Expeditious Saliency-guided Mix-up through Random Gradient Thresholding

Anonymous submission

Abstract

Mix-up training approaches have proven to be effective in improving the generalization ability of Deep Neural Networks. Over the years, the research community expands mix-up methods into two directions, with extensive efforts to improve saliency-guided procedures but minimal focus on the arbitrary path, leaving the randomization domain unexplored. In this paper, inspired by the superiority qualities of each direction over one another, we introduce a novel method that lies at the junction of the two routes. By combining the best elements of randomness and saliency utilization, our method balances speed, simplicity, and accuracy. We name our method R-Mix following the concept of "Random Mix-up". We demonstrate its effectiveness on generalization, weakly supervised object localization, calibration, and robustness to adversarial attack. Finally, in order to address the question whether there exists a better decision protocol, we train a Reinforcement Learning agent that decides the mix-up policies based on the classifier's performance, reducing dependency on human-designed objectives and hyperparameter tuning. Extensive experiments further show that the agent is capable of performing at the cutting-edge level, laying the foundation of fully automatic mix-up.

Introduction

Mix-up, a data augmentation strategy to increase a deep neural network (DNN)'s predictive performance, has drawb a lot of attention in recent years, along with the numerous initiatives made to pushing various deep learning models to move up the state-of-the-art leaderboard on multiple benchmarks and different applications. The pioneering idea, Input Mix-up, introduced by (Zhang et al. 2018), simply interpolates two samples in a linear manner and has been proven to play a significant role in improving a model's predictive performance with hardly any additional computing cost. Recently, theoretical explanations for how Input Mix-up enhances robustness and generalization have been studied (Zhang et al. 2021b).

Building upon the empirical success of these mix-up methods, the community has explored multiple directions to further improve the mix-up idea. Manifold Mixup (Verma et al. 2019) extends the original mix-up by mixing at a random layer in the model. AugMix (Hendrycks et al. 2020) first augments the images by different combinations of augmentation techniques, then finally mixes them together to increase the robustness of DNNs. CutMix (Yun et al. 2019) uses spatial copy and paste-based strategy on other samples to create the new mixed-up sample and has also been used widely in various applications.

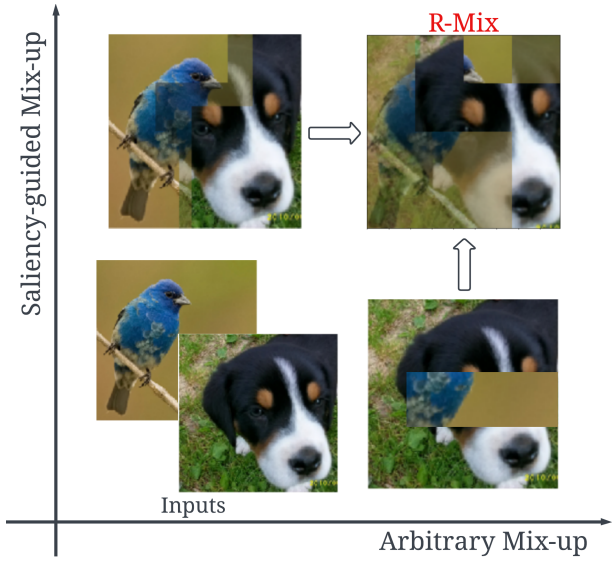

Figure 1: Illustration of our proposed method R-Mix. Arbitrary Mix-up linearly interpolates images or employs cut-and-paste strategy. Saliency-guided Mix-up preserves the rich supervisory signals of the images. Our method R-Mix works by combining finest aspects of both approaches and demonstrates its effectiveness on a variety of tasks.

Among the rich family of mix-up extensions, a popular branch in it is mix-up methods that leverage the information of *saliency maps*, because intuitively, one way to improve the efficiency of mix-up would be to replace its random procedure with a directed procedure guided by some additional knowledge, and a saliency map appears to be a natural choice of such knowledge.

Probably driven by the same intuition, the community has investigated the saliency-based mix-up idea deeply in recent years, such as SaliencyMix (Uddin et al. 2021), PuzzleMix (Kim, Choo, and Song 2020), and Co-Mixup (Kim et al. 2021). SaliencyMix generates the saliency map, and

then employs the cut-and-paste strategy of CutMix. PuzzleMix further introduces secondary optimization objectives that first optimize the saliency map, then optimize the transport plan in order to preserve the rich supervisory signals of the image. Co-Mixup extends PuzzleMix's idea by further introducing objectives to find the most suitable image to mix in the whole batch.

We observe that each "direction" of image combining has its own advantages and disadvantages. **Arbitrary Mix-up** techniques, such as Input Mix-up and CutMix, offer fast training speed and simplicity while maintaining competitive performance. Contrarily, **Saliency-guided Mix-up**, like PuzzleMix and Co-Mixup, compromises speed and simplicity in favor of accuracy, expected calibration error, and robustness to adversarial attack. Over time, significant efforts have been proposed to further improve the saliency-guided direction with minimal focus on the other (Uddin et al. 2021; Kim, Choo, and Song 2020; Kim et al. 2021; Venkataramanan et al. 2022), resulting in an unexplored randomness domain. We raise the question: *is it feasible to have a method that is expeditious, simple, and effective at the same time?*

In this paper, we identify a straightforward learning heuristic that sits in the middle of two paths. Our throughout examination of saliency-guided methodologies suggests that they typically fall under a three-step optimization: First, calculate the saliency of the image. Then, mix the images in accordance with a secondary optimization objective. Finally, train the DNN with the mixed images and labels. Roughly speaking, all three levels require the same amount of training time, making the training takes at least three times longer. We notice that, by swapping out the second step with a randomness-driven mixing approach, we are able to design a strategy that gives competitive performance with state-of-the-art methods while maintaining the speed and ease of implementation of arbitrary mix-up. We name our method **R-Mix** and empirically validate its performance on **four different tasks**: image classification, weakly supervised object localization, expected calibration error, and robustness to adversarial attack. On all benchmarks, R-Mix shows an improvement or on-par performance with state-of-the-art methods.

In summary, our contributions in this paper are as follows:

- We begin by demonstrating that our implementation of arbitrary mix-up is capable of outperforming saliency-guided mix-up, indicating that existing attempts have not yet fully investigated the effectiveness of randomization. (Section: Background and Motivation).
- Motivated from the superiority of each mix-up direction over one another, we propose a novel method R-Mix that combines the two mix-up routes and eliminates a third of the computational complexity (Section: R-Mix). With regard to four benchmarks on different model architectures: image classification, weakly supervised object localization, robustness to adversarial attack, and expected calibration error (Section: Experiments), we highlight that R-Mix performs better or equally well as state-of-the-art approaches.
- Finally, to answer the question whether coupled randomness and saliency are sufficient to the gain of R-Mix or there exists a superior decision protocol, we present several experiments in the case of Reinforcement Learning controlled mix-up scenario. Our Reinforcement Learning agent adapts and chooses the mix-up rules based on the performance of the classifiers, aiming to reduces reliance on human-designed objectives and hyperparameter tuning of mix-up in general. We validate its effectiveness on CIFAR-100 image classification task, and find that it performs competitively with other baselines. (Section: Ablation Studies).

Background and Motivation

In this Section, we provide background knowledge about mix-up training, and empirical results serving as motivation for our method.

Mix-up Background

Let C, W, H, N denote the number of channels, image width, image height, and number of classes, respectively. We assume that $W = H$ for simplicity, and will use only W from now on. Let $x \in \mathcal{X}, x \in \mathbb{R}^{C\times W\times W}$ be the input image and $y \in \mathcal{Y}, y \in \{0,1\}^N$ be the output label. Let $f(\cdot;\theta_c)$ denote a classifier specified by parameter θ_c. Let $\mathcal{D}$ be the distribution over $\mathcal{X}\times\mathcal{Y}$. In mix-up based data augmentation, the goal is to optimize the model's loss $\ell : \mathcal{X}\times\mathcal{Y}\times\Theta\to\mathbb{R}$ given the mix-up function for the inputs $h(\cdot)$, for the labels $g(\cdot)$, and the mixing distribution, usually $Beta(\alpha,\alpha)$ with the scalar parameter α, as follows:

$$\underset{\theta}{\text{minimize}} \mathop{\mathbb{E}}_{(x_0,y_0),(x_1,y_1)\in\mathcal{D}} \mathop{\mathbb{E}}_{\lambda\sim Beta} \ell(h(x_0,x_1), g(y_0,y_1);\theta_c)) \tag{1}$$

Mix-up typically requires two tuples of images-labels. Input Mix-up (Zhang et al. 2018) defines $h(x_0,x_1) = \lambda x_0 + (1-\lambda)x_1$ and $g(x_0,x_1) = \lambda y_0 + (1-\lambda)y_1$. Manifold Mixup (Verma et al. 2019) extends Input Mix-up by mixing the inputs at a hidden representation F as: $h(x_0,x_1) = \lambda F(x_0)+(1-\lambda)F(x_1)$, that is, at a random layer of $f(\cdot;\theta_c)$. CutMix randomly copies a rectangular region from x_0 and pastes it to x_1. PuzzleMix (Kim, Choo, and Song 2020) uses $h(x_0,x_1) = z\odot\Pi^T x_0 + (1-z)\odot\Pi'^T x_1$ where Π is a transport plan, z is a binary mask and $\odot$ is element-wise multiplication. Co-Mixup (Kim et al. 2021) extends $h(\cdot)$ to operate on a batch of data instead of two pairs: $h(x_B)$.

While some early techniques simply mix the images using weights sampled from the $Beta$ distribution (Input Mix-up, Manifold Mixup), or choose an arbitrary rectangular region and then apply mix-up (CutMix), both PuzzleMix and Co-Mixup require additional optimization objectives to ensure rich supervisory signals, introducing heavy computational cost.

Motivation

In this Section, we provide empirical evidence demonstrating the usefulness of randomization on generalization ability. CutMix (Yun et al. 2019), which randomly cuts a rectangular patch from one image and pastes it into another, is

Scheduler	MultiStepLR	OneCycleLR
CutMix (300)	78.71	**80.60**†
PuzzleMix (300)	79.38	79.75
Co-Mixup (300)	**80.13**	79.79
PuzzleMix (1200)	80.36	80.48
Co-Mixup (1200)	80.47	80.30

Table 1: Top-1 Accuracy on CIFAR-100 using PARN-18 with diffferent learning rate schedulers and training epochs. Bold indicates the best result. † denotes the CutMix+ version we will use throughout this paper.

typically justified on the grounds that it can create abnormal images by unintentionally choosing the fragments that do not contain any information about the source object (for example, cutting the grass-only region in an image of a cow on grass), which results in the so-called "learning false feature representations". Recent saliency-based methods aim to solve the issue by enhancing the saliency of the combined pictures, and report an increase in performance (Uddin et al. 2021; Kim, Choo, and Song 2020; Venkataramanan et al. 2022; Kim et al. 2021). Naturally, one may think that saliency is the main contributing factor to this increment. However, we provide several counter-examples suggesting that this notion is only partially persuasive, as randomness is still essential for generalization.

- **OneCycleLR elevates arbitrary-mixup to cutting-edge tier.** We simply change another LR scheduler instead of using the default MultiStepLR, specifically the OneCycleLR scheduler (Smith and Topin 2018) and reproduce CutMix. We denote this method as CutMix+. We train PreActResNet-18 (PARN-18) (He et al. 2016b) on CIFAR-100 for 300 epochs. From Table 1 (first row), CutMix+ performs better than the most advanced saliency-guided mix-up by increasing accuracy by 1.89%. Swapping another LR scheduler takes a few lines of code and introduces no additional computational cost.
- **OneCycleLR does not help saliency-guided mix-up.** We evaluate the performance of the OneCycleLR scheduler by using hyperparameters from the prior CutMix+ experiment and reproduce these two saliency-guided methods again for 300 epochs. Table 1 (right column) demonstrates that OneCycleLR improves PuzzleMix by 0.37% accuracy, but not Co-Mixup.
- **Training saliency-guided mix-up for four times as many epochs still underperform CutMix+**. We run PuzzleMix and Co-Mixup for 1200 epochs using both schedulers and compare the results. Table 1 (last two rows) further shows that despite the improvement, both methods still fall short of CutMix+.
- **The finding is in line with other model architectures**. Finally, we further solidify the findings by running CutMix+ on three more model architectures: WideResNet (WRN) 16-8 (Zagoruyko and Komodakis 2017), ResNeXt29-4-24 (Xie et al. 2016) for 300 epochs, and WRN 28-10 for 400 epochs following the original implementations. Table 2 shows that CutMix+ consistently bests other state-of-the-art methods up to 2.08% accuracy.

Method	CutMix	PuzzleMix	Co-Mixup	CutMix+
PARN18	78.71	79.38	80.13	**80.60**
WRN16-8	79.86	80.76	80.85	**81.79**
WRN28-10	82.50	**84.05**	-	83.97
RNX	78.14	78.88	80.22	**82.30**

Table 2: Top-1 Accuracy (%) on CIFAR-100 with various model architectures. Higher is better. Bold indicates the best result.

These empirical results hint that although CutMix, and arbitrary mix-up in general, may produce distorted visuals, they are not as problematic in practice as they would appear. We then build a method with the following objectives as our driving force:

1. **Experimenting with OneCycleLR scheduler**: CutMix simply requires a different LR scheduler to perform better. While MultiStepLR clearly helps PuzzleMix and Co-Mixup, OneCycleLR helps CutMix and provides little or no performance benefit for the other two. In this work, OneCycleLR serves as the primary engine for our experiments.
2. **Un-natural images help in generalization**: Complex mix-up techniques only produce marginal performance benefits when training for extended period of time, and CutMix still remains a competitive method. Despite the fact that maximizing the saliency generates better-looking images for human eyes (Kim, Choo, and Song 2020; Kim et al. 2021), CutMix+'s performance suggests that it may not be the optimal way to mix images. Instead, striking a balance between the most and least salient regions by combining randomness and saliency may yield a more promising outcome.
3. **Low computational overhead**: Recent saliency-based mix-up algorithms have an excessively high computational cost. For instance, if all factors are held constant, Co-Mixup takes 15 hours and PuzzleMix takes 27 hours, meanwhile Vanilla, CutMix (and CutMix+ variation) training takes approximately about 2.5 hours. Having an alternate mix-up method that compromises between simplicity, performance, and computing cost will tremendously benefit low-resource academic labs, businesses, and competitors in the data science field, where limited hardware is provided.

R-Mix: An Expeditious Saliency-based Mixup

In our proposed R-Mix, we extend Input Mix-up and CutMix to the patch level but also utilize saliency information. We break down our method in four main steps.

(1) Generating the Saliency Map: First, we compute the saliency map $\phi(x)$ as the gradient values of training loss with respect to the input data and measure the ℓ_2 norm

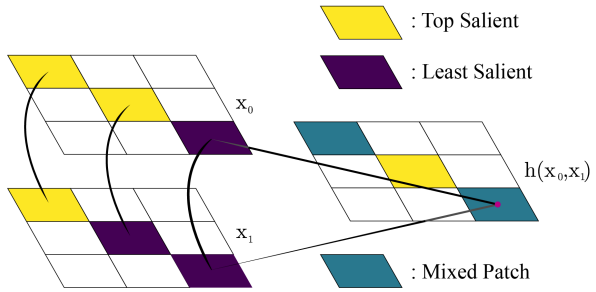

Figure 2: Illustration of R-Mix mix-up process. Given two images (x_0, x_1) and their saliency maps, for two patches at the same position, if they both belong to the top (yellow) and least (purple) salient regions, we mix the patch (blue), else we only select the top salient one. Best viewed in color

across the input channels (Simonyan, Vedaldi, and Zisserman 2014):

$$\phi(x) = \sqrt{\sum_{i=0}^{C} \frac{1}{C} (\nabla^i_{\theta_c} \ell(x, y; \theta_c))^2} \tag{2}$$

where $\nabla^i_{\theta_c} \ell(x, y; \theta_c)$ denotes the gradient at channel i.

Second, we normalize $\phi(x)$ so that all elements of the map sum up to 1, and down-sample it to size $p \times p$, where p is arbitrary chosen as a multiple of 2. Intuitively, normalizing and down-sampling ensure numerical stability and decrease compute cost for the subsequent operations. Moreover, choosing a random p for each batch enhances sample diversity. Specifically:

$$\phi'(x) = \text{AvgPool}\left(\frac{\phi(x)}{\sum \phi(x)}, \text{kernel\_size} = p, \text{stride} = p\right) \tag{3}$$

(2) Splitting the Saliency Map into two regions: Next, we randomly partition $\phi'(x)$ into two regions: the most and least salient regions, using the percentile value. For the top-k space $\mathcal{A}$ with K equally spaced values from 0.0 to 0.99, we sample a value $q \in \mathcal{A}, q \in [0, 0.99]$. We compute the q-th percentile value of the down-sampled saliency map $\phi'(x)$, denoted as q_{perc} and construct a binary mask m as follows. For the i-th element in $\phi'(x)$:

$$m(i) = \begin{cases} 1, & \text{if } \phi'_i(x) \geq q_{\text{perc}} \text{ (\textbf{top} salient region)} \\ 0, & \text{otherwise (\textbf{least} salient region)} \end{cases} \tag{4}$$

After construction, mask m is upsampled by replicating elements to match the size of the inputs x. After this step, we obtain the mask of top-least salient regions of the inputs.

(3) Creating the soft mix-up filter: For a pair of (x_0, x_1), we obtain (m_0, x_0) and (m_1, x_1). We construct another mask m_{inter} (inter stands for *intersection*) as follows. For the i-th element in m:

$$m_{\text{inter}}(i) = \begin{cases} 1, & \text{if } m_0(i) = m_1(i) \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

	Mix-up	Cutout	CutMix	Co-Mixup	R-Mix
Speed	1/1	1/1	1/1	3/1	2/1
Top Salient	?	?	Mixed	Yes	Yes
Least Salient	?	?	Mixed	No	Yes
Full Image	Yes	No	Yes	Yes	Yes
Dropout	No	Yes	Yes	Yes	Yes
Mixed (x, y)	Yes	No	Yes	Yes	Yes
2nd objective	No	No	No	Yes	No

Table 3: Major distinctions between R-Mix and other techniques. Training speed is measured on the same GPU.

The value of the element is 1 if the two corresponding patches both belong to the top and least salient regions, and 0 otherwise (Figure 2).

(4) Mixing the images and labels: Finally, we sample the mixing coefficient $\lambda \sim Beta(\alpha, \alpha)$, then our mix-up function is defined as:

$$\begin{aligned} h(x_0, x_1) = m_{\text{inter}} \odot (\lambda x_0 + (1 - \lambda) x_1) \\ + \neg m_{\text{inter}} \odot (m_0 \odot x_0 + m_1 \odot x_1) \end{aligned} \tag{6}$$

where $\neg$ denotes the logical NOT operator, that is, the binary mask is flipped. In short, for the i-th element in m_0 and m_1, we mix the element if $m_0(i) = m_1(i)$ (analogous to Input Mix-up). For the elements that $m_0(i) \neq m_1(i)$, we use $m_{\text{inter}}(i) = \max(m_0(i), m_1(i))$ (analogous to CutMix). Note that m is a binary mask.

Let $c(m)$ denote the number of elements that are active, that is, $c(m) = |\{i | m(i) = 1\}|$ where $|\cdot|$ denotes the cardinality of a set. The label mix-up function is defined as:

$$\begin{aligned} g(y_0, y_1) = \frac{c(m_{\text{inter}})}{W \times W} (\lambda y_0 + (1 - \lambda) y_1) \\ + \frac{c(\neg m_{\text{inter}} \odot m_0) y_0 + c(\neg m_{\text{inter}} \odot m_1) y_1}{W \times W} \end{aligned} \tag{7}$$

This label mix-up function takes into account both the mix-up λ, and how many patches of the image are mixed.

In practice, all operations can operate on a batch level, with the current batch being randomly permuted to obtain the other input. The mixed sample produced by Equation 6 and 7 is used to train the classifier $f(\cdot; \theta_c)$ to minimize the soft target labels by minimizing the multi-label binary cross-entropy loss in Equation 1.

We list the major distinctions between R-Mix and alternative techniques in Table 3. In summary, our method R-Mix has fast training speed, utilizes the entire image, but has no additional optimization objective. Figure 2 further illustrates the mix-up process of two images on the patch level.

Experiments

In this Section, we describe the datasets, models and training pipelines to benchmark our method on for different tasks: Image Classification, Weakly Supervised Object Localization, Expected Calibration Error, and Robustness to Adversarial Attack.

Method	Vanilla	Input	Manifold	CutMix	PuzzleMix	Co-Mixup	**CutMix+**	**R-Mix**
PARN-18	76.41	77.57	78.36	78.71	79.38	80.13	80.60	**81.49**
WRN16-8	78.30	79.92	79.45	79.86	80.76	80.85	81.79	**82.32**
WRN28-10	78.86	81.73	82.60	82.50	84.05	-	83.97	**85.00**
RNX	78.21	78.30	77.72	78.14	78.88	80.22	82.30	**83.02**

Table 4: Top-1 Accuracy (%) on CIFAR-100 with various models and methods trained for 300 epochs. Higher is better. Bold indicates the best result.

Datasets. We test our methods on two standard classification dataset benchmarks. CIFAR-100 (Krizhevsky 2009) contains 50k images of size 32×32 for training and 10k images for validation, equally distributed among 100 classes. ImageNet (Russakovsky et al. 2015) has 1.3M images for training distributed among 1k classes and has 100k images for validation. We normalize the data channel-wise, and average the results over 10 runs on CIFAR-100, 5 runs on ImageNet. Similar to earlier works, traditional augmentations, such as Random Horizontal Flip and Random Crop with Padding, are employed.

Model Architecture. To remain consistent with earlier works, we use five different model architectures to test our method. We use PreActResNet-18 (PARN18) (He et al. 2016b), Wide Res-Net (WRN) 16-8 and 28-10 (Zagoruyko and Komodakis 2017), and ResNeXt 29-4-24 (RNX) (Xie et al. 2016) on CIFAR-100. For ImageNet we use ResNet-50 (He et al. 2016a).

Pipeline and Hyperparameters. For CIFAR-100, we set $p \in \{2, 4\}, K = 10, \alpha = 1.0$ and use OneCycleLR scheduler with initial LR $3e-3$, max LR 0.3 and final LR $3e-5$, increasing for 30% of the total number of epochs. We train for a total of 300 epochs with batch size 100. For ImageNet we use the identical protocol (such as image size and LR scheduler) described in PuzzleMix and Co-Mixup, which trains ResNet-50 for 100 epochs. We set $p \in \{2, 4\}, K = 10, \alpha = 0.2$.

Image Classification

For fair comparison, we include results that were reported using the same training pipeline, that are: Input Mix-up (Zhang et al. 2018), Manifold Mixup (Verma et al. 2019), CutMix (Yun et al. 2019), PuzzleMix (Kim, Choo, and Song 2020), Co-Mixup (Kim et al. 2021), but add other methods with different pipelines for comparison in the Appendix. All methods are trained using PARN-18, WRN16-8, and RNX on CIFAR-100 for 300 epochs, except WRN28-10 is trained for 400 epochs.

From Table 4, R-Mix outperforms CutMix by 2% and CutMix+ by 1% on average. It outperforms Co-Mixup by 1.47% with WRN16-8, by 0.85% with WRN28-10 and by 2.8% with RNX. As noted in other works (Zhang et al. 2018), mix-up methods generally benefit more from models with higher capacity, explaining the higher gain on bigger models.

We further test R-Mix on ImageNet (ILSVRC 2012) dataset (Russakovsky et al. 2015). We use the same training protocol as specified in Co-Mixup which trains ResNet-50 for 100 epochs. Table 5 shows that R-Mix shows an improvement over Vanilla by 1.44% and CutMix by 0.28%.

Metric	Accuracy	Localization
Vanilla	75.97	54.36
Input	77.03	55.07
Manifold	76.70	54.86
CutMix	77.08	54.91
PuzzleMix	77.51	55.22
Co-Mixup	**77.61**	55.32
R-Mix	77.41	**55.58**

Table 5: Top-1 Accuracy and Localization Accuracy (%) on ImageNet using ResNet-50 trained for 100 epochs. Higher is better. Bold indicates the best result.

Weakly Supervised Object Localization

Weakly Supervised Object Localization (WSOL) aims to localize an object of interest using only class labels without bounding boxes at training time. WSOL operates by extracting visually discriminative cues to guide the classifier to focus on prominent areas of the image.

We compare WSOL performance of classifiers trained on ImageNet to demonstrate that, despite the fact that R-Mix produces un-natural images, it is *more effective* in focusing on salient regions compared to other saliency-guided methods. From Table 5, using the Class Activation Map method (Zhou et al. 2015) and the protocol described in Co-Mixup, it is interesting that, even with a lower Top-1 Accuracy, our method *increases* the Localization Accuracy by 0.26% and outperforms all other baselines. This further suggests that by striking a balance between the most and least salient regions, R-Mix better guides the classifier to focus on salient regions.

Metric	ECE	FGSM
Vanilla	10.25	87.12
Input	18.50	81.30
Manifold	7.60	80.29
CutMix	18.41	86.96
PuzzleMix	8.22	78.70
Co-Mixup	5.83	77.61
R-Mix	**3.73**	**77.08**

Table 6: Expected Calibration Error (ECE) (%) and Top-1 Error Rate (%) of PARN-18 to FGSM attack. Lower is better.

Expected Calibration Error

We evaluate the expected calibration error (ECE) (Guo et al. 2017) of PARN-18 trained on CIFAR-100. ECE is calculated by the weighted average of the absolute difference between the confidence and accuracy of a classifier. From Table 6, we show that while Arbitrary Mix-up methods tend to have *under-confident* predictions, resulting in higher ECE value, Saliency-guided Mix-up methods tend to have best-calibrated predictions. Our method R-Mix successfully alleviates the over-confidence issue and does not suffer from under-confidence predictions.

Robustness to Adversarial Attack

Adversarial Attack attempts to trick DNNs into classifying an object incorrectly by applying small perturbations to the input images, resulting in an indistinguishable image for the human eye. (Szegedy et al. 2013). Following previous evaluation protocol (Kim, Choo, and Song 2020), we evaluate PARN-18 model's robustness to FGSM adversarial attack with $8/255$ ℓ_∞ ϵ-ball. As shown in Table 6, we observe that Saliency-guided methods have lower FGSM error. By leveraging this Saliency information, R-Mix further establishes the best result among other competitors by lowering the Error Rate by 0.53%.

Computational Analysis

We compare the wall time on CIFAR-100 and ImageNet by investigating the released checkpoints and reproducing experiments. Specifically, including training and validation at each epoch, for **CIFAR-100** with batch size 100, Co-Mixup takes 15 hours on one RTX 2080Ti, whereas R-Mix takes **4.0** hours. For **ImageNet** with 4 RTX 2080Ti, vanilla training takes 0.4s per batch, R-Mix takes 0.77s per batch while Co-Mixup takes 1.32s per batch. It should be noted that that the saliency map is built on the gradient information (Simonyan, Vedaldi, and Zisserman 2014) which requires two passes to the classifier. As a result, the running time is expected to be twice as long as with vanilla training. During validation, all classifiers need the same amount of time.

Ablation Studies

We conduct ablation studies about hyperparamter sensitivity and experiments about a mix-up method that automatically decides the mix-up policies based on the model's performance, with the goal of laying the groundwork for future mix-up methods that require minimal human-designed objectives and low hyperparameter tuning effort.

Sensitivity to Hyperparameters.

Number of patches p and top-k space K. We conduct hyperparameter tuning with different choices of the downsampling Kernel Size p and the top-k space that consists of K equally-spaced values from 0.0 to 0.99 on CIFAR-100. We then use the best found combination: $K = 10, p \in \{2, 4\}$ to report the final result as in previous Tables and Figures. We report the result in Table 7. We observe that, the higher the value p, the less efficient the method is. We hypothesize that, since each image patch has its own mix-up rule depending on the "other" patch, thus the higher the p value, the higher the probability that a patch has different mixing rules compared to its neighbor patches. This diversity "breaks" the connectivity of the patches, which in turn hurts the convolution operations.

	K=5	K=10	K=16	K=20
$p \in \{2, 4\}$	81.22	81.49	81.35	81.18
$p \in \{4, 8\}$	80.98	80.80	80.30	80.54
$p \in \{8, 16\}$	79.94	80.33	79.29	79.50
$p \in \{16, 32\}$	79.60	79.06	78.86	79.20
$p \in \{2, 32\}$	79.64	79.69	79.21	79.64
$p \in \{2, 4, 8\}$	80.87	80.34	80.04	80.94
$p \in \{4, 8, 16\}$	80.12	80.37	80.34	80.41
$p \in \{8, 16, 32\}$	79.53	79.12	79.55	79.67

Table 7: Top-1 Accuracy on CIFAR-100 using PARN18 with different choices of hyperparameters. Higher is better.

Mixing parameter α. We then conduct sensitivity analysis on the mixing parameter α used in sampling weights from the Beta distribution on CIFAR-100 with PARN-18 model. Table 8 shows that for the majority of options, R-Mix is still ourperforming other baselines and only suffers from minor accuracy lost, demonstrating its robustness to hyperparameter tuning.

	$\alpha = 0.2$	$\alpha = 0.5$	$\alpha = 1.0$	$\alpha = 2.0$
R-Mix	81.29	81.40	81.49	81.01

Table 8: Top-1 Accuracy of R-Mix on CIFAR-100 with different α values. Higher is better.

Is Randomness Enough? Reinforcement Learning-Powered Decisions with RL-Mix.

In this section, we perform early experiments in an attempt to answer the question "whether coupled randomness and saliency are sufficient to the gain of R-Mix or there exists a superior decision protocol" using Reinforcement Learning. Inspired by AutoAugment (Cubuk et al. 2019), we use the Proximal Policy Optimization (Schulman et al. 2017) from Stable-Baselines3 (Raffin et al. 2019) using default hyperparameters suggested by a large-scale study (Andrychowicz et al. 2021). With the inputs as the saliency map $\phi'(x)$ and the logits, the agent determines the top-k value for each image in a batch. An episode of the agent ends when the classifier $f(\cdot, \theta_c)$ finishes training one epoch. Since the agent requires a fixed input size, we arbitrary choose $p = 8$. Based on the findings from Deep AutoAugment (Zheng et al. 2022), the reward function is the cosine similarity between the gradients of the original input x and the mixed input x', that is, $CosSim(\phi(x), \phi(x'))$. We call this method **RL-Mix**.

Table 9 reports the result of RL-Mix and other baselines. We can see that in most cases, R-Mix is still better than RL-Mix. Interestingly, with a fixed size of p and no hyperparameter tuning, it is still capable of delivering good perfor-

Model	CutMix	CutMix+	R-Mix	RL-Mix
PARN-18	78.71	80.60	**81.49**	80.75
WRN16-8	79.86	81.79	**82.32**	82.16
WRN28-10	82.50	83.97	**85.00**	84.90
RNX	78.14	82.30	**83.02**	82.43

Table 9: Top-1 Accuracy of RL-Mix on CIFAR-100 trained for 300 epochs. Higher is better.

mance. On the same GPU used throughput the paper, *RL-Mix* is slower than R-Mix by 2.0 times with a runtime of 7.5-8 hours.

Although RL-Mix is only early work, we believe it has the potential to open a new research direction of fully automatic mix-up, a branch in AutoML that requires minimal human-designed objectives and has low hyperparameter tuning effort.

Related Work

Saliency Maps

There have been many works towards interpretability techniques for trained neural networks in recent years. Saliecny maps (Simonyan, Vedaldi, and Zisserman 2014) and Class Activation Maps (Zhou et al. 2015) have focused on explanations where decisions about single images are inspected. The work of (Simonyan, Vedaldi, and Zisserman 2014) generates the saliency map directly from the DNN without any additional training of the network by using the gradient information with respect to the label. Following it, (Zhao et al. 2015) measures the saliency of the data using another neural network, and (Zhou et al. 2016) aims to reduce the saliency map computational cost. We follow the method from (Simonyan, Vedaldi, and Zisserman 2014), which generates a saliency map without any modification to the model.

Data Augmentation

Data Augmentation is a technique to increase the amount of training data without additional data collection and annotation costs. There are two types of data augmentation techniques popularly used in various vision tasks: (1) transformation-based augmentation on a single image, and (2) mixture-based augmentation across different images.

Transformations on a single image. Geometric-based augmentation and photometric-based augmentation have been widely used in computer vision tasks. Survey papers (Halevy, Norvig, and Pereira 2009; Sun et al. 2017; Shorten and Khoshgoftaar 2019) show that inexpensive data augmentation techniques such as applying random flip, random crop, random rotation, etc., increase the diversity of the data and the robustness of the DNNs, and have been widely adopted in popular deep learning frameworks.

Mixture across images. (i) Mixture of images with a pre-defined distribution. Input Mix-up (Zhang et al. 2018) is a simple augmentation technique that blends two images by linearly interpolating them, and the labels are reweighted by the blending coefficient sampled from a distribution. Manifold Mixup (Verma et al. 2019) extends Input Mix-up to the perturbations of embeddings. CutMix (Yun et al. 2019) randomly copies a rectangular-shaped region of an image, and pastes it to a region of another image; **(ii) Mixture through Saliency Maps.** Saliency-based mixtures, such as PuzzleMix (Kim, Choo, and Song 2020), Co-Mixup (Kim et al. 2021), and SaliencyMix (Uddin et al. 2021) first generate a saliency map, and then use the map to optimize secondary objective functions that maximize the saliency to mix the images and ensure reliable supervisory signals.

For a more comprehensive summary of recent mix-up methods (Guo, Mao, and Zhang 2019; Harris et al. 2020; Faramarzi et al. 2020; Qin et al. 2020; Hendrycks et al. 2021; Zhou et al. 2021; Li et al. 2021; Dabouei et al. 2021a; Venkataramanan et al. 2022; Liu et al. 2022b; Dabouei et al. 2021b; Park et al. 2022; Liu et al. 2022a), we refer readers to the survey paper (Naveed 2021).

Deep Neural Networks Training Techniques

Techniques such as Weight Decay (Goodfellow, Bengio, and Courville 2016), Dropout (Srivastava et al. 2014), Batch Normalization (Ioffe and Szegedy 2015), and Learning Rate schedulers are widely used to efficiently train deep networks. The literature of learning rate (LR) scheduler is now nearly as extensive as that of optimizers (Schmidt, Schneider, and Hennig 2021). Generally, the training is divided into multiple phases. The LR of the classifier is kept constant during a phase, and then is decayed by a positive value in the next phase. One of the most common scheduler is MultiStepLR (Goodfellow, Bengio, and Courville 2016; Zhang et al. 2021a) or step-wise decay, which divides the training into phases where each consists of tens or hundreds of epochs. OneCycleLR, introduced in (Smith and Topin 2018) employs the cyclic learning rate scheduler (Smith 2017) but only for one cycle. The LR starts with a small value, increases to the max value then gradually decreases to an even smaller value until training finishes.

In this paper, we show that the LR scheduler can have a large impact on the performance of existing mix-up methods, sometimes removing any performance gains of more sophisticated mix-up strategies compared to vanilla mix-up strategies.

Conclusion

In this paper, we show that randomization is capable of performing at the cutting-edge tier, suggesting an unexplored domain in recent advances of mix-up research. Driven by the effectiveness of a mix-up research path over one another, we propose R-Mix, a simple training heuristic that lies at the junction of the two routes. Extensive experiments on image classification, weakly supervised object localization, calibration, and robustness to adversarial attack show a consistent improvement or on-par performance with state-of-the-art methods, while offering speed and simplicity of Arbitrary Mix-up. Finally, we describe RL-Mix, an early experiment of a Reinforcement Learning - powered agent to automatically decides the mixing regions based on the performance of the classifier, which has shown a competitive capability on CIFAR-100, laying the foundation of low-effort hyperparameter tuning mix-up.

References

Andrychowicz, M.; Raichuk, A.; Stańczyk, P.; Orsini, M.; Girgin, S.; Marinier, R.; Hussenot, L.; Geist, M.; Pietquin, O.; Michalski, M.; Gelly, S.; and Bachem, O. 2021. What Matters for On-Policy Deep Actor-Critic Methods? A Large-Scale Study. In *ICLR*.

Cubuk, E. D.; Zoph, B.; Mane, D.; Vasudevan, V.; and Le, Q. V. 2019. AutoAugment: Learning Augmentation Policies from Data. In *CVPR*.

Dabouei, A.; Soleymani, S.; Taherkhani, F.; and Nasrabadi, N. M. 2021a. SuperMix: Supervising the Mixing Data Augmentation. *CVPR*.

Dabouei, A.; Soleymani, S.; Taherkhani, F.; and Nasrabadi, N. M. 2021b. SuperMix: Supervising the Mixing Data Augmentation. *CVPR*.

Faramarzi, M.; Amini, M.; Badrinaaraayanan, A.; Verma, V.; and Chandar, S. 2020. PatchUp: A Regularization Technique for Convolutional Neural Networks. arXiv:2006.07794.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Guo, C.; Pleiss, G.; Sun, Y.; and Weinberger, K. Q. 2017. On Calibration of Modern Neural Networks. arXiv:1706.04599.

Guo, H.; Mao, Y.; and Zhang, R. 2019. MixUp as Locally Linear Out-of-Manifold Regularization. *AAAI*.

Halevy, A.; Norvig, P.; and Pereira, F. 2009. The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*.

Harris, E.; Marcu, A.; Painter, M.; Niranjan, M.; Prügel-Bennett, A.; and Hare, J. 2020. FMix: Enhancing Mixed Sample Data Augmentation. arXiv:2002.12047.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep Residual Learning for Image Recognition. *CVPR*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity Mappings in Deep Residual Networks. In *ECCV*.

Hendrycks, D.; Mu, N.; Cubuk, E. D.; Zoph, B.; Gilmer, J.; and Lakshminarayanan, B. 2020. AugMix: A Simple Method to Improve Robustness and Uncertainty under Data Shift. In *ICLR*.

Hendrycks, D.; Zou, A.; Mazeika, M.; Tang, L.; Li, B.; Song, D.; and Steinhardt, J. 2021. PixMix: Dreamlike Pictures Comprehensively Improve Safety Measures. arXiv:2112.05135.

Ioffe, S.; and Szegedy, C. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. arXiv:1502.03167.

Kim, J.; Choo, W.; Jeong, H.; and Song, H. O. 2021. Co-Mixup: Saliency Guided Joint Mixup with Supermodular Diversity. In *ICLR*.

Kim, J.-H.; Choo, W.; and Song, H. O. 2020. PuzzleMix: Exploiting Saliency and Local Statistics for Optimal Mixup. In *ICML*.

Krizhevsky, A. 2009. Learning multiple layers of features from tiny images. Technical report.

Li, B.; Wu, F.; Lim, S.-N.; Belongie, S.; and Weinberger, K. Q. 2021. On Feature Normalization and Data Augmentation. *CVPR*.

Liu, J.; Liu, B.; Zhou, H.; Liu, Y.; and Li, H. 2022a. TokenMix: Rethinking Image Mixing for Data Augmentation in Vision Transformers. *arXiv:2207.08409*.

Liu, Z.; Li, S.; Wu, D.; Liu, Z.; Chen, Z.; Wu, L.; and Li, S. Z. 2022b. AutoMix: Unveiling the Power of Mixup for Stronger Classifiers. *ECCV*.

Naveed, H. 2021. Survey: Image Mixing and Deleting for Data Augmentation. arXiv:2106.07085.

Park, J.; Yang, J. Y.; Shin, J.; Hwang, S. J.; and Yang, E. 2022. Saliency Grafting: Innocuous Attribution-Guided Mixup with Calibrated Label Mixing. *AAAI*.

Qin, J.; Fang, J.; Zhang, Q.; Liu, W.; Wang, X.; and Wang, X. 2020. ResizeMix: Mixing Data with Preserved Object Information and True Labels. arXiv:2012.11101.

Raffin, A.; Hill, A.; Ernestus, M.; Gleave, A.; Kanervisto, A.; and Dormann, N. 2019. Stable Baselines3. https://github.com/DLR-RM/stable-baselines3.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; Berg, A. C.; and Fei-Fei, L. 2015. ImageNet Large Scale Visual Recognition Challenge. *IJCV*.

Schmidt, R. M.; Schneider, F.; and Hennig, P. 2021. Descending through a Crowded Valley - Benchmarking Deep Learning Optimizers. In *ICML*.

Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; and Klimov, O. 2017. Proximal Policy Optimization Algorithms. arXiv:1707.06347.

Shorten, C.; and Khoshgoftaar, T. M. 2019. A survey on Image Data Augmentation for Deep Learning.

Simonyan, K.; Vedaldi, A.; and Zisserman, A. 2014. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. In *ICLR Workshop*.

Smith, L. N. 2017. Cyclical Learning Rates for Training Neural Networks. arXiv:1506.01186.

Smith, L. N.; and Topin, N. 2018. Super-Convergence: Very Fast Training of Neural Networks Using Large Learning Rates. arXiv:1708.07120.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *JMLR*.

Sun, C.; Shrivastava, A.; Singh, S.; and Gupta, A. 2017. Revisiting Unreasonable Effectiveness of Data in Deep Learning Era. In *ICCV*.

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. arXiv:1312.6199.

Uddin, A. F. M. S.; Monira, M. S.; Shin, W.; Chung, T.; and Bae, S.-H. 2021. SaliencyMix: A Saliency Guided Data Augmentation Strategy for Better Regularization. In *ICLR*.

Venkataramanan, S.; Kijak, E.; Amsaleg, L.; and Avrithis, Y. 2022. AlignMixup: Improving Representations by Interpolating Aligned Features. In *CVPR*.

Verma, V.; Lamb, A.; Beckham, C.; Najafi, A.; Mitliagkas, I.; Lopez-Paz, D.; and Bengio, Y. 2019. Manifold Mixup: Better Representations by Interpolating Hidden States. In *ICML*.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2016. Aggregated Residual Transformations for Deep Neural Networks. In *CVPR*.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features. In *ICCV*.

Zagoruyko, S.; and Komodakis, N. 2017. Wide Residual Networks. arXiv:1605.07146.

Zhang, A.; Lipton, Z. C.; Li, M.; and Smola, A. J. 2021a. Dive into Deep Learning. *arXiv preprint arXiv:2106.11342*.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. In *ICLR*.

Zhang, L.; Deng, Z.; Kawaguchi, K.; Ghorbani, A.; and Zou, J. 2021b. How Does Mixup Help With Robustness and Generalization? In *ICLR*.

Zhao, R.; Ouyang, W.; Li, H.; and Wang, X. 2015. Saliency detection by multi-context deep learning. In *CVPR*.

Zheng, Y.; Zhang, Z.; Yan, S.; and Zhang, M. 2022. Deep AutoAugment. In *ICLR*.

Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2015. Learning Deep Features for Discriminative Localization. arXiv:1512.04150.

Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2016. Learning Deep Features for Discriminative Localization. In *CVPR*.

Zhou, K.; Yang, Y.; Qiao, Y.; and Xiang, T. 2021. Domain Generalization with MixStyle.