AutoMix: Unveiling the Power of Mixup for Stronger Classifiers

Zicheng Liu[1,2]⋆, Siyuan Li[1,2]⋆, Di Wu[1,2], Zihan Liu[1,2],
Zhiyuan Chen[2], Lirong Wu[1,2], and Stan Z. Li[2,✉]

[1] Zhejiang University, Hangzhou, 310000, China
[2] AI Lab, School of Engineering, Westlake University, Hangzhou, 310000, China
`{liuzicheng, lisiyuan, wudi, liuzihan, chenzhiyuan, wulirong,`
`stan.z.li}@westlake.edu.cn`
⋆ Equal contribution, ✉ Corresponding author

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 image processing [63,56,43,44,65], graph learning [58,55,3], and video processing [26,28,7,29]. 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 [9,47], are prone to over-fitting and overconfidence [14], resulting in poor generalization abilities [53,42,1].

To improve the generalization of DNNs, a series of data mixing augmentation techniques emerged. As shown in Figure 1, MixUp [64] generates augmented
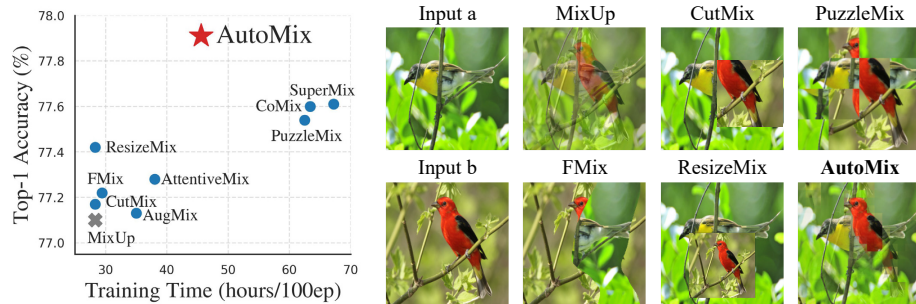
Fig. 1. The plot of efficiency *vs.* accuracy on ImageNet-1k and visualization of mixup methods. AutoMix improves performance without the heavy computational overhead.

samples via a linear combination of corresponding data pairs; CutMix [61] designs a patch replacement strategy that randomly replaces a patch in an image with patches from the other image. However, these *hand-crafted* methods [50,11,15] cannot guarantee mixed samples containing target objects and might cause the *label mismatch* problem. Subsequently, [52,48,37] 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 [8,22,21] 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 $f_\theta : x \longmapsto y$ with network parameters θ by optimizing a classification loss $\ell(.)$, say the cross entropy (CE) loss,

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

Then we consider the mixup classification task: given a sample mixup function h, a label mixup function g, and a mixing ratio λ 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 $f_\theta : x_{mix} \longmapsto y_{mix}$ by mixup cross-entropy (MCE) loss,

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

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_θ 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_ϕ as the sub-task with another set of parameters ϕ. The final goal is to optimize ℓ_{MCE} given θ and ϕ as below:

$$\min_{\theta, \, \phi} \ell_{MCE}\Big(f_\theta\big(h_\phi(x_i, x_j, \lambda)\big), g(y_i, y_j, \lambda)\Big), \tag{3}$$
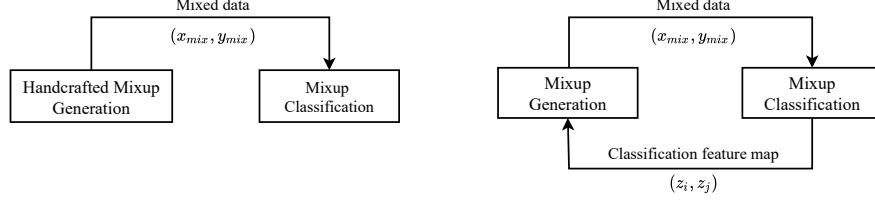
4 Liu. et al.

Fig. 2. The difference between AutoMix and offline approaches. **Left**: Offline mixup methods, where a fixed mixup policy generates mixed samples for the classifier to learn from. **Right**: AutoMix, where the mixup policy is trained with the feature map.

Offline mixup limits the power of mixup. Keep the reformulation in mind, the previous methods focus on manually designing $h(\cdot)$ in an offline and non-parametric manner based on their prior hypotheses, or arguably, such mixup policies are separated from the ultimate optimization of the model, e.g., an optimization algorithm with the goal of maximizing saliency information. Specifically, they build an implicit connection between the two sub-tasks, as shown in the left of Figure 2. Therefore, the mixed samples generated from these offline mixup policies could be redundant or mislead the training. To address this, we propose AutoMix, *which combines these two sub-tasks in a mutually beneficial manner and unveils the power of mixup.*

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.
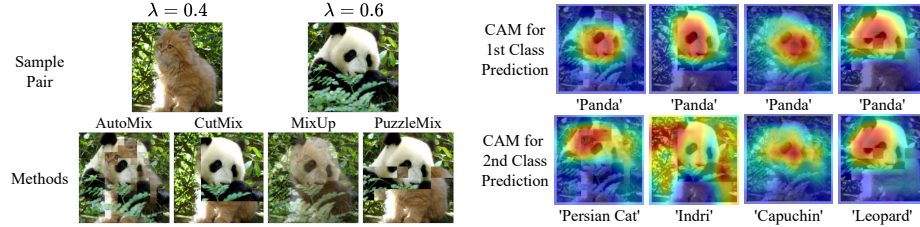

Fig. 3. Illustration of *label mismatch* by visualizing mixed samples and class activation mapping (CAM) [41] on 'Panda' and 'Persian Cat'. From top to bottom rows, we show the original images, mixed images, CAM for top-2 predicted classes, respectively.
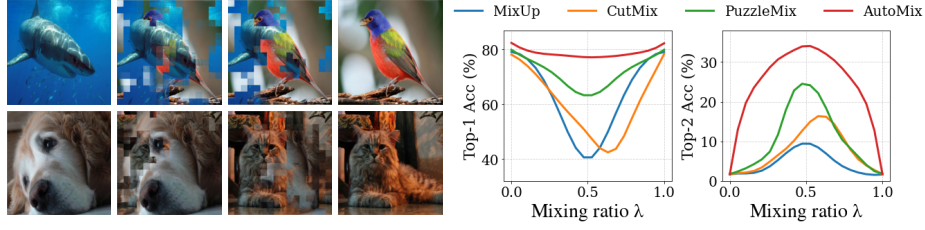
Fig. 4. Left: AutoMix samples with different λ (0, 0.3, 0.7, 1). **Right**: Top-1 accuracy of mixed data. Prediction is counted as correct if the top-1 prediction belongs to $\{y_i, y_j\}$; Top-2 accuracy is calculated by counting the top-2 predictions are equal to $\{y_i, y_j\}$.

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 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 $s_{w,h} \in [0, 1]$. We regard the mask-based mixup policy as an adaptive selection process in terms of λ, 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 λ 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_θ 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, ℓ_{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 λ, MB predicts the probability that each pixel (or patch) on x_{mix} belongs to x_i according to the feature map pair (z_i, z_j) and mixing ratio λ. The overall parametric mixup function of AutoMix can be formulated as follows:

$$h_\phi(x_i, x_j, \lambda) = \mathcal{M}_\phi(z_{i,\lambda}^l, z_{j,1-\lambda}^l) \odot x_i + (1 - \mathcal{M}_\phi(z_{i,\lambda}^l, z_{j,1-\lambda}^l)) \odot x_j, \quad (4)$$

where $\odot$ denotes element-wise product; z_λ^l is λ embedded feature map at l-th layer. As shown in the right of Figure 5, we first embed λ with the l-th feature map in a simple and efficient way by concatenating, $z_\lambda^l = \text{concat}(z, \lambda)$, 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
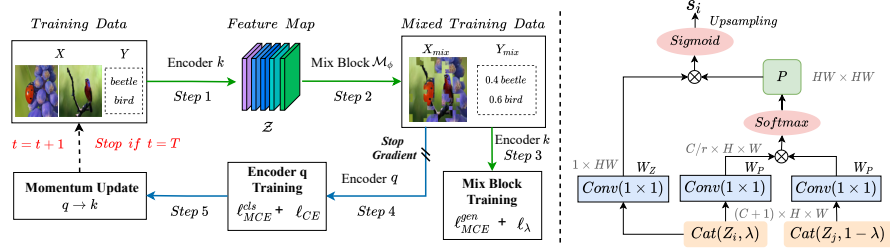
Fig. 5. The **left** diagram represents the five key steps of AutoMix. (1) Extract feature map $\mathcal{Z}$ from the frozen encoder k. (2) Mix Block $\mathcal{M}_\phi$ generates mixed samples by using $\mathcal{Z}$ and mixup ratio $\lambda \in [0,1]$. (3) and (4) Decoupled training $\mathcal{M}_\phi$ and encoder q via *stop gradient*, the blue and green lines indicate the encoder training and the $\mathcal{M}_\phi$ training, correspondingly. (5) Update the k's parameters through momentum moving. The **right** diagram is the architecture of proposed $\mathcal{M}_\phi$.

on λ embedded $z_{i,\lambda}^l$ and $z_{j,1-\lambda}^l$ 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 $\mathcal{M}_\phi : z_{i,\lambda}^l, z_{j,1-\lambda}^l \longrightarrow s_i$,

$$s_i = U\Big(\sigma\big(P(z_{i,\lambda}^l, z_{j,1-\lambda}^l) \otimes W_Z\, z_{i,\lambda}^l\big)\Big), \tag{5}$$

where W_Z is a linear transformation matrix; σ 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, $W_z z_{i,\lambda}^l$, the discriminative features in $x_{i,\lambda}$ relative to $x_{j,1-\lambda}$ are then selected. Symmetrically, the mask s_j for x_j can be calculated in this way, $s_j = 1 - s_i$. Furthermore, the similarity matrix P has to consider both λ information and relative relationships in a sample pair, thus the *cross-attention mechanism* is introduced to achieve this purpose. When x_i in a sample pair (x_i, x_j) is taken as the input, a mask can be generated dynamically from corresponding $z_{i,\lambda}^l$ and P matrix. Formally, our cross-attention can be formulated as:

$$P(z_{i,\lambda}^l, z_{j,1-\lambda}^l) = \mathrm{softmax}\Big(\frac{(W_P\, z_{i,\lambda}^l)^T \otimes W_P\, z_{j,1-\lambda}^l}{C(z_{i,\lambda}^l, z_{j,1-\lambda}^l)}\Big), \tag{6}$$

where W_P denotes shared linear transformation matrices (e.g., 1×1 convolution), $\otimes$ denotes matrix multiplication, and $C(z_{i,\lambda}^l, z_{j,1-\lambda}^l)$ is a normalization factor. Notice that P is the row normalized pair-wise similarity matrix between every spatial position on $z_{i,\lambda}^l$ and $z_{j,1-\lambda}^l$. Similarly, if we take $z_{j,1-\lambda}^l$ as the value, then the mask can be computed by transposing P and $s_i = 1 - s_j$.

AutoMix in end-to-end training. The framework is shown in Figure 5, given a set of labeled data $\mathcal{D} = \{(x_i, y_i)\}_{i=1}^n$ and the corresponding l-th layer

feature map $\mathcal{Z} = \{z_i^l\}_{i=1}^n$, $\mathcal{M}_\phi$ is nested in encoder for optimization. Under the supervision of the same loss ℓ_{MCE}, the encoder is trained using the mixed sample generated by $\mathcal{M}_\phi$, which in turn uses backbone's feature to generate the mixed sample. To enable $\mathcal{M}_\phi$ to find the λ correspondence between the x_{mix} and y_{mix} at the early stage of training, our auxiliary loss is proposed:

$$\ell_\lambda = \gamma \max \left(||\lambda - \frac{1}{HW} \sum_{h,w} s_{i,h,w}|| - \epsilon, 0 \right), \tag{7}$$

where γ is a loss weight linearly decreased to 0 during training. We set the initial γ to 0.1 and $\epsilon = 0.1$. Notice that AutoMix uses standard cross-entropy loss ℓ_{CE} as default. ℓ_{CE} loss facilitates the backbone to provide a stable feature map at the early stage so that speeds up $\mathcal{M}_\phi$ converges. To differentiate the function of ℓ_{MCE}, cls denotes classification task for training encoder and gen denotes generation task for training $\mathcal{M}_\phi$. AutoMix can be optimized by a joint loss:

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

Obviously, the purpose of the classification task is to optimize θ while the generation task is to optimize ϕ. 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 θ and ϕ will lead to instability. To address this problem properly, we use the momentum pipeline to decouple the training of θ and ϕ. As indicated in Eq. 8, though the same ℓ_{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 Eq. 3 and Eq. 8, for each iteration, the gradient entanglement problem of $\mathcal{L}^{cls}$ in $\mathcal{M}_\phi$ can be formulated as
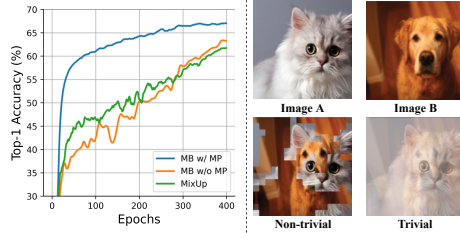

Fig. 6. Accuracy on Tiny-ImageNet and different results of the mixed sample. Momentum pipeline decoupled mixup generation and classification, which mitigates the trivial solution problem.
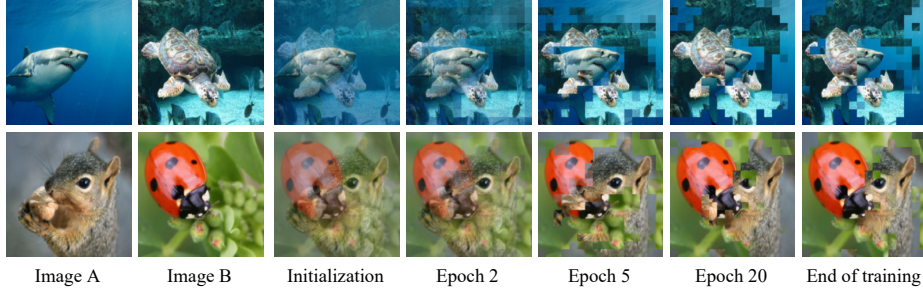
$$\nabla_\phi \mathcal{L}_{MCE}^{cls} \propto \nabla_\phi h_\phi(x_i, x_j, \lambda) \odot f_\theta'(h_\phi(x_i, x_j, \lambda)). \tag{9}$$

| Image A | Image B | Initialization | Epoch 2 | Epoch 5 | Epoch 20 | End of training |

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.

It is notable that the instability of f_θ 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_θ outputs stable features and, to the extent possible, that ϕ and θ 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_{θ_k} of the Siamese network shows more stable performance than student network f_{θ_q}. Along this path, we designed a new MP for decoupling the nested bi-level optimization problem of AutoMix: the student network f_{θ_q} focuses on the classification task, while the stable teacher network f_{θ_k} is connected with $\mathcal{M}_\phi$ to perform generation task. Moreover, optimizing Eq. 8 with batch approach requires X_{mix} generated by f_{θ_k} and $\mathcal{M}_\phi$ first and then using X_{mix} to optimize f_{θ_q}. By analogy, referring to the Expectation-Maximization (EM) algorithm, the two sets of parameters θ and ϕ can be optimized in an alternating way by the designed MP, i.e., first fix one set of parameters optimizing the other:

$$\theta_q^t \leftarrow \underset{\theta}{\operatorname{argmin}} \, \mathcal{L}(\theta_q^{t-1}, \phi^{t-1}), \tag{10}$$

$$\phi^t \leftarrow \underset{\phi}{\operatorname{argmin}} \, \mathcal{L}(\theta_k^t, \phi^{t-1}), \tag{11}$$

where t is the iteration step, θ_q and θ_k represent the parameters of student and teacher network, respectively. Note that f_{θ_q} and f_{θ_k} share the same network structure with the same initialized parameters, but f_{θ_k} is updated via an exponential moving average (EMA) strategy [36] from f_{θ_q}:

$$\theta_k \leftarrow m\theta_k + (1 - m)\theta_q, \tag{12}$$

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_θ. 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 [23], Tiny-ImageNet [5], ImageNet-1k [39], CUB-200-2011 (CUB) [51], FGVC-Aircraft (Aircraft) [34], iNaturalist2017/2018 (iNat2017/2018) [20], and Place205 [66]. We verify generalizabilities of AutoMix for **eight network architectures**, the experiments adopt popular ConvNets, including ResNet (R) [17], Wide-ResNet (WRN) [62], ResNeXt (32x4d) (RX) [59], MobileNet.V2 [40], EfficientNet [45], and ConvNeXt [31], and Transformer-based architectures (DeiT [47] and Swin Transformer (Swin) [30]) as backbone networks. For a fair comparison, we use open-source codebase OpenMixup [25] for most mixup methods: (i) *hand-crafted* methods: Mixup [64], CutMix [61], ManifoldMix [50], AugMix [19], AttentiveMix [52], SaliencyMix [48], FMix [15], and ResizeMix [37]; (ii) *optimization-based* methods: PuzzleMix [22], Co-Mixup [21], and SuperMix [8]. Notice that AugMix is reproduced by timm [54], $*$ denotes open-source arXiv preprint work, and methods without source codes (AlignMix [49] and TransMix [2]) are not compared. All mixup methods use the optimal α 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	67.83
Co-Mixup	97.15	97.32	81.17	82.91	85.05	65.92	68.02
ResizeMix*	96.76	97.21	80.01	81.82	84.87	63.74	65.87
AutoMix	**97.34**	**97.65**	**82.04**	**83.64**	**85.18**	**67.33**	**70.72**
Gain	**+0.19**	**+0.32**	**+0.87**	**+0.76**	**+0.13**	**+1.41**	**+2.70**

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 [32]. 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 [17] 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 (+0.32∼1.94%), PuzzleMix (+0.16∼0.91%), and Co-Mixup (+0.13∼0.87%) based on various architectures on CIFAR.

Calibration. DNNs tend to predict over-confidently in classification tasks [46], 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 achieved the best calibration effect among all competitors with the ECE error rate of 2.3%.

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	78.69	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	74.08	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 [61,37]; (b) DeiT setting trains 300 epochs by AdamW optimizer [33] with the batch size of 1024, the basic learning rate of 0.001, and the weight decay of 0.05; (c) timm [54] RSB A2/A3 settings train 300/100 epochs by LAMB optimizer [60] 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.*, $+0.26 \sim 0.44\%$ for 100 epochs and $+0.22 \sim 0.34\%$ for 300 epochs. More results related to Transformer-based architectures are presented in Appendix A.3.

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 3 shows that AutoMix achieves the best performance and noticeably improves the vanilla (2.19%/3.55% on CUB-200 and 1.14%/1.62% on Aircraft), which verifies that AutoMix has strong adaptability to more challenging scenarios.

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 3 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
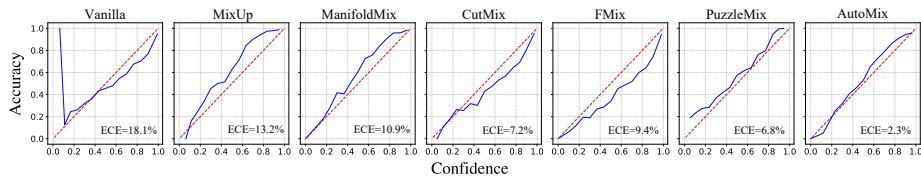

Fig. 8. *Calibration* plots of Mixup variants and AutoMix on CIFAR-100 using ResNet-18. The red line indicates the expected prediction tendency.

Table 3. Top-1 accuracy (%)↑ of various algorithms based on ResNet variants on fine-grained and scenic classification datasets.

Method	CUB-200 R-18	CUB-200 RX-50	FGVC-Aircraft R-18	FGVC-Aircraft RX-50	iNat2017 R-50	iNat2017 RX-101	iNat2018 R-50	iNat2018 RX-101	Place205 R-18	Place205 R-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 4. Top-1 accuracy (%)↑ and FGSM error (%)↓ on CIFAR-100 based on ResNeXt-50 (32x4d) trained 400 epochs.

	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**

Table 5. Trasfer learning of object detection task with Faster-RCNN on Pascal VOC and COCO datasets.

Methods	VOC mAP	COCO mAP	COCO AP_{50}^{bb}	COCO AP_{75}^{bb}
Vanilla	81.0	38.1	59.1	41.8
Mixup	80.7	37.9	59.0	41.7
CutMix	81.9	38.2	59.3	42.0
PuzzleMix	81.9	38.3	59.3	42.1
ResizeMix	82.1	38.4	59.4	42.1
AutoMix	**82.4**	**38.6**	**59.5**	**42.2**

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 [18], which is designed for evaluating the corruption robustness and provides 19 different corruptions (*e.g.*, noise, blur, and digital corruption, *etc*). AugMix [19] 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 4, 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 ℓ_∞ epsilon ball following [64], and AutoMix outperforms previous methods in Table 4.

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

Table 6. MaxBoxAcc (%)↑ for the WSOL task on CUB-200 based on ResNet variants.

Backbone	Vanilla	Mixup	CutMix	FMix*	PuzzleMix	Co-Mixup	**Ours**
R-18	49.91	48.62	51.85	50.30	53.95	54.13	**54.46**
RX-50	53.38	50.27	57.16	59.80	59.34	59.76	**61.05**

in Table 6, which aims to localize objects of interest without bounding boxes supervision. We use CAM to extract attention maps, and caculate the maximal box accuracy with a threshold $\delta \in \{0.3, 0.5, 0.7\}$, following MaxBoxAccV2 [4].

Object detection. We then evaluate transferable abilities of the learned features to object detection task with Faster R-CNN [38] on PASCAL VOC *train-val07+12* [10] and COCO *train2017* [27] based on Detectron2 [57]. 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 5 shows that AutoMix achieves better performances than previous cutting-based mixup variants.

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, λ embedding and ℓ_λ 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 7. (2) In Table 8, 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 EMA, which means f_{θ_k} is a copy of f_{θ_q} with the same weights. Therefore, we can confirm the effectiveness of $\mathcal{M}_\phi$. (3) AutoMix has two core hyper-parameters, α and l, which are fixed for all

Table 7. Ablation of modules in MixBlock.

module	Tiny-ImageNet R-18	RX-50
(random grids)	64.40	66.83
+cross attention	66.87	69.76
+λ embedding	67.15	70.41
+ℓ_λ	**67.33**	**70.72**

Table 8. Ablation of the proposed momentum pipeline (MP) and the cross-entropy loss l_{CE} (CE) based on ResNet-18.

modules	CIFAR-100 MixUp	CutMix	$\mathcal{M}_\phi$	Tiny-ImageNet MixUp	CutMix	$\mathcal{M}_\phi$	ImageNet-1k MixUp	CutMix	$\mathcal{M}_\phi$
(none)	79.12	78.17	79.46	63.39	64.40	64.84	69.98	68.95	70.04
+MP(m=0)	-	-	81.75	-	-	67.05	-	-	70.41
+MP	**80.82**	79.57	81.93	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**

Fig. 9. Ablation of hyperparameter α of AutoMix on CIFAR-100 and Tiny-ImageNet.
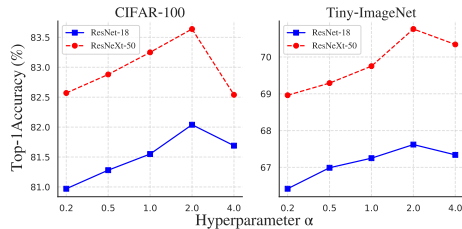
Table 9. Ablation of feature layer l on Tiny-ImageNet, reporting top-1 Acc (%)↑ *vs.* params (M)↓ *vs.* the total training time (hours)↓.

	R-18			RX-50		
	Acc(%)	Params	Time	Acc(%)	Params	Time
Mixup	63.86	11.27	20	66.36	23.38	113
l_1	67.30	11.38	67	70.70	23.80	413
l_2	67.27	11.39	41	70.43	23.86	252
l_3	**67.33**	11.44	34	**70.72**	24.84	196
l_4	67.32	11.64	28	70.67	27.99	174

experiments. A larger α 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 9.

5   Related Work

MixUp [64], 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 [50] extends MixUp to the hidden space of DNNs and [11,49] improves ManifoldMix. CutMix [61] 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 [52] and SaliencyMix [48] 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 [37] maintains the information integrity by replacing one resized image directly into a rectangular area of another image. Furthermore, PuzzleMix [22] and Co-Mixup [21] propose combinatorial optimization strategies to find optimal mixup masks by maximizing the saliency information.

6   Conclusion

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 semantic samples with adaptive masks. Extensive experiments have shown the effectiveness and excellent generalizability of AutoMix on broad 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.

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. Cheng, Z., Liang, J., Choi, H., Tao, G., Cao, Z., Liu, D., Zhang, X.: Physical attack on monocular depth estimation with optimal adversarial patches (2022) 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) 9, 19
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. Cui, Y., Yan, L., Cao, Z., Liu, D.: Tf-blender: Temporal feature blender for video object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV). pp. 8138–8147 (2021) 1
8. 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, 19
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) 8
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., Niranjan, M., Hare, A.P.B.J.: Fmix: Enhancing mixed sample data augmentation. arXiv preprint arXiv:2002.12047 **2**(3), 4 (2020) 2, 9, 19
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) 8
17. 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. 770–778 (2016) 9, 10, 19

18. Hendrycks, D., Dietterich, T.: Benchmarking neural network robustness to common corruptions and perturbations. arXiv preprint arXiv:1903.12261 (2019) 12

19. 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, 19

20. 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) 9, 19

21. 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, 19

22. 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, 19

23. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009) 9, 19

24. 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) 19

25. Li, S., Liu, Z., Wu, D.: OpenMixup: Open mixup toolbox and benchmark for visual representation. https://github.com/Westlake-AI/openmixup (2022) 9

26. Li, S., Zhang, Z., Liu, Z., Wang, A., Qiu, L., Du, F.: Tlpg-tracker: Joint learning of target localization and proposal generation for visual tracking. In: Proceedings of the 29th International Joint Conference on Artificial Intelligence (IJCAI). pp. 708–715 (2020) 1

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, D., Cui, Y., Tan, W., Chen, Y.: Sg-net: Spatial granularity network for one-stage video instance segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 9816–9825 (2021) 1

29. Liu, D., Cui, Y., Yan, L., Mousas, C., Yang, B., Chen, Y.: Densernet: Weakly supervised visual localization using multi-scale feature aggregation. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 6101–6109. No. 7 (2021) 1

30. 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

31. Liu, Z., Mao, H., Wu, C.Y., Feichtenhofer, C., Darrell, T., Xie, S.: A convnet for the 2020s (2022) 9

32. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016) 10, 19

33. Loshchilov, I., Hutter, F.: Decoupled weight decay regularization. In: International Conference on Learning Representations (ICLR) (2019) 11

34. Maji, S., Rahtu, E., Kannala, J., Blaschko, M., Vedaldi, A.: Fine-grained visual classification of aircraft. arXiv preprint arXiv:1306.5151 (2013) 9, 19

35. 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) 19

36. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. SIAM Journal on Control and Optimization pp. 838–855 (1992) 8
37. 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, 11, 14, 19
38. 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) 13
39. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. International journal of computer vision pp. 211–252 (2015) 9
40. Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 9
41. Selvaraju, R.R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., Batra, D.: Gradcam: Visual explanations from deep networks via gradient-based localization. arXiv preprint arXiv:1610.02391 (2019) 4
42. 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
43. Tan, C., Gao, Z., Wu, L., Li, S., Li, S.Z.: Hyperspherical consistency regularization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 7244–7255 (2022) 1
44. Tan, C., Xia, J., Wu, L., Li, S.Z.: Co-learning: Learning from noisy labels with self-supervision. In: Proceedings of the 29th ACM International Conference on Multimedia. pp. 1405–1413 (2021) 1
45. Tan, M., Le, Q.V.: Efficientnet: Rethinking model scaling for convolutional neural networks. In: International Conference on Machine Learning (ICML) (2019) 9
46. 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
47. 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, 19
48. 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, 19
49. Venkataramanan, S., Avrithis, Y., Kijak, E., Amsaleg, L.: Alignmix: Improving representation by interpolating aligned features (2021) 9, 14
50. 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, 19
51. Wah, C., Branson, S., Welinder, P., Perona, P., Belongie, S.: The caltech-ucsd birds-200-2011 dataset. California Institute of Technology (2011) 9, 19
52. 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 Processing (ICASSP). pp. 3642–3646 (2020) 2, 9, 14

53. 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
54. Wightman, R., Touvron, H., Jégou, H.: Resnet strikes back: An improved training procedure in timm (2021) 9, 11, 19
55. Wu, L., Lin, H., Tan, C., Gao, Z., Li, S.Z.: Self-supervised learning on graphs: Contrastive, generative, or predictive. IEEE Transactions on Knowledge and Data Engineering (2021) 1
56. Wu, L., Yuan, L., Zhao, G., Lin, H., Li, S.Z.: Deep clustering and visualization for end-to-end high-dimensional data analysis. IEEE Transactions on Neural Networks and Learning Systems (2022) 1
57. Wu, Y., Kirillov, A., Massa, F., Lo, W.Y., Girshick, R.: Detectron2. https://github.com/facebookresearch/detectron2 (2019) 13
58. Xia, J., Zhu, Y., Du, Y., Li, S.Z.: Pre-training graph neural networks for molecular representations: Retrospect and prospect. In: ICML 2022 2nd AI for Science Workshop (2022) 1
59. Xie, S., Girshick, R., Dollár, P., Tu, Z., He, K.: Aggregated residual transformations for deep neural networks. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 1492–1500 (2017) 9
60. 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
61. 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) 2, 9, 11, 14, 19
62. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: Proceedings of the British Machine Vision Conference (BMVC) (2016) 9
63. Zang, Z., Li, S., Wu, D., Wang, G., Shang, L., Sun, B., Li, H., Li, S.Z.: Dlme: Deep local-flatness manifold embedding (2022) 1
64. 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, 19
65. Zhao, Z., Wu, Z., Zhuang, Y., Li, B., Jia, J.: Tracking objects as pixel-wise distributions (2022) 1
66. 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) 9, 19