AutoMix: Unveiling the Power of Mixup for Stronger Classifiers

Zicheng Liu1⋆ Siyuan Li1⋆, Di Wu1, Zihan Liu1, Zhiyuan Chen1, Lirong Wu1, and Stan Z. Li1⋆⋆

AI Lab, School of Engineering, Westlake University, Hangzhou, Zhejiang, China

Abstract. Data mixing augmentation have proved to be effective for improving the generalization ability of deep neural networks. While early methods mix samples by hand-crafted policies (e.g., linear interpolation), recent methods utilize saliency information to match the mixed samples and labels via complex offline optimization. However, there arises a trade-off between precise mixing policies and optimization complexity. To address this challenge, we propose a novel automatic mixup (AutoMix) framework, where the mixup policy is parameterized and serves the ultimate classification goal directly. Specifically, AutoMix reformulates the mixup classification into two sub-tasks (i.e., mixed sample generation and mixup classification) with corresponding sub-networks and solves them in a bi-level optimization framework. For the generation, a learnable lightweight mixup generator, Mix Block, is designed to generate mixed samples by modeling patch-wise relationships under the direct supervision of the corresponding mixed labels. To prevent the degradation and instability of bi-level optimization, we further introduce a momentum pipeline to train AutoMix in an end-to-end manner. Extensive experiments on nine image benchmarks prove the superiority of AutoMix compared with state-of-the-arts in various classification scenarios and downstream tasks.

Keywords: data augmentation, mixup, image classification

1 Introduction

Recent years have witnessed the great success of Deep Neural Networks (DNNs) in various tasks, such as object recognition [18,40], semantic segmentation [36,3,17], and natural language processing [8]. Most of these successes can be attributed to the use of complex network architectures with numerous parameters and a sufficient amount of data. However, when the data is insufficient, models with high complexity, e.g., Transformer-based networks [9,42], are prone to over-fitting and overconfidence [14], resulting in poor generalization abilities [48,39,1].

To improve the generalization of DNNs, a series of data mixing augmentation techniques emerged. As shown in Figure 1, MixUp [55] generates augmented samples via a linear combination of corresponding data pairs; CutMix [53] desig
Fig. 1. The plot of efficiency vs. accuracy on ImageNet-1k and visualization of mixup methods. AutoMix improves performance without the heavy computational overhead.

a patch replacement strategy that randomly replaces a patch in an image with patches from the other image. However, these hand-crafted methods [45,11,15] cannot guarantee mixed samples containing target objects and might cause the label mismatch problem. Subsequently, [47,43,35] try to guide CutMix by saliency information to relieve this problem. Recently, optimization-based methods try to solve the problem by searching an approximate mixing policy [7,23,22] based on portfolio optimization, e.g., maximizing the saliency regions to confirm the co-presence of the targets in the mixed samples. Although they design more precise mixing policies than head-crafted methods, their indirect optimization and heavy computational overhead limit the algorithms’ efficiency. Evidently, it is not efficient to transform the mixup policy from a random linear interpolation to a complex portfolio optimization problem.

This paper mainly discusses two questions: (1) how to design an accurate mixing policy and serve directly to the mixup classification objective; (2) how to solve generation-classification optimization problems efficiently instead of portfolio optimizations. As a basis for solving these two issues, we first reformulate the mixup training into two sub-tasks, mixed sample generation and mixup classification. Then, we propose a novel automatic mixup framework (AutoMix) that generates accurate mixed samples by a generation sub-network, Mix Block (MB), with a good complexity-accuracy trade-off. Specifically, MB is a cross-attention-based module that dynamically selects discriminative pixels based on feature maps of the sample pair to match the corresponding mixed labels. However, MB may collapse into trivial solutions when optimized jointly with the classification encoder due to a gradient entanglement problem. Thus, Momentum Pipeline (MP) is further introduced to stabilize AutoMix and decouple the training process of this bi-level optimization problem. Comprehensive experiments on eight classification benchmarks (CIFAR-10/100, Tiny-ImageNet, ImageNet-1k, CUB-200, FGVC-Aircraft, iNaturalist2017/2018, and Place205) and eight network architectures show that AutoMix consistently outperforms state-of-the-art mixup methods across different tasks. We further provide an extensive analysis to verify the effectiveness of proposed components and the robustness of hyper-parameters. Our main contributions are three-fold:
– From a fresh perspective, we divide the mixup training into bi-level subtasks: mixed sample generation and mixup classification, and regard the generation as an auxiliary task to the classification. We unify them into a framework named AutoMix to optimize the mixup policy in an end-to-end manner.

– A novel Mix Block is designed for mixed sample generation. The combination of Mix Block and Momentum Pipeline optimizes the two sub-tasks in a decoupled manner and improves mixup training accuracy and stability.

– AutoMix surpasses counterparts significantly on various classification scenarios based on eight popular network architectures and downstream tasks.

2 Preliminaries

Mixup training. We first consider the general image classification task with $k$ different classes: given a finite set of $n$ samples $X = \{x_i\}_{i=1}^{n} \in \mathbb{R}^{n \times W \times H \times C}$ and their ground-truth class labels $Y = \{y_i\}_{i=1}^{n} \in \mathbb{R}^{n \times k}$, encoded by a one-hot vector $y_i \in \mathbb{R}^k$. We seek the mapping from the data $x_i$ to its class label $y_i$ modeled by a deep neural network $f_{\theta}(x) \mapsto y$ with network parameters $\theta$ by optimizing a classification loss $\ell(\cdot)$, say the cross entropy (CE) loss,

$$\ell_{CE}(f_{\theta}(x), y) = -y \log f_{\theta}(x). \quad (1)$$

Then we consider the mixup classification task: given a sample mixup function $h$, a label mixup function $g$, and a mixing ratio $\lambda$ sampled from $Beta(\alpha, \alpha)$ distribution, we can generate the mixup data $X_{mix}$ with $x_{mix} = h(x_i, x_j, \lambda)$ and the mixup label $Y_{mix}$ with $y_{mix} = g(y_i, y_j, \lambda)$. Similarly, we learn $f_{\theta} : x_{mix} \mapsto y_{mix}$ by mixup cross-entropy (MCE) loss,

$$\ell_{MCE} = \lambda \ell_{CE}(f_{\theta}(x_{mix}), y_i) + (1 - \lambda) \ell_{CE}(f_{\theta}(x_{mix}), y_j). \quad (2)$$

Mixup reformulation. Comparing Eq. 1 and Eq. 2, the mixup training has the following features: (1) extra mixup policies, $g$ and $h$, are required to generate $X_{mix}$ and $Y_{mix}$. (2) the classification performance of $f_{\theta}$ depends on the generation policy of mixup. Naturally, we can split the mixup task into two complementary sub-tasks: (i) mixed sample generation and (ii) mixup classification. Notice that the sub-task (i) is subordinate to (ii) because the final goal is to obtain a stronger classifier. Therefore, from this perspective, we regard the mixup generation as an auxiliary task for the classification task. Since $g$ is generally designed as a linear interpolation, i.e., $g(y_i, y_j, \lambda) = \lambda y_i + (1 - \lambda) y_j$, $h$ becomes the key function to determine the performance of the model. Generalizing previous offline methods, we define a parametric mixup policy $h_{\phi}$ as the sub-task with another set of parameters $\phi$. The final goal is to optimize $\ell_{MCE}$ given $\theta$ and $\phi$ as below:

$$\min_{\theta, \phi} \ell_{MCE}(f_{\theta}(h_{\phi}(x_i, x_j, \lambda)), g(y_i, y_j, \lambda)). \quad (3)$$
3 AutoMix

We build a bridge between the mixup generation and classification task with a unified optimization framework named as AutoMix to improve the mixup training efficiency. In this framework, the proposed Mix Block (MB) and Momentum Pipeline (MP) in AutoMix not only can generate semantic mixed samples but reduces computational overhead significantly. A comparison overview with offline approaches is presented in Figure 2.

3.1 Label Mismatch: MixBlock

In Figure 3, we further examined that offline approaches are incapable of addressing the label mismatch issue in mixup training. It is difficult for offline methods to preserve the discriminative features in the mixed sample if detach from the final optimization goal. As a result, the prediction of the accuracy of the mixed sample is limited (see the right of Figure 4). This paper presents a parametric mixup generation function named Mix Block (MB) $\mathcal{M}_\phi$ for learning
AutoMix 5

Fig. 4. Left: AutoMix samples with different $\lambda$ (0, 0.3, 0.7, 1). Right: Top-1 accuracy of mixed data. Prediction is counted as correct if the top-1 prediction belongs to $\{y_i, y_j\}$; Top-2 accuracy is calculated by counting the top-2 predictions are equal to $\{y_i, y_j\}$.

a mixup policy without requiring extensive saliency computation. $\mathcal{M}_\phi$ generates a pixel-wise mixup mask $s \in \mathbb{R}^{H \times W}$ for the pairs of input images, where $s \in [0, 1]$. We regard the mask-based mixup policy as an adaptive selection process in terms of $\lambda$, which can automatically select the discriminative patches from sample pairs to generate label-matched mixed samples. Thus, the core of $\mathcal{M}_\phi$ is the devised $\lambda$ embedded cross-attention mechanism to learn the pixel-level proportional relationships in a given data pair. To do so, the deep feature maps $z$ from $f_\theta$ with rich spatial and semantic information can be utilized to bootstrap the two sub-tasks of mixup. Additionally, to facilitate the capture of task-relevant information in the generated mixed samples, the $\mathcal{M}_\phi$ training is directly supervised by the target loss, $\ell_{MCE}$, in an end-to-end manner.

**Parametric mixup generation.** The generation task can be formulated as a dynamic regression problem: given a sample pair $(x_i, x_j)$ and a mixing ratio $\lambda$, MB predicts the probability that each pixel (or patch) on $x_{mix}$ belongs to $x_i$ according to the feature map pair $(z_i^l, z_j^l)$ and mixing ratio $\lambda$. The overall parametric mixup function of AutoMix can be formulated as follows:

$$h_\phi(x_i, x_j, \lambda) = \mathcal{M}_\phi(z_i^l, \lambda, z_j^{l, 1-\lambda}) \odot x_i + (1 - \mathcal{M}_\phi(z_i^l, \lambda, z_j^{l, 1-\lambda})) \odot x_j,$$

where $\odot$ denotes element-wise product; $z_i^l$ is $\lambda$ embedded feature map at $l$-th layer. As shown in the right of Figure 5, we first embed $\lambda$ with the $l$-th feature map in a simple and efficient way by concatenating, $z_i^l_\lambda = \text{concat}(z, \lambda)$, whose effectiveness has been shown in the left of Figure 4. As we can see from Equation 4, our aim is to obtain a pixel-level mask $s$ in the input space from $\mathcal{M}_\phi(\cdot)$ based on $\lambda$ embedded $z_i^l_\lambda$ and $z_j^{l, 1-\lambda}$ to generate semantic mixed samples. In order to achieve this goal, a pair-wise similarity matrix $P$ and an upsampling function $U(\cdot)$ is required. Due to the symmetry of mixup, i.e., the sum of the two masks used to generate a mixed sample is equal to 1, for $x_i$ of a pair $(x_i, x_j)$, we can denote $\mathcal{M}_\phi : z_i^l_\lambda, z_j^{l, 1-\lambda} \rightarrow s_i$,

$$s_i = U(\sigma(P(z_i^l_\lambda, z_j^{l, 1-\lambda}) \odot W_Z z_i^l_\lambda)),$$

where $W_Z$ is a linear transformation matrix; $\sigma$ is the Sigmoid activation function, which is used to probabilize the mask; and $s_i$ is the $H \times W$ mask we are looking for. By multiplying $P$ and the value embedding, $W_z z_i^l_\lambda$, the discriminative features
Fig. 5. The left diagram represents the five key steps of AutoMix. (1) Extract feature map \( Z \) from the frozen encoder \( k \). (2) Mix Block \( M_\phi \) generates mixed samples by using \( Z \) and mixup ratio \( \lambda \in [0, 1] \). (3) and (4) Decoupled training \( M_\phi \) and encoder \( q \) via stop gradient, the blue and green lines indicate the encoder training and the \( M_\phi \) training, correspondingly. (5) Update the \( k \)'s parameters through momentum moving. The right diagram is the architecture of proposed \( M_\phi \).
of $\ell_{MCE}$, $\textsc{cls}$ denotes classification task for training encoder and $\textsc{gen}$ denotes generation task for training $\mathcal{M}_\phi$. AutoMix can be optimized by a joint loss:

$$L(\theta, \phi) = \underbrace{\ell_{CE}}_{\text{classification}} + \underbrace{\ell_{MCE}^{\text{cls}} + \ell_{MCE}^{\text{gen}}} + \underbrace{\ell_\lambda}_{\text{classification}} + \underbrace{\ell_{MCE}^{\text{gen}}}_{\text{generation}}.$$ (8)

Obviously, the purpose of the classification task is to optimize $\theta$ while the generation task is to optimize $\phi$. Therefore, this is a typical bi-level optimization problem. Although $\mathcal{M}_\phi$ does not need extra computational overhead to maximize the saliency information, using SGD to directly update the nested $\theta$ and $\phi$ will lead to instability. To address this problem properly, we use the momentum pipeline to decouple the training of $\theta$ and $\phi$. As indicated in Eq. 8, though the same $\ell_{MCE}$ is used, the focus of each is different.

### 3.2 Bi-level Optimization: Momentum Pipeline

Although MB is designed to be lightweight and efficient, it also poses a bi-level optimization problem with *gradient entanglement*. Experiments demonstrate that the entanglement problem may cause $\mathcal{M}_\phi$ trapped into a trivial solution (degraded to MixUp, in Figure 6). $\mathcal{M}_\phi$ with a much smaller parameters than the encoder will be disturbed by the classification task when optimizing both the two sub-tasks at the same time. MB thus cannot generate semantic mixed samples stably and eventually collapse. According to gradient entanglement problem of $\ell_{MCE}^{\text{cls}}$

$$\nabla_\phi L_{MCE}^{\text{cls}} \propto \nabla_\phi h_\phi(x_i, x_j, \lambda) \odot f_\theta'(h_\phi(x_i, x_j, \lambda)).$$ (9)

It is notable that the instability of $f_\theta$ may result in a vicious cycle of the joint training. As a consequence, the primary goal of getting the Eq. 3 operating well is to ensure that $f_\theta$ outputs stable features and, to the extent possible, that $\phi$ and $\theta$ can focus on their own tasks in the case of using the same loss. Inspired by methods in self-supervised learning [16,13], they adopted momentum pipeline (MP) to avoid the feature collapse and realized that the teacher network $f_\theta_k$ of the Siamese network shows more stable performance than student network $f_\theta_q$. Along this path, we designed a new MP for decoupling the nested bi-level optimization problem of AutoMix: the student network $f_\theta_q$ focuses on the classification task, while the stable teacher network $f_\theta_k$ is connected with $\mathcal{M}_\phi$ to perform generation task. Moreover, optimizing Eq. 8 with batch approach requires $X_{mix}$ generated by $f_\theta_k$ and $\mathcal{M}_\phi$ first and then using $X_{mix}$ to optimize $f_\theta_q$. By analogy, referring to the Expectation-Maximization (EM) algorithm, the two sets of parameters $\theta$
Fig. 7. Visualization of mixed samples generated by $\mathcal{M}_\phi$ with $\lambda = 0.5$ at different training periods on ImageNet-1k (100 epochs in total). It is worth noting that $\mathcal{M}_\phi$ is able to generate mixed samples stably and converge quickly with the addition of MP.

and $\phi$ can be optimized in an alternating way by the designed MP, i.e., first fix one set of parameters optimizing the other:

$$
\theta_q^t \leftarrow \arg\min_{\theta} \mathcal{L}(\theta_{q}^{t-1}, \phi^{t-1}),
$$

(10)

$$
\phi^t \leftarrow \arg\min_{\phi} \mathcal{L}(\theta_{k}^t, \phi^{t-1}),
$$

(11)

where $t$ is the iteration step, $\theta_q$ and $\theta_k$ represent the parameters of student and teacher network, respectively. Note that $f_{\theta_q}$ and $f_{\theta_k}$ share the same network structure with the same initialized parameters, but $f_{\theta_k}$ is updated via an exponential moving average (EMA) strategy [34] from $f_{\theta_q}$:

$$
\theta_k^t \leftarrow m\theta_k + (1 - m)\theta_q,
$$

(12)

where $m \in [0, 1)$ is the momentum coefficient. It is worthy to notice that MP not only solves optimization instability but also significantly speeds up and stabilizes the convergence of AutoMix. In Figure 7, $\mathcal{M}_\phi$ gets close to convergence in the first few epochs and consistently delivers high-quality mixed samples to $f_\theta$. Moreover, detailed AutoMix architecture and pseudo code are provided in Appendix.

4 Experiments

We evaluate AutoMix in three aspects: (1) Image classification in various scenarios based on various network architectures, (2) Robustness against corruptions and adversarial samples, and (3) Transfer learning capacities to downstream tasks.

4.1 Evaluation on Image Classification

This subsection demonstrates performance gains of AutoMix for various classification tasks on eight classification benchmarks, including CIFAR-10/100 [24], Tiny-ImageNet [5], ImageNet-1k [37], CUB-200-2011 (CUB) [46], FGVC-Aircraft (Aircraft) [32], iNaturalist2017/2018 (iNat2017/2018) [21], and Place205 [56]. We verify generalizabilities of AutoMix for eight network architectures, the experiments adopt popular ConvNets, including ResNet (R) [18], Wide-ResNet
AutoMix 9

(WRN) [54], ResNeXt (32x4d) (RX) [51], MobileNet.V2 [38], EfficientNet [40], and ConvNeXt [29], and Transformer-based architectures (DeiT [42] and Swin Transformer (Swin) [28]) as backbone networks. For a fair comparison, we use open-source codebase OpenMixup [26] for most mixup methods: (i) hand-crafted methods: Mixup [55], CutMix [53], ManifoldMix [45], AugMix [20], AttentiveMix [47], SaliencyMix [43], FMix [15], and ResizeMix [35]; (ii) optimization-based methods: PuzzleMix [23], Co-Mixup [22], and SuperMix [7]. Notice that AugMix is reproduced by timm [49], * denotes open-source arxiv preprint work, and methods without source codes (AlignMix [44] and TransMix [2]) are not compared. All mixup methods use the optimal $\alpha$ among $\{0.2, 0.5, 1, 2, 4\}$, while the rest of hyper-parameters follow the original paper. AutoMix uses the same set of hyper-parameters in all experiments: $\alpha = 2$, the feature layer $l = 3$, the momentum coefficient in MP starts from $m = 0.999$ and is increased to 1 in a cosine curve. As for all classification results, we report the mean performance of 3 trials where the median of top-1 test accuracy in the last 10 training epochs is recorded for each trial, and bold and blue denote the best and second best results.

Table 1. Top-1 accuracy (%)↑ of various algorithms based on ResNet variants for small-scale classification on CIFAR-10/100 and Tiny-ImageNet datasets.

| Method       | CIFAR-10 | CIFAR-100 | Tiny-ImageNet |
|--------------|----------|-----------|---------------|
|              | R-18 RX-50 | R-18 RX-50 WRN-28-8 | R-18 RX-50 |
| Vanilla      | 95.50 96.23 | 78.04 81.09 81.63 | 61.68 65.04 |
| MixUp        | 96.62 97.30 | 79.12 82.10 82.82 | 63.86 66.36 |
| CutMix       | 96.68 97.01 | 78.17 81.67 84.45 | 65.53 66.47 |
| ManifoldMix  | 96.71 97.33 | 80.35 82.88 83.24 | 64.15 67.30 |
| SaliencyMix  | 96.53 97.18 | 79.12 81.53 84.35 | 64.60 66.55 |
| FMix*        | 96.58 96.76 | 79.69 81.90 84.21 | 63.47 65.08 |
| PuzzleMix    | 97.10 97.27 | 81.13 82.85 85.02 | 65.81 66.92 |
| Co-Mixup     | 97.15 97.32 | 81.17 82.91 85.05 | 65.92 67.13 |
| ResizeMix*   | 96.76 97.21 | 80.01 81.82 84.87 | 63.74 65.87 |
| **AutoMix**  | 97.40 98.04 | 82.04 83.64 85.18 | 67.33 70.72 |
| **Gain**     | +0.25 +0.71 | +0.87 +0.76 +0.13 | +1.41 +3.42 |

Small-scale Datasets

**Settings.** On CIFAR-10/100, RandomFlip and RandomCrop with 4 pixels padding for 32×32 resolutions are basic data augmentations, and we use the following training settings: SGD optimizer with SGD weight decay of 0.0001, the momentum of 0.9, the batch size of 100, and training 800 epochs; the basic learning rate is 0.1 adjusted by Cosine Scheduler [30]. On Tiny-ImageNet, the basic augmentations include RandomFlip and RandomResizedCrop for 64×64 resolutions, and we use the similar training ingredients as CIFAR except for the basic learning rate of 0.2 and training 400 epochs. CIFAR version of ResNet variants are used, i.e., replacing the 7 × 7 convolution and MaxPooling by a 3 × 3 convolution.

**Classification.** Table 1 shows small-scale classification results on CIFAR-10/100 and Tiny datasets. Compared to the previous state-of-the-art methods, AutoMix
consistently surpasses ManifoldMix (+0.71∼2.21%), PuzzleMix (+0.16∼0.91%), and Co-Mixup (+0.13∼0.87%) based on various ResNet architectures on CIFAR-10/100. Moreover, AutoMix noticeably outperforms previous best algorithms by 1.41% and 3.42% on more challenging Tiny-ImageNet.

**Calibration.** DNNs tend to predict over-confidently in classification tasks [41], mixup methods can significantly alleviate this problem. To verify the calibration ability of AutoMix, we evaluate popular mixup algorithms by the expected calibration error (ECE) [14] on CIFAR-100, i.e., the absolute discrepancy between accuracy and confidence. As shown in Figure 8, AutoMix has the best calibration effect among all competitors with the ECE error rate of 2.3%, closest to the red diagonal. We can see from the figure that the Cut series perform not well on calibration, but may further aggravate the overconfidence; while MixUp and ManifoldMix calibrate the predictions, but cause the under-confidence problem.

### Table 2. Top-1 accuracy (%)† of image classification based on ResNet variants on ImageNet-1k using PyTorch-style 100-epoch and 300-epoch training procedures.

| Methods          | PyTorch 100 epochs | PyTorch 300 epochs |
|------------------|-------------------|-------------------|
|                  | R-18        | R-34        | R-50        | R-101       | RX-101      | R-18        | R-34        | R-50        | R-101       |
| Vanilla          | 70.04       | 73.85       | 76.83       | 78.18       | 78.71       | 71.83       | 75.29       | 77.35       | 78.91       |
| MixUp            | 69.98       | 73.97       | 77.12       | 78.97       | 79.98       | 71.72       | 75.73       | 78.44       | 80.60       |
| CutMix           | 68.95       | 73.58       | 77.17       | 78.96       | 80.42       | 71.01       | 75.16       | 76.69       | 80.59       |
| ManifoldMix      | 69.98       | 73.98       | 77.01       | 79.02       | 79.93       | 71.73       | 75.44       | 78.21       | 80.64       |
| SaliencyMix      | 69.16       | 73.56       | 77.14       | 79.32       | 80.27       | 70.21       | 75.01       | 78.46       | 80.45       |
| FMix*            | 69.96       | 73.08       | 77.19       | 79.09       | 80.06       | 70.30       | 75.12       | 78.51       | 80.20       |
| PuzzleMix        | 70.12       | 74.26       | 77.54       | 79.43       | 80.53       | 71.64       | 75.84       | 78.86       | 80.67       |
| ResizeMix*       | 69.50       | 73.88       | 77.42       | 79.27       | 80.55       | 71.32       | 75.64       | 78.91       | 80.52       |
| **AutoMix**      | **70.50**   | **74.52**   | **77.91**   | **79.87**   | **80.89**   | **72.05**   | **76.10**   | **79.25**   | **80.98**   |
| Gain             | +0.38       | +0.26       | +0.37       | +0.44       | +0.34       | +0.22       | +0.26       | +0.34       | +0.31       |

### ImageNet Datasets

**Settings.** In the more challenging large-scale classification scenarios, mixup methods are widely used, especially for recently proposed Transformer-based networks. We evaluate AutoMix and popular mixup variants on ImageNet-1k using three popular training procedures: (a) PyTorch-style setting trains 100 or 300 epochs by SGD optimizer with the batch size of 256, the basic learning rate of 0.1, the SGD weight decay of 0.0001, and the SGD momentum of 0.9, which is the standard benchmarks for mixup methods [53,35]; (b) DeiT setting

![Fig. 8. Calibration plots of Mixup variants and AutoMix on CIFAR-100 using ResNet-18. The red line indicates the expected prediction tendency.](image-url)
Table 3. Top-1 accuracy (%)↑ on ImageNet-1k based on various ConvNets using RSB A2/A3 training settings.

| Methods    | R-50 A3 | EfficientNet B0 A2 | MobileNet.V2 A3 | Gain  |
|------------|---------|--------------------|-----------------|-------|
| RSB        | 78.98   | 77.26              | 74.02           | +0.36 |
| MixUp      | 77.66   | 77.19              | 73.87           | +0.23 |
| CutMix     | 77.62   | 77.24              | 73.46           | +0.51 |
| ManifoldMix| 77.78   | 77.22              | 73.83           | +0.51 |
| SaliencyMix| 77.93   | 77.06              | 73.42           | +0.23 |
| FMix*      | 77.76   | 77.33              | 73.71           | +0.23 |
| PuzzleMix  | 78.02   | 77.35              | 74.10           | +0.31 |
| ResizeMix* | 77.85   | 77.27              | 73.67           | +0.31 |
| AutoMix    | 78.44   | 77.58              | 74.61           | +0.32 |

Table 4. Top-1 accuracy (%)↑ on ImageNet-1k based on ViTs and ConvNeXt using DeiT training settings.

| Methods     | DeiT-S Swin-T | ConvNeXt-T |
|-------------|---------------|-------------|
| DeiT        | 79.80         | 81.28       |
| MixUp       | 79.65         | 80.71       |
| CutMix      | 79.78         | 80.83       |
| AttentiveMix| 77.63         | 77.27       |
| SaliencyMix | 79.32         | 80.68       |
| FMix*       | 79.41         | 80.37       |
| PuzzleMix   | 79.84         | 81.03       |
| ResizeMix* | 79.93         | 80.94       |
| AutoMix     | 80.78         | 81.80       |

trains 300 epochs by AdamW optimizer [31] with the batch size of 1024, the basic learning rate of 0.001, and the weight decay of 0.05; (c) timm [49] RSB A2/A3 settings train 300/100 epochs by LAMB optimizer [52] with the batch size of 2048, the basic learning rate of 0.005/0.008, and the weight decay of 0.02. More detailed ingredients and hyper-parameters are provided in Appendix. These three settings adopt the basic data augmentations (RandomResizedCrop and RandomFlip) for 224 × 224 resolutions with Cosine Scheduler by default, (b) and (c) use RandAugment [6] for better performances.

Classification. Table 2 and Figure 1 show regular image classification results using only one mixup methods: AutoMix consistently outperforms previous state-of-the-art methods with light/median/heavy ResNet architectures, e.g., +0.26∼0.44% for 100 epochs and +0.22∼0.34% for 300 epochs. Table 3 and Table 4 report results on more practical training settings: RSB and DeiT denote randomly combining Mixup and CutMix which produces competitive performs as previous state-of-the-art methods (e.g., PuzzleMix), while AutoMix still brings significantly gains over the original RSB (+0.32∼1.30%) and DeiT (+0.18∼0.98%). It worth notice that previous mixup variants yield little performance gain when training with light ConvNets (e.g., R-18 in Table 2, EfficientNet B0 and MobileNet.V2 (1×) in Table 3), while AutoMix achieving stable performance gains on these backbones. Moreover, AutoMix achieves competitive performances than the recently proposed Transformer-based mixup method, TransMix.

4.2 Evaluation on Fine-grained and Scenic Classification

Small-scale datasets. We first perform small-scale fine-grained classification following transfer learning settings on CUB-200 and Aircraft: training 200 epochs by SGD optimizer with the initial learning rate of 0.001, the weight decay of 0.0005, the batch size of 16, using the standard augmentations as in Sec. 4.1; the official PyTorch pre-trained models on ImageNet-1k are adopted as initialization. Table 5 shows that AutoMix achieves the best performance and noticeably improves the vanilla (2.19%/3.55% on CUB-200 and 1.14%/1.52% on Aircraft), which verifies that AutoMix has strong adaptability to more challenging scenarios.
Table 5. Top-1 accuracy (%)↑ of various algorithms based on ResNet variants on fine-grained and scenic classification datasets.

| Method       | CUB-200 R-18 RX-50 | FGVC-Aircraft R-18 RX-50 | iNat2017 R-50 RX-101 | iNat2018 R-50 RX-101 | Place205 R-18 RX-50 |
|--------------|---------------------|-------------------------|----------------------|----------------------|---------------------|
| Vanilla      | 77.68 83.01 80.23 85.10 | 60.23 63.70 62.53 66.94 | 59.63 63.10 |
| MixUp        | 78.39 84.58 79.52 85.18 | 61.22 66.27 62.69 67.56 | 59.33 63.01 |
| CutMix       | 78.40 85.68 78.84 84.55 | 62.34 67.59 63.91 69.75 | 59.21 63.75 |
| ManifoldMix  | 79.76 86.38 80.68 86.60 | 61.47 66.08 63.46 69.30 | 59.46 63.23 |
| SaliencyMix  | 77.95 83.29 80.02 84.31 | 62.51 67.20 64.27 70.01 | 59.50 63.33 |
| FMix*        | 77.28 84.06 79.36 86.23 | 61.90 66.64 63.71 69.46 | 59.51 63.63 |
| PuzzleMix    | 78.63 84.51 80.76 86.23 | 62.66 67.72 64.36 70.12 | 59.62 63.91 |
| ResizeMix*   | 78.50 84.77 78.10 84.08 | 62.29 66.82 64.12 69.30 | 59.66 63.88 |
| AutoMix      | 79.87 86.56 81.37 86.72 | 63.08 68.03 64.73 70.49 | 59.74 64.06 |
| Gain         | +0.11 +0.18 +0.61 +0.12 +0.42 +0.31 +0.37 +0.37 +0.08 +0.15 |

Table 6. Top-1 accuracy (%)↑ and FGSM error (%)↓ on CIFAR-100 based on ResNeXt-50 (32x4d) trained 400 epochs.

| Clean Acc(%)↑ | Corruption Acc(%)↑ | FGSM Error(%)↓ | Clean Acc(%)↑ | Corruption Acc(%)↑ | FGSM Error(%)↓ | Clean Acc(%)↑ | Corruption Acc(%)↑ | FGSM Error(%)↓ |
|--------------|-------------------|----------------|--------------|-------------------|----------------|--------------|-------------------|----------------|
| Vanilla      | 80.24 51.71 63.92 |                |              |                   |                |              |                   |                |
| MixUp        | 82.44 58.10 56.60 |                |              |                   |                |              |                   |                |
| CutMix       | 81.09 49.32 76.84 |                |              |                   |                |              |                   |                |
| AugMix       | 81.18 66.54 55.59 |                |              |                   |                |              |                   |                |
| PuzzleMix    | 82.76 57.82 63.71 |                |              |                   |                |              |                   |                |
| AutoMix      | 83.13 58.35 55.34 |                |              |                   |                |              |                   |                |

Large-scale datasets. Then, we adopt similar settings as (a) in Sec. 4.1 with the total epoch of 100 epochs (training from scratch) on large-scale datasets based on ResNet variants. As for the imbalanced and long-tail fine-grained recognition tasks on iNat2017/2018, Table 5 shows that AutoMix surpasses the previous best methods and improves the vanilla by large margins (2.74%/4.33% on iNat2017 and 2.20%/3.55% on iNat2018), which demonstrates that AutoMix can alleviate the long-tail and imbalance issues. As for scenic classification on Place205, AutoMix still sets state-of-the-art performances. Therefore, we can conclude that AutoMix can adapt to more challenging scenarios.

4.3 Robustness

We first evaluate robustness against corruptions on CIFAR-100-C [19], which is designed for evaluating the corruption robustness and provides 19 different corruptions (e.g., noise, blur, and digital corruption, etc). AugMix [20] is proposed to improve robustness against natural corruptions by minimizing Jensen-Shannon divergence (JSD) between logits of a clean image and two AugMix images. However, the improvement of AugMix is very limited on clean data. In Table 6, AutoMix shows a consistent top level in both clean and corruption data. We further study robustness against the FGSM [12] white box attack of $8/255 \ell_\infty$ epsilon ball following [55], and AutoMix outperforms previous methods in Table 6.
Table 9. Ablation of modules in MixBlock.

| module                        | Tiny-ImageNet | CIFAR-100 | ImageNet-1k |
|-------------------------------|--------------|-----------|-------------|
| (random grids)                | 64.40 66.83  | 79.12     | 69.98       |
| + cross attention             | 66.87 69.76  | 80.82     | 66.02       |
| +λ embedding                  | 67.15 70.41  | 80.41     | 66.10       |
| +ℓλ                           | **67.33 70.72** | **79.64 82.04** | **66.10 65.05** |

Table 10. Ablation of the proposed momentum pipeline (MP) and the cross-entropy loss \( l_{CE} \) (CE) based on ResNet-18.

| module                        | Tiny-ImageNet | CIFAR-100 | ImageNet-1k |
|-------------------------------|--------------|-----------|-------------|
| (none)                        | 79.12 78.17  | 63.39     | 69.98       |
| +MP (m=0)                     | - - 81.75   | - - 67.05 | - - 70.41   |
| +MP                           | **80.82 79.57 81.33** | **66.02 65.72 67.19** | **70.13 70.02 70.45** |
| +MP + CE                      | 80.41 **79.64 82.04** | 66.10 65.05 67.33 | 70.10 70.04 70.50 |

4.4 Transfer Learning

**Weakly supervised object localization.** Following CutMix, we also evaluate AutoMix on the weakly supervised object localization (WSOL) task on CUB-200, which aims to localize objects of interest without bounding boxes supervision. We use CAM to extract attention maps, and calculate the maximal box accuracy with a threshold \( \delta \in \{0.3, 0.5, 0.7\} \), following MaxBoxAccV2 [4]. Table 8 shows that AutoMix achieves the best performance to localize semantic regions.

**Object detection.** We then evaluate transferable abilities of the learned features to object detection task with Faster R-CNN [36] on PASCAL VOC train-val07+12 [10] and COCO train2017 [27] based on Detectron2 [50]. We fine-tune Faster R-CNN with R50-C4 pre-trained on ImageNet-1k with mixup methods on VOC (24k iterations) and COCO (2× schedule). Table 7 shows that AutoMix achieves better performances than previous cutting-based mixup variants.

Table 8. MaxBoxAcc (%)↑ for the WSOL task on CUB-200 based on ResNet variants.

| Backbone | Vanilla | Mixup | FMix* | PuzzleMix | Co-Mixup | OurS |
|----------|---------|-------|-------|-----------|----------|------|
| R-18     | 49.91   | 48.62 | 51.85 | 50.30     | 53.95    | **54.13** |
| RX-50    | 53.38   | 50.27 | 57.16 | 59.80     | 59.34    | 59.76 | **61.05** |

4.5 Ablation Study

We conduct an ablation study to prove that each component of AutoMix plays an essential role to make the framework operate properly. Three main questions are answered here: (1) Are the modules in MB effective? (2) How many gains can MB bring without EMA and CE? (3) Is AutoMix robust to hyperparameters?

(1) Cross-attention mechanism enables MB to capture the task-relevant pixels between two samples, which is the core design of MB to generate useful mixed masks. Based on this, \( \lambda \) embedding and \( \ell_\lambda \) encourage MB to learn proportional correspondence on a different scale. Without these modules, the performance drops by almost 4% (66.83% vs. 70.72%), as shown in Figure 9.

(2) In Table 10, we show that the EMA and CE adopted in the MP improves performance of MB by ensuring training stability, however CE is not as effective for other mixup methods. Most importantly, without these them, i.e. EMA and CE, we show MB still delivers significant gains (e.g. +2.29% and +2.21% on CIFAR-100 and Tiny). Note that \( m = 0 \) indicates removing
Fig. 9. Ablation of hyperparameter $\alpha$ of AutoMix on CIFAR-100 and Tiny-ImageNet.

Table 11. Ablation of feature layer $l$ on Tiny-ImageNet, reporting top-1 Acc (%)↑ vs. params (M)↓ vs. the total training time (hours)↓.

| M | R-18 | RX-50 |
|---|------|-------|
| Mixup | Acc(%) | Params | Time |
| $l_4$ | 67.32 | 11.64 | 28 | 70.67 | 27.99 | 174 |
| $l_3$ | 67.33 | 11.44 | 34 | 70.72 | 24.84 | 196 |
| $l_2$ | 67.27 | 11.39 | 41 | 70.43 | 23.86 | 252 |
| $l_1$ | 67.30 | 11.38 | 67 | 70.70 | 22.80 | 413 |
| $l_0$ | 63.86 | 11.27 | 20 | 66.36 | 23.38 | 113 |

EMA, which means $f_{\theta_k}$ is a copy of $f_{\theta_q}$ with the same weights. Therefore, we can confirm the effectiveness of $M_{\phi}$.

(3) AutoMix has two core hyper-parameters, $\alpha$ and $l$, which are fixed for all experiments. A larger $\alpha$ facilitates MB to learn intra-class relationships. Figure 9 shows that AutoMix with $\alpha = 2$ as default achieves the best performances on various datasets. The feature layer $l_3$ makes a good trade-off between the performance and complexity, as shown in Table 11.

5 Related Work

MixUp [55], the first mixing-based data augmentation algorithm, was proposed to generate mixed samples with mixed labels by convex interpolations of any two samples and their unique one-hot labels. ManifoldMix [45] extends MixUp to the hidden space of DNNs and [11,44] improves ManifoldMix. CutMix [53] incorporates the Dropout strategy into the mixup strategy and proposes a mixing strategy based on the patch of the image, i.e., randomly replacing a local rectangular area in images. Based on CutMix, AttentiveMix [47] and SaliencyMix [43] guide mixing patches by saliency regions in the image (based on CAM or a saliency detector) to obtain mixed samples with more class-relevant information; ResizeMix [35] maintains the information integrity by replacing one resized image directly into a rectangular area of another image. Furthermore, PuzzleMix [23] and Co-Mixup [22] propose combinatorial optimization strategies to find optimal mixup masks by maximizing the saliency information. FMix [15] transforms images into the spectrum domain to generate binary masks by setting a threshold. Compared with previous methods, AutoMix does not require a hand-crafted sample mixing strategy or saliency information but adaptively generates mixed samples based on mixing ratios and feature maps in end-to-end manner.

6 Conclusion and Limitations

In this paper, we propose an AutoMix framework, which optimizes both the mixed sample generation task and the mixup classification task in a momentum training pipeline. Without adding cost to inference, AutoMix can generate out-of-manifold samples with adaptive masks. Extensive experiments have shown the effectiveness and excellent generalizability of the proposed AutoMix on CIFAR, ImageNet, and fine-grained datasets. On top of that, we also outperformed other
mixup algorithms when comparing with robustness and localization tasks as well. Furthermore, the proposed momentum training pipeline serves a significant improvement in convergence speed and overall performance.

However, AutoMix is now learning the mixed policy on only two samples, and it would be more efficient if it could be extended to multiple samples. Although the time complexity of AutoMix is much faster than that of the combinatorial optimization-based method, it is still a big gap with the hand-crafted method. A trained Mix Block will be a promising avenue in future research.
References

1. Bishop, C.M.: Pattern recognition and machine learning. springer (2006) 1
2. Chen, J.N., Sun, S., He, J., Torr, P., Yuille, A., Bai, S.: Transmix: Attend to mix for vision transformers (2021) 9
3. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE transactions on pattern analysis and machine intelligence 40(4), 834–848 (2017) 1
4. Choe, J., Oh, S.J., Lee, S., Chun, S., Akata, Z., Shim, H.: Evaluating weakly supervised object localization methods right. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3133–3142 (2020) 13
5. Chrabaszcz, P., Loshchilov, I., Hutter, F.: A downsampled variant of imagenet as an alternative to the cifar datasets. arXiv preprint arXiv:1707.08819 (2017) 8, 20
6. Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. pp. 702–703 (2020) 11
7. Dabouei, A., Soleymani, S., Taherkhani, F., Nasrabadi, N.M.: Supermix: Supervising the mixing data augmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 13794–13803 (2021) 2, 9, 20
8. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018) 1
9. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., Houlsby, N.: An image is worth 16x16 words: Transformers for image recognition at scale. In: International Conference on Learning Representations (ICLR) (2021) 1
10. Everingham, M., Van Gool, L., Williams, C.K., Winn, J., Zisserman, A.: The pascal visual object classes (voc) challenge. International journal of computer vision 88(2), 303–338 (2010) 13
11. Faramarzi, M., Amini, M., Badrinaaraayanan, A., Verma, V., Chandar, S.: Patchup: A regularization technique for convolutional neural networks. arXiv preprint arXiv:2006.07794 (2020) 2, 14
12. Goodfellow, I.J., Shlens, J., Szegedy, C.: Explaining and harnessing adversarial examples. In: International Conference on Learning Representations (ICLR) (2015) 12
13. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P.H., Buchatskaya, E., Doersch, C., Pires, B.A., Guo, Z.D., Azar, M.G., et al.: Bootstrap your own latent: A new approach to self-supervised learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2020) 7
14. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On calibration of modern neural networks. In: International Conference on Machine Learning. pp. 1321–1330. PMLR (2017) 1, 10
15. Harris, E., Marcu, A., Painter, M., Niranjjan, M., Hare, A.P.B.J.: Fmix: Enhancing mixed sample data augmentation. arXiv preprint arXiv:2002.12047 2(3), 4 (2020) 2, 9, 14, 20
16. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9729–9738 (2020) 7
17. He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: Proceedings of the International Conference on Computer Vision (ICCV) (2017) 1
18. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR). pp. 777–787 (2016) 1, 8, 20
19. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261 (2019) 12
20. Hendrycks, D., Mu, N., Cubuk, E.D., Zoph, B., Gilmer, J., Lakshminarayanan, B.: Augmix: A simple data processing method to improve robustness and uncertainty. arXiv preprint arXiv:1912.02781 (2019) 9, 12, 20
21. Horn, G.V., Aodha, O.M., Song, Y., Cui, Y., Sun, C., Shepard, A., Adam, H., Perona, P., Belongie, S.: The inaturalist species classification and detection dataset. In: Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 8, 20
22. Kim, J.H., Choo, W., Jeong, H., Song, H.O.: Co-mixup: Saliency guided joint mixup with supermodular diversity. arXiv preprint arXiv:2102.03065 (2021) 2, 9, 14, 20
23. Kim, J.H., Choo, W., Song, H.O.: Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In: International Conference on Machine Learning. pp. 5275–5285. PMLR (2020) 2, 9, 14, 20
24. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 8, 20
25. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012) 20
26. Li, S., Liu, Z., Wu, D.: OpenMixup: Open mixup toolbox and benchmark for visual representation. https://github.com/Westlake-AI/openmixup (2022) 9
27. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: Proceedings of the European Conference on Computer Vision (ECCV) (2014) 13
28. Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: International Conference on Computer Vision (ICCV) (2021) 9
29. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s (2022) 9
30. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 9, 20
31. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2019) 11
32. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013) 8, 20
33. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, 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 (NeurIPS) (2019) 20
34. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. SIAM Journal on Control and Optimization pp. 838–855 (1992) 8
35. Qin, J., Fang, J., Zhang, Q., Liu, W., Wang, X., Wang, X.: Resizemix: Mixing data with preserved object information and true labels. arXiv preprint arXiv:2012.11101 (2020) 2, 9, 10, 14, 20
36. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object
detection with region proposal networks. arXiv preprint arXiv:1506.01497 (2015) 1, 13
37. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpa-
thy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition
challenge. International journal of computer vision pp. 211–252 (2015) 8
38. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: In-
verted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on
Computer Vision and Pattern Recognition (CVPR) (2018) 9
39. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout:
a simple way to prevent neural networks from overfitting. The journal of machine
learning research 15(1), 1929–1958 (2014) 1
40. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolulational neural
networks. In: International Conference on Machine Learning (ICML) (2019) 1, 9
41. Thulasidasan, S., Chennupati, G., Bilmes, J., Bhattacharya, T., Michalak, S.: On
mixup training: Improved calibration and predictive uncertainty for deep neural
networks. arXiv preprint arXiv:1905.11001 (2019) 10
42. Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jegou, H.: Training
data-efficient image transformers & distillation through attention. In: International
Conference on Machine Learning (ICML). pp. 10347–10357 (2021) 1, 9, 20
43. Uddin, A., Monira, M., Shin, W., Chung, T., Bae, S.H., et al.: Saliencymix: A
saliency guided data augmentation strategy for better regularization. arXiv preprint
arXiv:2006.01791 (2020) 2, 9, 14, 20
44. Venkataramanan, S., Avrithis, Y., Kijak, E., Amsaleg, L.: Alignmix: Improving
representation by interpolating aligned features (2021) 9, 14
45. Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Lopez-Paz, D.,
Bengio, Y.: Manifold mixup: Better representations by interpolating hidden states.
In: International Conference on Machine Learning. pp. 6438–6447 (2019) 2, 9, 14,
20
46. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd
birds-200-2011 dataset. California Institute of Technology (2011) 8, 20
47. Walawalkar, D., Shen, Z., Liu, Z., Savvides, M.: Attentive cutmix: An enhanced data
augmentation approach for deep learning based image classification. In: ICASSP
2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Pro-
cessing (ICASSP). pp. 3642–3646 (2020) 2, 9, 14
48. Wan, L., Zeiler, M., Zhang, S., Le Cun, Y., Fergus, R.: Regularization of neural
networks using dropconnect. In: International conference on machine learning. pp.
1058–1066. PMLR (2013) 1
49. Wightman, R., Touvron, H., Jégou, H.: Resnet strikes back: An improved training
procedure in timm (2021) 9, 11, 20
50. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://
github.com/facebookresearch/detectron2 (2019) 13
51. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transforma-
tions for deep neural networks. In: Proceedings of the IEEE conference on computer
vision and pattern recognition. pp. 1492–1500 (2017) 9
52. 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
(ICLR) (2020) 11
53. Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: Proceedings of the International Conference on Computer Vision (ICCV). pp. 6023–6032 (2019) 1, 9, 10, 14, 20
54. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: Proceedings of the British Machine Vision Conference (BMVC) (2016) 9
55. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017) 1, 9, 12, 14, 20
56. Zhou, B., Lapedriza, A., Xiao, J., Torralba, A., Oliva, A.: Learning deep features for scene recognition using places database. In: Advances in Neural Information Processing Systems (NeurIPS). pp. 487–495 (2014) 8, 20
A Appendix

A.1 More Implementation Details

Dataset information. We briefly introduce image datasets used in Section 4. (1) Small-scale classification benchmarks: CIFAR-10/100 [24] contains 50,000 training images and 10,000 test images in 32×32 resolutions, with 10 and 100 classes settings. (2) Large-scale classification benchmarks: ImageNet-1k (IN-1k) [25] contains 1,281,167 training images and 50,000 validation images of 1000 classes. Tiny-ImageNet (Tiny) [5] is a rescale version of ImageNet-1k, which has 10,000 training images and 10,000 validation images of 200 classes in 64×64 resolutions. (3) Small-scale fine-grained classification scenarios: CUB-200-2011 (CUB) [46] contains 11,788 images from 200 wild bird species for fine-grained classification. FGVC-Aircraft (Aircraft) [32] contains 10,000 images of 100 classes of aircrafts. (4) Large-scale fine-grained classification datasets iNaturalist2017/2018 (iNat2017/2018) [21] contains a total of 5,089 categories with 579,184 training images and 95,986 validation images. (5) Scenic classification dataset Place205 [56] contains around 2,500,000 images from 205 common scene categories. Notice that we use modified structures [18] of ResNet and ResNeXt for CIFAR-10/100 and Tiny experiments, i.e., replacing the 7×7 convolution and MaxPooling by a 3×3 convolution, while using normal structures.

Training settings. Detailed training settings of PyTorch [33], DeiT [42], and RSB A2/A3 [49] on ImageNet-1k are provided in Table 12. Notice that we replace the step learning rate decay by Cosine Scheduler [30] and remove ColorJitter and PCA lighting in PyTorch training setting for better performances.

Reproduction settings. We adopt OpenMixup¹ implemented in PyTorch [33] as the open-source codebase, where we implement AutoMix and reproduce most comparison methods (Mixup [55], CutMix [53], ManifoldMix [45], PuzzleMix [23], SaliencyMix [43], FMix [15], and ResizeMix [35]). Notice that optimization-based methods adopt a consistent α for all datasets, PuzzleMix adopts α = 1, CoMixup and AutoMix adopts α = 2. Hand-crafted methods use dataset-specific hyper-parameter settings as follows: For CIFAR-10/100, Mixup and ResizeMix use α = 1, and CutMix, FMix and SaliencyMix use α = 0.2, and ManifoldMix uses α = 2, respectively. For Tiny-ImageNet, ImageNet-1k, iNat2017/2018, and Place205 using PyTorch-style training setting, ManifoldMix uses α = 0.2 and the rest methods adopt α = 1 for median and large backbones (e.g., ResNet-50), while all these methods use α = 0.2 for ResNet-18. For ImageNet-1k using DeiT and RSB A2/A3 settings, all these methods use α = 0.2. For small-scale fine-grained datasets (CUB-200 and Aircraft), SaliencyMix and FMix use α = 0.2, and ManifoldMix uses α = 0.5, while the rest use α = 1. As for other methods, we reproduce results of AugMix [20], Co-Mixup [22], and SuperMix [7] with their official implementations.

¹ https://github.com/Westlake-AI/openmixup
Table 12. Ingredients and hyper-parameters used for ImageNet-1k training settings.

| Procedure       | PyTorch | DeiT   | RSB A2 | RSB A3 |
|-----------------|---------|--------|--------|--------|
| Train Res       | 224     | 224    | 224    | 160    |
| Test Res        | 224     | 224    | 224    | 224    |
| Test crop ratio | 0.875   | 0.875  | 0.95   | 0.95   |
| Epochs          | 100/300 | 300    | 300    | 100    |
| Batch size      | 256     | 1024   | 2048   | 2048   |
| Optimizer       | SGD     | AdamW  | LAMB   | LAMB   |
| LR              | 0.1     | $1 \times 10^{-3}$ | $5 \times 10^{-3}$ | $8 \times 10^{-3}$ |
| LR decay        | cosine  | cosine | cosine | cosine |
| Weight decay    | $10^{-4}$ | 0.05   | 0.02   | 0.02   |
| Warmup epochs   | X       | 5      | 5      | 5      |
| Label smoothing | X       | 0.1    | X      | X      |
| Dropout         | X       | X      | X      | X      |
| Stoch. Depth    | X       | 0.1    | 0.05   | X      |
| Repeated Aug    | X       | ✓      | ✓      | ✓      |
| Gradient Clip.  | X       | 1.0    | X      | X      |
| H. flip         | ✓       | ✓      | ✓      | ✓      |
| RRC             | ✓       | ✓      | ✓      | ✓      |
| Rand Augment    | X       | 9/0.5  | 7/0.5  | 6/0.5  |
| Auto Augment    | X       | X      | X      | X      |
| Mixup alpha     | X       | 0.8    | 0.1    | 0.1    |
| Cutmix alpha    | X       | 1.0    | 1.0    | 1.0    |
| Erasing prob.   | X       | 0.25   | X      | X      |
| ColorJitter     | X       | X      | X      | X      |
| EMA             | X       | ✓      | X      | X      |
| CE loss         | ✓       | ✓      | X      | X      |
| BCE loss        | X       | X      | ✓      | ✓      |
| Mixed precision | X       | X      | ✓      | ✓      |

A.2 More Experiments and Ablations

More Experiments. We evaluate AutoMix for various training epochs on CIFAR-10/100 based on ResNet-18 (R-18) and ResNeXt-50 (RX-50), as shown in Table 13 and Table 14. It is worth noting that some methods converge fast while suffering performance decay with longer train times, such as CutMix and SaliencyMix, and some methods perform better when train longer, such as ManifoldMix training 1200 epochs. Unlike these methods, AutoMix steadily outperforms them by a large margin regardless of the training time setting.

Hyperparameters for AutoMix. We further analyze the hyper-parameter setting for AutoMix with extra ablation studies conducted on Tiny-ImageNet
Fig. 10. Top-1 accuracy ablation study on feature layer $l$.

and ImageNet-1k with various network architectures. As the same conclusion we provided in main body of experiment, the result in Figure 10 also recommends the choice of $l = 3$, which reflects the hyper-parameter robustness of AutoMix.

### A.3 Architecture of Network

The detailed structure of AutoMix is illustrated in Figure 11. Similar to the flow chart in the method, the module colored as blue can be updated by backpropagation but not green. Furthermore, the dotted line means stop-gradient. Notice that we use the encoder $k$ for inference and drop $M_\phi$ after training. The training process contains three steps: (1) using the momentum encoder $k$ to generate the feature maps $z$ for $M_\phi$; (2) generating $X_{mix}^q$ and $X_{mix}^k$ based on two mixing factors $\lambda_q$ and $\lambda_k$ and the feature maps; (3) training the active encoder $q$ with mixed samples $X_{mix}^q$ and optimizing $M_\phi$ with $X_{mix}^k$ separately.

![Figure 11](image-url)  
**Fig. 11.** The network architecture of AutoMix. The parameters in blue modules (active) are updated by backpropagation while the green (freeze) using momentum update in Equation 12.

Table 13. Top-1 accuracy (%)↑ on CIFAR-10 based on ResNet-18 and ResNeXt-50 (32x4d) trained with various epochs. * denotes unpublished open-source work on arxiv.

| Backbone | R-18 | RX-50 |
|----------|------|-------|
| Epoch    | 200ep| 400ep| 800ep| 1200ep|
|          | 200ep| 400ep| 800ep| 1200ep|
| Vanilla  | 94.87| 95.10| 95.50| 95.59| 95.92| 95.81| 96.23| 96.26|
| MixUp    | 95.70| 96.55| 96.62| 96.84| 96.88| 97.19| 97.30| 97.33|
| CutMix   | 96.11| 96.13| 96.68| 96.56| 96.78| 96.54| 96.60| 96.35|
| ManifoldMix | 96.04| 96.57| 96.71| 97.02| 96.97| 97.39| 97.33| 97.36|
| SaliencyMix | 96.05| 96.42| 96.20| 96.18| 96.65| 96.89| 96.70| 96.60|
| FMix*    | 96.17| 96.53| 96.18| 96.01| 96.72| 96.76| 96.76| 96.10|
| PuzzleMix | 96.42| 96.87| 97.10| 97.03| 97.05| 97.24| 97.27| 97.34|
| ResizeMix* | 96.16| 96.91| 96.76| 97.04| 97.02| 97.38| 97.21| 97.36|
| **AutoMix** | **96.59**| **97.08**| **97.22**| **97.30**| **97.19**| **97.42**| **97.65**| **97.51**|
| Gain     | +0.17| +0.17| +0.12| +0.26| +0.14| +0.04| +0.32| +0.15|
Table 14. Top-1 accuracy (%)† on CIFAR-100 based on ResNet-18 and ResNeXt-50 (32x4d) trained with various epochs.

| Backbone      | R-18   | RX-50  |
|---------------|--------|--------|
| Epoch         | 200ep  | 400ep  | 800ep  | 1200ep |
|               | 200ep  | 400ep  | 800ep  | 1200ep |
| Vanilla       | 76.42  | 77.73  | 78.04  | 78.55  | 79.37  | 80.24  | 81.09  | 81.32  |
| MixUp         | 78.52  | 79.34  | 79.12  | 79.24  | 81.18  | 82.54  | 82.10  | 81.77  |
| CutMix        | 79.45  | 79.58  | 78.17  | 78.29  | 81.52  | 78.52  | 78.32  | 77.17  |
| ManifoldMix   | 79.18  | 80.18  | 80.35  | 80.21  | 81.59  | 82.56  | 82.88  | 83.28  |
| SaliencyMix   | 79.75  | 79.64  | 79.12  | 77.66  | 80.72  | 78.63  | 78.77  | 77.51  |
| FMix∗         | 78.91  | 79.91  | 79.69  | 79.50  | 79.87  | 78.99  | 79.02  | 78.24  |
| PuzzleMix     | 79.96  | 80.82  | 81.13  | 81.10  | 81.69  | 82.84  | 82.85  | 82.93  |
| Co-Mixup      | 80.01  | 80.87  | 81.17  | 81.18  | 81.73  | 82.88  | 82.91  | 82.97  |
| ResizeMix∗    | 79.56  | 79.19  | 80.01  | 79.23  | 79.56  | 79.78  | 80.35  | 79.73  |
| **AutoMix**   | **80.12** | **81.78** | **82.04** | **81.95** | **82.84** | **83.32** | **83.64** | **83.80** |
| Gain          | +0.11  | +0.91  | +0.87  | +0.77  | +1.11  | +0.44  | +0.76  | +0.52  |

A.4 Algorithm of AutoMix

We provide the pseudo code of AutoMix in Pytorch style:

**Algorithm 1** Pseudocode AutoMix in Pytorch style.

```python
# f_q, f_k, M: encoder networks and MixBlock
# lam_q, lam_k: sampled from Beta distribution
# idx_q, idx_k: rearrange index
# n: momentum coefficient

f_k.params = f_q.params  # initialize
for x, y in loader:  # load a minibatch
    # two different permutations of data pairs
    x_q, x_k = x[idx_q], x[idx_k]
    y_q, y_k = y[idx_q], y[idx_k]
    lat_f = f_k(x)  # hidden representation and logits: NxCxWxH

    # generate mixing sample, no gradient to q
    m_q, m_k = M(x, [lam_q, lam_k], [idx_q, idx_k], lat_f)
    logits_mix_k = f_k(m_k)  # mixed logits: NxQ
    logits_cls_q, logits_mix_q = f_q(x), f_q(m_q)  # one-hot logits: NxC

    # mixup cross-entropy losses for q and M
    loss_cls = ClassificationLoss(lam_q, logits_mix_q, y)  # including one-hot CE loss
    loss_gen = GenerationLoss(lam_k, logits_mix_k, y)  # including loss_lambda
    loss = loss_cls + loss_gen

    loss.backward()
    update(f_q.params, M.params)  # SGD update (q and M)
    f_k.params = m*f_k+(1-m)*f_q.params  # momentum update
```
Fig. 12. Visualization of mixed samples on ImageNet-1k. The upper part presents the plot of mixed samples from AutoMix ($t = 3$) for $\lambda = 0.5$; the lower shows the mixed samples when different $\lambda$ values are taken.
Fig. 13. Visualization of mixed samples on ImageNet-1k. The upper part presents the plot of mixed samples from AutoMix ($l = 3$) for $\lambda = 0.5$; the lower offers the mixed samples when different $\lambda$ values are taken.