (1-\gamma) \cdot y_1$

F Examples of Augmentation Results with SAGE w/o SM and SAGE w/o OR

In Sec. 4.4, we verified the effectiveness of our data augmentation strategy by ablating i) SAGE w/o OR (i.e., without optimal rearrangements) that always performs Saliency-guided Mixup on non-shifted images and ii) SAGE w/o SM (i.e., without Saliency-guided Mixup). Examples of the augmentation results are shown in Figure 3.

SAGE w/o OR SAGE w/o SM

Figure 3: Augmentation results with SAGE w/o OR and SAGE w/o SM

G Additional Ablations

We include two additional ablation studies in this section: i) reusing parameter gradients from un-augmented samples (Sec. 3.4) and ii) randomly rescaling of total saliency. For each

experiment, we use the best result as the control group (bold numbers), then we repeat the runs with modified task-related parameters.

η	CIFAR-10	CIFAR-100
0.5	96.65	79.36
0.7	**96.95**	**79.91**
1.0	96.58	79.24

u	CIFAR-10	CIFAR-100
0.5	96.75	**79.91**
0.6	**96.95**	79.7
1.0	96.6	79.29

Table 4: **Additional ablation studies of SAGE.** (left) Test accuracy of models trained with combined parameter gradients from un-augmented and SAGE-augmented samples. (right) Test accuracy of models trained with truncated total saliency.

Reusing the parameter gradients: In Sec. 3.4, we discuss performing gradient descent update by combining parameter gradients computed on un-augmented and SAGE-augmented samples. In particular, let g_s and g_a represent the gradients computed using un-augmented and augmented images, respectively. The final model update is based on $g = \eta \cdot g_s + (1-\eta) \cdot g_a$, where $\eta \in [0,1]$. In Table 4, we observe reusing the parameter gradients computed on un-augmented samples ($\eta \neq 1$) significantly increases accuracy on the test data.

Random rescaling of total saliency: A random mixing ratio in prior work [1, 3, 4, 8, 9] can be seen as a way to increase diversity of the augmentation results. Similarly, we randomly rescale the total saliency of smoothed s_0 and s_1 using $\lambda \sim \mathcal{U}(0,1)$ and $1-\lambda$ respective. In practice, we observe the diversity in the augmented images greatly decreases when $\lambda > 0.6$, since x_0 and $\tilde{s}_0$ dominate when computing the total saliency. Therefore, when the offset images are rescaled to having a small total saliency, it is often better to just exclude it in the augmented results. As such, we propose a simple heuristic to truncate the random rescaling factor: $\lambda \sim \mathcal{U}(0,u)$, where $u \in [0,1]$. Results in Table 4 shows with $u < 1.0$, the test accuracy on both datasets increase significantly.

References

[1] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

[2] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *International Conference on Learning Representations (ICLR)*, 2015.

[3] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle Mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning (ICML)*, 2020.

[4] JangHyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-Mixup: Saliency guided joint mixup with supermodular diversity. In *International Conference on Learning Representations (ICLR)*, 2021.

[5] Sebastian Montabone and Alvaro Soto. Human detection using a mobile platform and novel features derived from a visual saliency mechanism. In *Image and Vision Computing*, 2010.

[6] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[7] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold Mixup: Better representations by interpolating hidden states. In *International Conference on Learning Representations (ICLR)*, pages 6438–6447, 2019.

[8] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. CutMix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[9] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. Mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018.