AutoMix: Unveiling the Power of Mixup

Zicheng Liu[*1] Siyuan Li[*1] Di Wu[1] Zhiyuan Chen[1]
Lirong Wu[1] Jianzhu Guo[2] Stan Z. Li[1]

[1]AI Lab, School of Engineering, Westlake University &
Institute of Advanced Technology, Westlake Institute for Advanced Study
Hangzhou, Zhejiang, China
[2]CBSR & NLPR, Institute of Automation, Chinese Academy of Sciences,
Beijing, China

Abstract

Mixup-based data augmentation has achieved great success as regularizer for deep neural networks. However, existing mixup methods require explicitly designed mixup policies. In this paper, we present a flexible, general Automatic Mixup (AutoMix) framework which utilizes discriminative features to learn a sample mixing policy adaptively. We regard mixup as a pretext task for classification and split it into two sub-problems: ***mixed samples generation*** *and* ***mixup classification****. To this end, we design a lightweight mix block to generate synthetic samples based on feature maps and mix labels. Since the two sub-problems are in the nature of Expectation-Maximization (EM), we propose a momentum training pipeline to optimize the mixup process and mixup classification process alternatively in an end-to-end fashion. Extensive experiments on six popular classification benchmarks show that AutoMix consistently outperforms other leading mixup methods and improves generalization abilities to downstream tasks. We hope AutoMix will motivate the community to rethink the role of mixup in representation learning. The code will be released soon.*

1. Introduction

Recent years have witnessed great success of Deep Neural Networks (DNNs) in various tasks, such as object recognition [15, 21], semantic segmentation [26, 4, 27], natural language processing [9], and reinforcement learning [28]. Most of these successes can be attributed to the use of complex network architectures with a numerous parameters and sufficient amount of data. Complex network architectures enable powerful feature extraction capabilities. However, in the case of insufficient data, models with high complexity are also prone to suffer from overfitting, resulting in poor generalization [29]. Moreover, this problem may be further worsened for samples with batch effects [1].

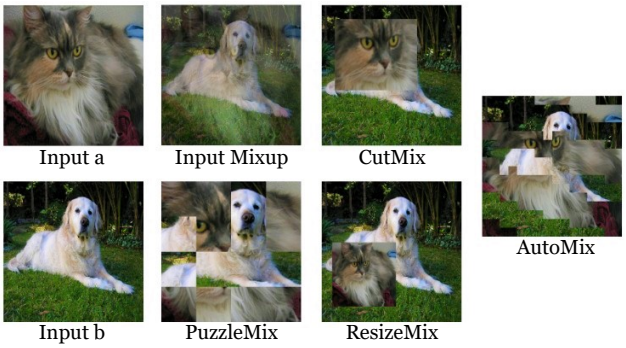

Figure 1: Visualization comparison of different mixup methods. AutoMix generates mixed samples by a mask learned adaptively rather than predefined mixup policies.

In order to improve the generalization ability of DNNs, several data augmentation strategies have been studied. In particular, a series of data augmentation research aims at mixing data via convex combination or local replacement to increase data diversity. CutMix [35] randomly replaces a patch of an image with a patch from another image and then mixes the labels of the two images based on the ratio of mixed pixels. However, since the patch is cut randomly from the image and thus the cut patch is not guaranteed to contain the target object. It might cause *label misallocation* where the mixed label might not be in correspondence with the mixed image. To address this problem, SaliencyMix[31] and PuzzleMix[17] introduce more precise mixing strategies based on the saliency information of images. Besides, ResizeMix[25] improve the cutting manner by preserving

[*]Equal contribution.

substantial information for mixing. Despite their demonstrated effectiveness, mixup is still regarded as one of the augmentation policies by mixing paired samples which litmits its power in classification.

To summarize, the mixing policies in existing methods are hand-crafted to some extent and can we parameterize the mixup process and optimize it alongside a classification task? To answer this question, we propose a novel mixup framework named AutoMix to unveil the power of mixup. The proposed AutoMix allows an automatic image/label mixup process in the mix block which is optimized together with the classification task in an end-to-end fashion rather than a hand-crafted policy. We also propose an efficient momentum-based pipeline to optimize the mixed samples generation task and the classification task alternatively. With the proposed pipeline, the model converges significantly faster with SGD. Extensive experiments on six classification benchmarks including CIFAR-10/100, Tiny-ImageNet, ImageNet, STL-10, and CUB-200, show that our AutoMix significantly ourperforms other mixup methods.

Our main contributions are summarized as follows:

- We regard mixup as a pretext task and decompose it into two sub-problems: mixed sample generation and mixed label classification. Based on that, we propose a lightweight mix block to learn the data mixing process automatically in lieu of hand-crafted policies, as done by previous methods.
- Propose a momentum training pipeline that uses momentum encoder and stop-gradient to optimize two sets of parameters for the mixup and classification tasks. The proposed pipline improves the stability and convergence speed of the mixup training process.
- Extensive experiments on classification benchmarks and the down-stream task show that Automix outperforms existing mixup methods by a large margin.

2. Related Works

Data augmentation methods aim to regularize the model to avoid over-fitting and improve generalization performance. [2] Specifically, we categorize these methods into two types: data-independent and data-dependent.

2.1. Mixup

As a data-dependent regularization method, input mixup applies convex interpolation between two images and its one-hot labels which makes the decision boundary smoother and thus alleviate the over-fitting problem [36]. Manifold mixup extends input mixup to hidden spaces, creating virtual samples between the feature maps [32]. AdaMixUp [13] aims to determine the sample's ratio for bilinear interpolation via network learning to alleviate the manifold intrusion issue. In addition, CutMix [35] incorporates the dropout strategy into input mixup and design a new sample mixup function, which randomly replaces a rectangle with the input data while the labels are mixed according to the area. Instead, ResizeMix [25] turns the same-scale replacement into resize and paste. However, completely random mixups can sometimes be misleading to the model, especially when the mixed sample does not match the mixed label (e.g., cropping a large background onto another image). Therefore, some recent methods introduce saliency information into the mixup policy. For example, PuzzleMix [17] learns to find the optimal masks and transport plans for mixup by maximizing saliency information. Similar to PuzzleMix, SaliencyMix [31] choose saliency regions in their mixup policy. To generate synthetic samples more visually meaningful for human, [37] trains a mixed image generator network adversarially to fit the barycenter. Moreover, Fmix [14] utilizes a mask obtained from Fourier space by setting a threshold to mixup. Unlike these methods, our approach does not require a predefined mixup policy but updates the mixup strategy online alongside the classification task.

2.2. Automatic data augmentation

As for data-independent augmentations, some simple image transformations are often used as augmentation operations, such as rotate, flip, crop, etc [19]. Recently, thanks to the development of neural architecture search (NAS), automatic data augmentation has emerged with some influential works [38, 23, 11]. For example, AutoAugment [7] searches optimal combinations of augmentation operations and scales applicable to a given dataset. To improve computational efficiency, PBA [16] and FastAA [20] strives to reduce the computation effort without trading off performance. Furthermore, RandAugment [8] reduces the search space to only two parameters, but still achieves decent results. In this paper, the AutoMix framework learns to generate mixed samples by the mix block and employs the momentum pipeline to optimize both the mixup sample generation and classification tasks.

3. AutoMix

In this section, we first describe mixup as a pretext task for classification and introduce the proposed AutoMix in Sec. 3.1. We then detail the proposed momentum training pipline in Sec. 3.2. Finally, we show how to generate mixed sample adaptively by the mix block in Sec. 3.3.

3.1. MixUp as Pretext Task

3.1.1 Preliminaries

Let $x \in \mathbb{R}^{W\times H\times C}$ be the input data and y be its one-hot label. Let $\mathcal{D}$ be the joint distribution over $\mathcal{X} \times \mathcal{Y}$. Given a sample mixup function $h(\cdot)$, a label mixup function $g(\cdot)$,

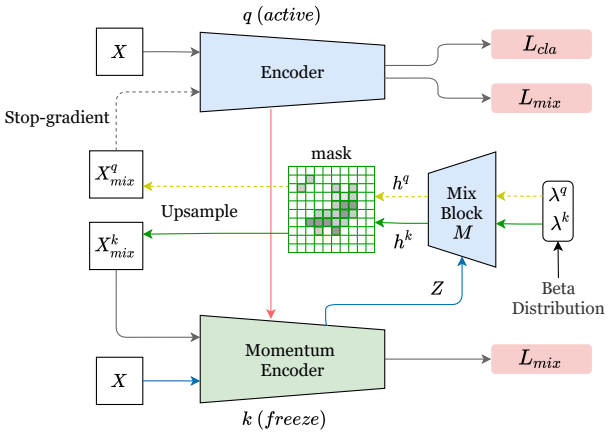

Figure 2: Overview of momentum training pipeline in AutoMix. The solid and dashed lines denote different gradient states, with the former requiring the gradient backpropagation for updating while the latter does not. Similarly, the blue modules is active and can be updated by the backpropagation, whereas the green encoder k is frozen and only updated by the momentum from q. The total process is divided into three steps: (1) using the momentum encoder k to genetate feature map; (2) using mix block $\mathcal{M}$ and two factors λ_q and λ_k to generate X^q_{mix} and X^k_{mix}, shown in green and yellow; (3) forward X, X^q_{mix} and X^k_{mix} through the two encoders. For inference, only the encoder k will be used.

and a mixing distribution $q \in Beta(\alpha, \alpha)$, the goal of mixup is to optimize the loss ℓ as below:

$$\min_{\theta} \mathop{\mathbb{E}}_{(x_i,y_i),(x_j,y_j)\in\mathcal{D}} \mathop{\mathbb{E}}_{\lambda\in q} \ell((h(x_i,x_j), g(y_i,y_j); \theta), \quad (1)$$

where λ is a mixing ratio scalar sampled from q and θ is a set of variables of the classification network $\mathcal{F}$. The label mixup function is $g(y_i, y_j) = (1-\lambda)y_i + \lambda y_j$, while the data mixup function varies in different algorithms. For examples, Input mixup uses $h(x_i, x_j) = (1-\lambda)x_i + \lambda x_j$; Manifold mixup extends it to hidden representation; CutMix combines dropout and mixup to define $h(x_i, x_j) = (1 - \mathbb{I}_B) \odot x_i + \mathbb{I}_B \odot x_j$, where $\mathbb{I}_B$ is a binary rectangular mask with the value of λ is proportional to the area of mask; PuzzleMix utilizes saliency information and employs $h(x_i, x_j) = (1-s) \odot \prod_i^T x_i + s \odot \prod_j^T x_j$ for $s \in \mathbb{R}^{W\times H}$ represents a mask and $s_k \in [0, 1]$ with ratio $\lambda = \frac{1}{n}\sum_k s_k$. $\prod_i$ and $\prod_j$ are the $W \times H$ transportation plans which maximize the saliency information on $h(x_i, x_j)$.

3.1.2 Mixup as Pretext Task

Notice that the label mixup function $g(\cdot)$ generates soft labels based on two hard labels to model the characters between two classes. These soft labels directly affect decision boundaries learned by $\mathcal{F}$ which provides extra prior knowledge. In other words, one-hot labels describe the intra-class relationship, while mixup labels describe the prior inter-class relationship. Therefore, mixup can be regarded as a pretext task complementary to classification. There are two characters of mixup pretext task: (1) mixup tries to embed the prior knowledge of inter-class relationship into $\mathcal{F}$ when given one-hot labels and (2) mixup indirectly optimizes $\mathcal{F}$ by mixed samples which correspond to mixup labels. We can split the mixup into two relevant sub-problems: (i) mixed samples generation and (ii) mixup classification. Both sub-problems can be parameterized and optimized end-to-end with mixup labels.

To solve the sub-problem (i) adaptively, we parameterize the sample mixup function $h(\cdot)$ as a sub-network $\mathcal{M}$ with another set of variables ϕ. $\mathcal{M}$ can be discarded after training. We formulate $h(\cdot)$ in AutoMix as

$$h(x_i, x_j) = (1-u(\mathcal{M}(z_i^l, z_j^l)))\odot x_i + u(\mathcal{M}(z_j))\odot x_j, \quad (2)$$

where z^l is the output feature of $\mathcal{F}(x)$ at the l-th layer, u is an up-sampling function, and $\odot$ represents the element-wise product. The output of $u(M(z_i^l))$ is a mask $s \in \mathbb{R}^{W\times H}$ and $s_{(w,h)} \in [0, 1]$ where (w, h) represents the coordinate on x. Unlike previous mixup methods, the label mixup function $g(\cdot)$ is the same as input mixup which does not require adjusting according to the mask s, as $\mathcal{M}$ captures the relationship between the mixed label and the mask.

3.2. Momentum Training Pipeline

With Eq. 2, we can optimize $\mathcal{F}$ using a multi-task loss, i.e. the conventional one-hot classification loss L_{ce} and mixup classification loss L_{mixup}. They are jointly optimized as following:

$$\mathcal{L}(\theta, \phi) = L_{ce} + L_{mixup}. \quad (3)$$

However, the mixup labels generated by Eq. 2 in L_{mixup} involve both θ and ϕ. We can regard θ as the explicit variables for classification in Eq. 3 and ϕ as the implicit variables for the mixup task in Eq. 2, which is an Expectation-Maximization (EM) like problem. It can be addressed by optimizing in an alternating fashion, fixing one set of variables and solving for the other set. Therefore, we can optimize both θ and ϕ by solving the following alternately:

$$\theta \leftarrow \mathop{argmin}_{\theta} \mathcal{L}(\theta, \phi^{t-1}), \quad (4)$$

$$\phi^t \leftarrow \mathop{argmin}_{\phi} \mathcal{L}(\theta^t, \phi), \quad (5)$$

where t is the alternation step. Inspired by MeanTeacher[30] and BYOL[12], we adopt a momentum training pipeline that contains a Siamese network and a mix block $\mathcal{M}$, as shown in Fig. 2. The active encoder f_q and

Algorithm 1 Pseudocode AutoMix in Pytorch style.

```
# f_q, f_k, M: encoder networks and mix block
# lam_q, lam_k: sampled from Beta distribution
# idx_q, idx_k: rearrange index
# m: momentum coefficient

f_k.params = f_q.params # initialize

for x, y in loader: # load a minibatch
   # two different permutation of data and label
   x_q, x_k = x[index_q],x[index_k]
   y_q, y_k = y[index_q],y[index_k]

   # hidden representation and logits: NxCxWxH
   lat_f = f_k(x)

   # generate mixing sample, no gradient to q
   m_q, m_k = M(x,[lam_q, lam_k],[idx_q, idx_k],lat_f)

   # mixed and cls logits: NxC
   logits_mix_k = f_k(m_k)
   logits_cls_q,logits_mix_q = f_q(x),f_q(m_q)

   # cross entropy losses for q and M
   loss_cls_q = CELoss(logits_cls_q,y)
   loss_mix_q = MIXLoss(lam_q,logits_mix_q,y)
   loss_mix_k = MIXLoss(lam_k,logits_mix_k,y)
   loss = loss_cls_q+loss_mix_q+loss_mix_k

   # SGD update (q and M)
   loss.backward()
   update(f_q.params)
   update(M.params)

   # momentum update
   f_k_params = m*f_k+(1-m)*f_q.params
```

`CELoss`: cross entropy loss; `MIXLoss`: mixed label weighted cross entropy loss

the momentum encoder f_k predict the class of the original and mixed samples, while the mix block $\mathcal{M}$ generates mixed samples at the same time. Specifically, the mix block $\mathcal{M}$ takes the output features of the specified layer of the encoder k as input to generate mask h by training and transforming the features at different granularity levels.

Formally, during training, we apply a multi-task loss on the encoder f_q as $L = L_{cls} + L_{mix}$, where L_{cls} and L_{mix} are standard cross-entropy loss and cross-entropy with mixed labels. For the encoder f_k, only loss L_{mix} takes effects, and its parameters are frozen, which are momentum updated from f_q. The encoder f_k focuses on training the mix block $\mathcal{M}$. Correspondingly, the encoder f_q solely concentrates on training the backbone without affecting $\mathcal{M}$. Our mix block $\mathcal{M}$ allows the generation of adaptive mixing masks for every sample pairs. In this end-to-end system, we rely on dedicated encoders to focus on individual tasks. It decouples the training between the mix block module and backbones. Moreover, this training approach brings a significant gain in the convergence speed. We later show by experiments that this formulation is key for sound mask generation and classification performance.

Momentum Update Optimizing the mix block and encoder simultaneously without decoupling strategy yields poor results. We hypothesize that such failure is due to the number of parameters of $\mathcal{M}$ is excessively small relative to the encoder and thus causes the mixing block to not converge under gradient oscillations. Therefore, on the basis of the Siamese structure, we propose to use momentum update to address this issue.

$$\theta_k \leftarrow g(m)\theta_k + (1 - g(m))\theta_q, \tag{6}$$

where the parameters of f_q and f_k are denoted as θ_q and θ_k, and $m \in [0, 1)$ is the momentum decay coefficient. $g(\cdot)$ is the scheduler function, which adjusts m in a predefined pattern during training. Only θ_q requires gradient for optimization. The momentum update makes the training of mix block more stable and smooth, while the convergence speed is significantly accelerated, the complete experiments are conducted in Sec. 4.

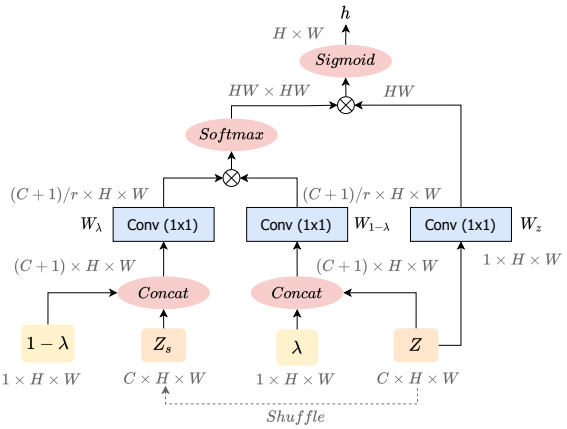

Figure 3: Architecture of the mix block. The shape of feature maps are shown in gray, e.g. $C \times H \times W$ for C channels, H height and W width. $\otimes$ represents matrix multiplication. The softmax operation is performed on each row. The scalar λ is sampled from $Beta(\alpha, \alpha)$ and spread out into a two-dimensional tensor for embedding into the input features Z. Z_s is a batch rearrangement of Z. The blue boxes represent 1×1 convolutions.

3.3. Mix Block $\mathcal{M}$

Here we discuss the proposed mix block $\mathcal{M}$ in detail, which learns to generate mixing masks based on input features and mixup labels. We design the mix block based on an important intuition: modeling the global information of two samples according to the given λ. As shown in Fig. 3, the mix block takes two feature maps and the mixing scalar λ as the input. The global information between the feature pair z^l and z^l_s in the condition of λ is modeled as

$$I(z^l, z^l_s|\lambda) = SoftMax(\frac{W[z^l, \lambda] \otimes W_s[z^l_s, 1 - \lambda]}{C(z^l, z^l_s)}), \tag{7}$$

where W and W_s denote linear transform matrices (e.g., 1x1 concolution), $[\cdot]$ represents concatenation operation, $\otimes$

is matrix multiplication, and $C(z^l, z_s^l)$ is a normalization factor. $I(z^l, z_s^l|)$ denotes the normalized pairwise relationship between every spatial position on z^l and z_s^l. The information of the mixup label is embedded by the concatenation. Based on $I(z^l, z_s^l|)$, the score map h for z^l can be produced as

$$h = Sigmoid(I(z^l, z_s^l|\lambda) \otimes W_z z^l), \tag{8}$$

where h devotes a two-dimensional tensor. Since the mix block only uses 1x1 concolution, it is lightweight to be optimized online. The choice of layers and computational costs of the mix block are discuss in Sec. 4.4.

4. Experiments

In this section, we evaluate AutoMix for its adaptive mixup capability and great generalizability on various tasks. We first study the effectiveness of AutoMix on supervised image classification in Sec. 4.1. To further demonstrate the broad applicability of AutoMix, we test AutoMix under the few-shot classification scenario in Sec. 4.2. Next, we show the transferability of AutoMix as a pre-trained model on object detection task in Sec. 4.3. We also show that AutoMix can improve training stability as well as accelerate training convergence speed. Finally, we prove the effectiveness of AutoMix with respect to each proposed innovation thorough ablation study.

4.1. Evaluation on Image Classification

The performance of AutoMix is evaluated on six image classification datasets including CIFAR-10, CIFAR-100 [18], Tiny ImageNet [5], ImageNet [19], STL-10 [6] and CUB-200 [33]. To demonstrate the applicability of AutoMix to different network architectures, we implement AutoMix on two residual neural networks with different parameter scale: ResNet and ResNeXt (32x4d) [34]. Besides, SGD optimizer with the cosine annealing schedule [24] is used for all experiments. For a fair comparison, grid search is performed for several key hyper-parameters of all mixup algorithms on the basis of vanilla classification, including $\alpha \in \{0.2, 0.5, 1, 2\}$ and cosine scheduler $lr_{min} \in$ {5e-2, 1e-2, 5e-3, 1e-3, 0}; the rest of the hyper-parameters are set the same as the original paper. Spesifically, the momentum decay coefficient m start from 0.999 and is gradually increased to 1 in a cosine curve. AutoMix reports the results of optimal feature layer among $l \in \{0, 1, 2, 3\}$ on each dataset. All experiments were conducted three times and **the averaged median performances of last 10 epochs during training are reported**.

4.1.1 CIFAR

Hyper-parameter settings. The CIFAR dataset contains 50K training images and 10K validation images with 32 × 32 resolution. In our experiments, the RandomCrop and Flipping augmentations are applied as preprocessing. The following hyper-parameters are set the same for all mixup approaches on the both CIFAR-10 and CIFAR-100 dataset: the SGD weight decay is set to 0.0001, the SGD momentum is 0.9, the initial learning rate is 0.1, and the batch size is 100. Then, we perform the grid search to determine the relative optimal hyper-parameters, α and lr_{min} for each mixup algorithm, and report the averaged result. As shown in Tab. 1, on the CIFAR-10 dataset, we set lr_{min} to 0 for every method except AutoMix. In addition, Manifold Mixup and AutoMix adopt larger α, but other compared methods produce better performance with smaller α values. On the other hand, as shown in Tab. 2 and Tab. 3, the lr_{min} of CutMix, FMix and SaliencyMix are raised to 1e-3 on the CIFAR-100 dataset, and their α are the same as CIFAR-10.

Methods	α	lr_{min}	ResNet-18	ResNeXt-50
Vanilla	-	0	95.50	96.23
MixUp	1	0	96.62	97.30
CutMix	0.2	0	96.68	97.01
ManifoldMix	2	0	96.71	97.33
FMix	0.2	0	96.58	96.76
SaliencyMix	0.2	0	96.53	97.18
ResizeMix	1	0	96.76	97.21
PuzzleMix	0.2	0	97.10	-
AutoMix	2	1e-3	**97.14**	**97.84**

Table 1: Top-1 accuracy (%) on CIFAR-10 with ResNet-18 and ResNeXt-50 (32x4d).

Methods	α	lr_{min}	200 ep	400 ep	800 ep	1200 ep
Vanilla	-	0	76.42	78.08	78.04	78.55
MixUp	1	0	78.52	79.32	79.12	79.24
CutMix	0.2	1e-3	79.45	79.88	78.17	78.29
ManifoldMix	2	0	79.18	79.80	80.25	80.21
FMix	0.2	1e-3	78.91	79.91	79.69	79.50
SaliencyMix	0.2	1e-3	**79.75**	79.63	79.12	77.66
ResizeMix	1	0	79.56	79.19	80.01	79.23
PuzzleMix	0.2	0	79.18	80.00	80.14	80.64
AutoMix	2	1e-2	79.31	**80.24**	**81.63**	**80.95**

Table 2: Top-1 accuracy (%) on CIFAR-100 with ResNet-18 trained with various epochs.

Conparison and analysis. The top-1 test accuracy comparisons of CIFAR-10 dataset are shown in Tab. 1. We compare the results of our method with other state-of-the-art mixup methods on ResNet-18/RexNext-50. On the CIFAR-10 dataset, we outperform all existing mixup algorithms, which shows the effectiveness of the proposed AutoMix. In particular, we improve the performance over the vanilla model by 1.61% on the ResNext-50. Moreover, as shown in Tab. 2, we can observe a slight oscillation in the per-

Methods	α	lr_{min}	200 ep	400 ep	800 ep	1200 ep
Vanilla	-	0	79.37	80.24	81.09	81.32
MixUp	1	0	81.18	82.54	82.10	81.77
CutMix	0.2	1e-3	81.52	78.52	78.32	77.17
ManifoldMix	2	0	81.59	82.56	82.88	83.28
FMix	0.2	1e-3	79.87	78.99	79.02	78.24
SaliencyMix	0.2	1e-3	80.72	78.63	78.77	77.51
ResizeMix	1	0	82.33	80.96	79.73	78.7
PuzzleMix	0.2	0	81.69	82.84	82.25	82.85
AutoMix	2	1e-2	**82.84**	**83.69**	**83.04**	**83.80**

Table 3: Top-1 accuracy (%) on CIFAR-100 with ResNeXt-50 (32x4d) trained with various epochs.

formance of methods based on the lightweight ResNet-18 at different training times on the CIFAR-100 dataset. It is worth noting that at 1200 epoch, the performance of most mixup methods shows a deteriorating trend. However, on the complex ResNext-50, AutoMix adapts well to all environments and shows state-of-the-art results and excellent stability. Furthermore, AutoMix outperforms other mixup algorithms by a large margin, with an average of 2% higher performance compared to the vanilla, as in shown Tab. 3.

4.1.2 ImageNet

Hyper-parameter settings. We evaluate the performance of the proposed AutoMix on the ImageNet-1K benchmark which contains 1.28M training images and 50K validation images from 1000 classes with 224×224 resolution. The Tiny-ImageNet dataset is a subset of ImageNet, which contains 200 classes with 500 training images and 50 test images per class with 64×64 resolution. The standard augmentation settings including *RandomResizedCrop* and *Flipping* are used for all mixup algorithms in our experiment. Besides, the SGD weight decay is set to 0.0001 and the SGD momentum is set to 0.9 for both ImageNet-1K and Tiny-ImageNet datasets. For experiments on ImageNet, we train all models for 300 epochs with an initial learning rate 0.1 and the batch size 256. For experiments on Tiny ImageNet, we train all models for 400 epochs with an initial learning rate 0.2 and the batch size 100. Then we briefly describe some dataset-specific hyper-parameter settings for all mixup methods.: we set the parameter α of Manifold Mixup and AutoMix to 0.2 and 2, respectively. And the rest methods' α is set to 1. In contrast to the experimental configuration of ImageNet, the lr_{min} of AutoMix is set to 5e-2 on Tiny ImageNet.

Conparison and analysis. The classification performance on the Tiny-ImageNet and ImageNet dataset are given in Tab. 4 and Tab. 5. In both datasets, AutoMix outperforms all other compared mixup methods in terms of the top-1 accuracy with different network structures. It is worth noting that other mixup methods cannot even match the vanilla

Methods	α	lr_{min}	ResNet-18	ResNeXt-50
Vanilla	0	0	61.68	65.04
MixUp	1	0	63.39	66.36
CutMix	1	0	64.4	66.47
ManifoldMix	0.2	0	62.76	67.30
FMix	1	0	62.28	65.08
SaliencyMix	1	0	64.95	66.55
ResizeMix	1	0	63.50	65.77
PuzzleMix	1	0	65.63	65.02
AutoMix	2	5e-2	**67.33**	**70.72**

Table 4: Top-1 accuracy on Tiny-ImageNet dataset for ResNet-18 and ResNeXt-50 (32x4d) trained with various mixup algorithms.

Methods	α	lr_{min}	ResNet-18	ResNet-50
Vanilla	-	0	71.83	77.35
MixUp	1	0	71.72	78.03
CutMix	1	0	70.03	78.62
ManifoldMix	0.2	0	71.73	78.28
FMix	1	0	70.30	77.61
SaliencyMix	1	0	70.21	78.67
ResizeMix	1	0	71.32	78.85
PuzzleMix	1	0	71.44	78.39
AutoMix	2	0	**71.85**	**79.60**

Table 5: Top-1 accuracy on ImageNet dataset for ResNet-18 and ResNet-50 trained with various mixup algorithms.

when training with lightweight ResNet-18 on the ImageNet dataset, which means those mixup methods fail to work as regularizer. On the contrary, AutoMix can still maintain a stable accuracy in the same settings. Furthermore, as shown in Tab. 4, AutoMix get better results with more complex network architecture. With ResNeXt-50, AutoMix is far ahead of other compared mixup algorithms, achieving 70.72% top-1 accuracy. Moreover, as shown in Fig. 4, AutoMix has the fastest convergence speed, far exceeding other comparison methods on the Tiny-ImageNet dataset, which demonstrates the advantages of optimizing mixup with the momentum training pipeline, resulting in faster convergence and more stable performance.

4.2. Experiments on Few-shot Classification

Hyper-parameter settings. We evaluate AutoMix on two few-shot classification scenarios, CUB-200 [33] and STL-10 [6] dataset. CUB-200 dataset is the most widely used fine-grained dataset, which contains 11,788 images spanning 200 sub-species of birds. STL-10 dataset is usually used for unsupervised learning, but we only use its labeled train set which contains 5K images of 10 classes with

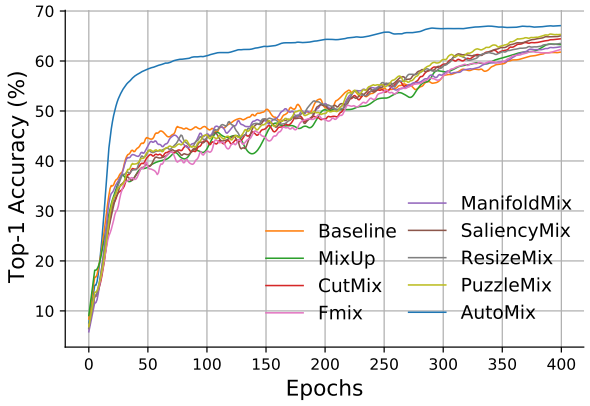

Figure 4: Comparison of the convergence speed and stability of various mixup methods with ResNet-18 on Tiny-ImageNet dataset.

Methods	α	lr_{min}	ResNet-18	ResNeXt-50
Vanilla	-	0	77.36	83.01
MixUp	1	0	78.83	84.58
CutMix	1	0	78.24	85.68
ManifoldMix	0.5	0	**79.44**	86.38
FMix	0.2	0	77.11	84.06
SaliencyMix	0.2	0	77.42	83.29
ResizeMix	1	0	78.51	84.77
PuzzleMix	1	0	78.65	84.51
AutoMix	2	5e-4	79.29	**86.56**

Table 6: Top-1 accuracy (%) on CUB-200 with ResNet-18 and ResNext-50 (32x4d).

96×96 resolution. We follow transfer learning settings on CUB-200 and follow the standard augmentation settings as in Sec. 4.1.2. For experiments on CUB-200, we initialize the model with the ImageNet pertained model, and train all models for 200 epochs with initial learning rate 0.001 and the batch size 16. For experiments on STL-10, we train all models for 800 epochs with initial learning rate 0.1 and the batch size 256. Besides, the SGD weight decay and SGD momentum are set to 0.0005 and 0.9 in this experiment.

Results Comparison. The results in Tab. 6 and Tab. 7 suggest that the proposed AutoMix achieves the best performance on the complex network architecture ResNeXt-50, and notably, we improved the vanilla model by 3.55% and 7.81% on the CUB-200 and STL-10 datasets, respectively. Even on the lightweight ResNet-18, our method still ranks second among all compared methods, only behind the best method Manifold Mix.

Methods	α	lr_{min}	ResNet-18	ResNeXt-50
Vanilla	-	0	76.35	75.47
MixUp	2	0	82.46	82.03
CutMix	0.2	0	80.24	78.33
ManifoldMix	0.5	0	**82.99**	83.12
FMix	0.2	1e-3	79.54	76.34
SaliencyMix	1	5e-3	78.38	76.31
ResizeMix	1	0	81.03	81.25
PuzzleMix	1	5e-3	79.47	78.76
AutoMix	2	5e-2	82.62	**83.28**

Table 7: Top-1 accuracy on STL-10 dataset for ResNet-18 and ResNext-50 (32x4) and 800 epoch trained with various mixup algorithms.

Methods	ImageNet Top-1 Acc (%)	MS-COCO mAP (%)	Pascal VOC mAP (%)
Vanilla	77.4	38.1	81.0
Mixup	78.0	37.6	80.7
CutMix	78.6	38.2	81.9
ResizeMix	78.9	38.4	82.1
AutoMix	**79.6**	**38.7**	**82.6**

Table 8: Generalization ability of ResNet-50 pretrained on ImageNet is evaluated on object detection task with Faster-RCNN on both MS-COCO and Pascal VOC datasets.

4.3. Evaluation on Object Detection

The use of ImageNet pre-training is standard practice for many downstream visual tasks. Therefore, we evaluate the generalization ability of AutoMix by transferring the ImageNet pre-trained model to the other object detection framework named Faster RCNN [26]. We perform the experiments on both MS-COCO [22] and Pascal VOC [10] datasets based on the object detection toolkit MMDetection [3] with pre-trained ResNet-50 as the backbone network.

As shown in Tab. 8, AutoMix shows excellent generalization ability for the object detection task. In a comprehensive comparison, AutoMix results in improvements across the board. In detail, our performance shows remarkable mAP improvements over the baseline, e.g., 0.6% mAP on MS-COCO and 1.6% on Pascal VOC with Faster-RCNN.

4.4. Ablation Studies

In this section, we perform a series of ablation studies with ResNet-18 on CIFAR-100 dataset to evaluate the effectiveness of different components of AutoMix from multiple aspects. The basic experiment setup of AutoMix is the same as CIFAR100 in Sec. 4.1.1. We first analyze the advantage of the momentum training pipeline, where we dive into each component of the momentum pipeline. Then we study the

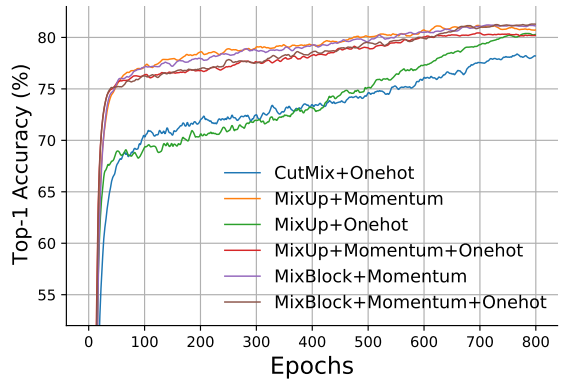

Figure 5: Comparison of the convergence speed and accuracy to evaluate the impact of **Onehot** and **Momentum** on Mixup, CutMix and the mix block on CIFAR-100 dataset.

Module	MixUp	CutMix	MixBlock
(none)	79.12	78.17	78.45
+Onehot	80.30	78.25	79.23
+Momentum	**80.82**	79.54	81.10
+Momentum+Onehot	80.21	**79.56**	**81.13**

Table 9: Comparison of the role of each module in the momentum training pipeline in term of Top-1 Acc on CIFAR-100 dataset.

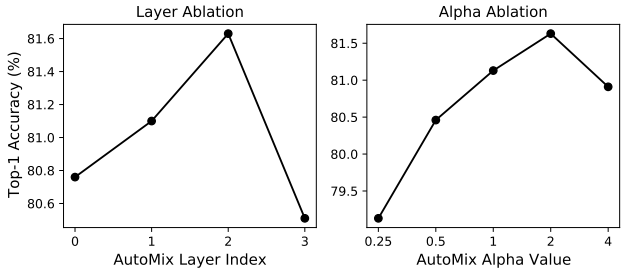

Figure 6: Impact of α and the layer index of MixBlock on CIFAR-100 top-1 accuracy.

mix block module with its hyper-parametric sensitivity.

4.4.1 Momentum Training Pipeline

The proposed momentum training pipeline is incorporated into other mixup methods to verify its effectiveness. We equip Mixup and CutMix with different modules - **Momentum** and **Onehot**, and evaluate their performance on CIFAR-100 dataset. In this experiment, we set $lr_{min} = 0$, $\alpha = 1$ and the mix block $layer = 2$. As shown in Tab. 9, optimizing the mix block directly is 0.67% lower that the MixUp and 2.65% lower than using the momentum module, which indicates that the momentum mechanism is vital

Method	Backbone	Vanilla	L_0	L_1	L_2	L_3
ResNet-18	11.17M	11.27M	11.38M	11.39M	11.44M	11.64M
ResNeXt-50	22.97M	23.38M	23.80M	23.86M	24.84M	27.99M

Table 10: The parameter number of the backbone, baseline and AutoMix varying with the backbone layer index.

lr_{min}	0.	1e-3	5e-3	1e-2	5e-2
Top-1 Acc	80.46	80.78	80.50	**81.63**	80.43

Table 11: Comparison on the effect of different lr_{min} values in AutoMix on CIFAR-100 dataset.

to optimize the mix block. The results in Fig. 5 show that the momentum mechanism can significantly speed up the convergence of the mixup methods which is nearly 10 times faster and improve the classification accuracy. Moreover, the introduction of the Onehot component brings a slight improvement for both AutoMix and other methods.

4.4.2 Mix Block: Hyper-parametric Sensitivity

The previous results demonstrate a significant performance improvement with the introduction of Mix Block, with a nearly 0.92% improvement over Input Mixup. Then we evaluate the hyper-parameter sensitivity of the proposed AutoMix with $\alpha \in \{0.2, 0.5, 1, 2, 4\}$; the results are provided in the right of Fig. 6. Four different values of α are considered here, and the best result is achieved when $\alpha = 2$.

To select the proper backbone layer for the mask generation, we experiment on the CIFAR-100 dataset with parameters configured as ResNet-18 and 800 epochs. The experimental results are shown on the left side of Fig. 6, where we consider the first three layers in the backbone network. We denote the index as (L_0=after the first *conv-bn* on ResNet structure, L_1=after *layer1*, L_2=after *layer2*, L_3=after *layer3*). The results show that AutoMix achieves the best performance at *layer2*. Moreover, according to Tab. 10, the deeper the number of layers selected, the larger the model parameters. Therefore, the output of the second layer is chosen as the input of Mix Block under a comprehensive consideration.

We also explored the best lr_{min} hyper-parameter of cosine scheduler for AutoMix, shown as Tab. 11. From previous experiments, there is no one fixed lr_{min} that is optimal. Thus we choose a different lr_{min} value depending on the dataset. Normally, for the CIFAR-10 dataset, we set the lr_{min} to 1e-3, while for CIFAR-100, the value is 1e-2; Tiny-imageNet 5e-2 and ImageNet 0.

5. Conclusion

In this paper, we propose a *Automatic Mixup* (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, SLT-10, and CUB-200 datasets. On top of that, we also outperformed other Mixup algorithms when transfering the pretrained model to detection tasks as well. Furthermore, the proposed momentum training pipeline serves a significant improvement in convergence speed and overall performance.

References

[1] Shai Ben-David, John Blitzer, Koby Crammer, Alex Kulesza, Fernando Pereira, and Jennifer Wortman Vaughan. A theory of learning from different domains. *Machine learning*, 79(1):151–175, 2010.

[2] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[3] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. 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.

[5] Patryk Chrabaszcz, Ilya Loshchilov, and Frank Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

[6] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[7] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation policies from data. *arXiv preprint arXiv:1805.09501*, 2018.

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Jan. 2015.

[11] Jiemin Fang, Yuzhu Sun, Kangjian Peng, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Fast neural network adaptation via parameter remapping and architecture search. *arXiv preprint arXiv:2001.02525*, 2020.

[12] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

[13] Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 3714–3722, 2019.

[14] Ethan Harris, Antonia Marcu, Matthew Painter, Mahesan Niranjan, and Adam Prügel-Bennett Jonathon Hare. Fmix: Enhancing mixed sample data augmentation. *arXiv preprint arXiv:2002.12047*, 2(3):4, 2020.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel. Population based augmentation: Efficient learning of augmentation policy schedules. In *International Conference on Machine Learning*, pages 2731–2741. PMLR, 2019.

[17] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, pages 5275–5285. PMLR, 2020.

[18] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

[19] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[20] Yonggang Li, Guosheng Hu, Yongtao Wang, Timothy Hospedales, Neil M Robertson, and Yongxin Yang. Differentiable automatic data augmentation. In *European Conference on Computer Vision*, pages 580–595. Springer, 2020.

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pages 2980–2988, 2017.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014.

[23] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[24] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

[25] Jie Qin, Jiemin Fang, Qian Zhang, Wenyu Liu, Xingang Wang, and Xinggang Wang. Resizemix: Mixing data with preserved object information and true labels. *arXiv preprint arXiv:2012.11101*, 2020.

[26] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497*, 2015.

[27] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015.

[28] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[29] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

[30] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. In *31st Conference on Neural Information Processing Systems*, 2017.

[31] AFM Uddin, Mst Monira, Wheemyung Shin, TaeChoong Chung, Sung-Ho Bae, et al. Saliencymix: A saliency guided data augmentation strategy for better regularization. *arXiv preprint arXiv:2006.01791*, 2020.

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

[33] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. California Institute of Technology, 2011.

[34] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500, 2017.

[35] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6023–6032, 2019.

[36] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[37] Jianchao Zhu, Liangliang Shi, Junchi Yan, and Hongyuan Zha. Automix: Mixup networks for sample interpolation via cooperative barycenter learning. In *European Conference on Computer Vision*. Springer, 2020.

[38] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.

A. Ablation Study of AutoMix

We provide more detailed analysis of each module in AutoMix on CIFAR, Tiny ImageNet, ImageNet and STL-10 datasets. And the other not mentioned parameter settings are the same as the corresponding dataset configuration in Sec. 4.

A.1. Momentum Traning Pipeline

To study the effects of the momentum training pipeline in optimizing the mix block, we perform ablation studies on **Momentum** and **Onehot** with ResNet-18 on CIFAR-100 and Tiny ImageNet datasets. In this experiment, we set $lr_{min} = 0$, $\alpha = 1$ and the mix block $layer = 2$. We train models with 800 epochs on CIFAR-100 and 400 epochs on Tiny ImageNet.

As shown in Tab. A1. Overall, the momentum pipeline brings the most significant improvement, the accuracy was elevated by about 2% on each of these two datasets. And On this basis, Onehot plays a supporting role.

Module	CIFAR-100	Tiny ImageNet
MixBlock	78.45	63.49
+Onehot	79.23	63.07
+Momentum	81.10	65.25
+Momentum+Onehot	**81.13**	**65.36**

Table A1: Analysis of the momentum training pipeline in AutoMix in term of Top-1 Acc on CIFAR-100 and Tiny ImageNet datasets.

A.2. Analysis of Feature Layers

We analyze the input feature layers $\{L_0, L_1, L_2, L3\}$ in the mix block with ResNet-18 on various datasets. The basic settings of AutoMix is the same as Sec. 4. We train models with 800 epochs on CIFAR-100, 400 epochs on Tiny ImageNet, 300 epochs on ImageNet, 400 epochs on STL-10.

Shown as Tab. A2, the feature maps of different layers have different performance on these four datasets, but the deeper layers usually perform better. As for large datasets like ImageNet, the layer 1 and 2 perform better. As for the smalle datasets, the layer 2 and 3 perform better.

Layer	CIFAR-100	Tiny ImageNet	ImageNet	STL-10
L_0	80.76	67.40	-	78.61
L_1	81.10	**67.57**	71.34	79.03
L_2	**81.63**	67.33	**71.85**	78.82
L_3	80.51	67.50	71.56	**80.09**

Table A2: Analysis of the momentum training pipeline in AutoMix in term of Top-1 Acc on CIFAR-100 and Tiny ImageNet datasets.

B. AutoMix Mixed Samples Visualization

We visualize the mixed samples generated by AutoMix with the features from $layer1$ to $layer3$ with various resolutions. Fig. A7, Fig. A8, Fig. A10 and Fig. A9 visualize the AutoMix results on CIFAR dataset, Tiny ImageNet, ImageNet and STL-10 datasets.

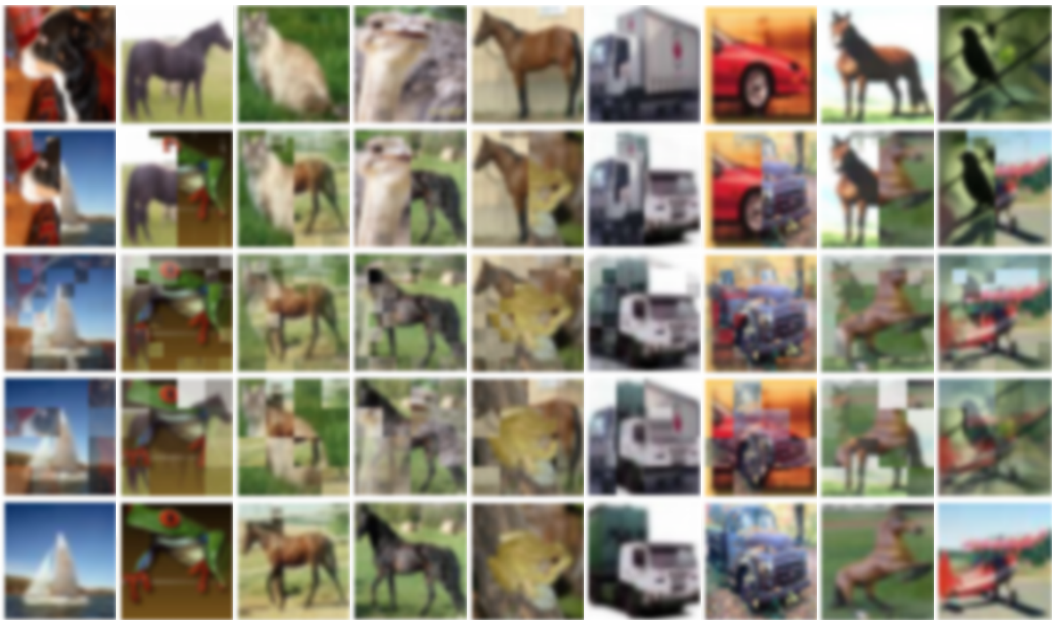

Figure A7: The mixed sample visualization of different feature layers of the CIFAR dataset as input to the mix block, from top to bottom, represents the input image a, the mix visualization from $layer1$ to $layer3$ and the input image b, respectively.

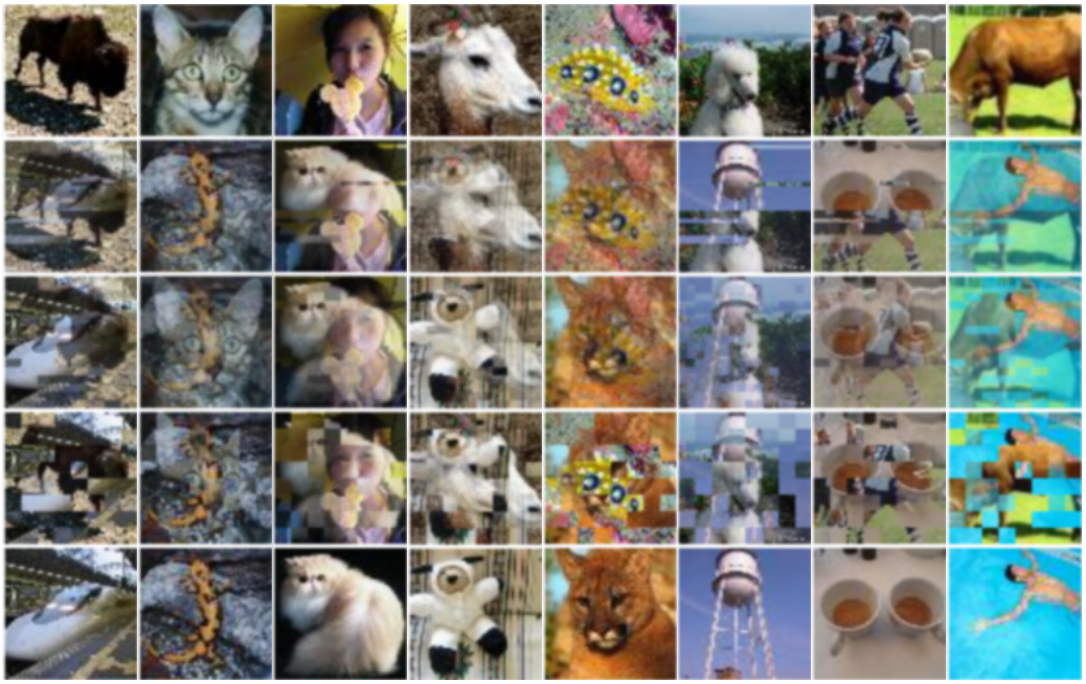

Figure A8: The mixed sample visualization of different feature layers of the Tiny-ImageNet dataset as input to the mix block, from top to bottom, represents the input image a, the mix visualization from $layer1$ to $layer3$ and the input image b.

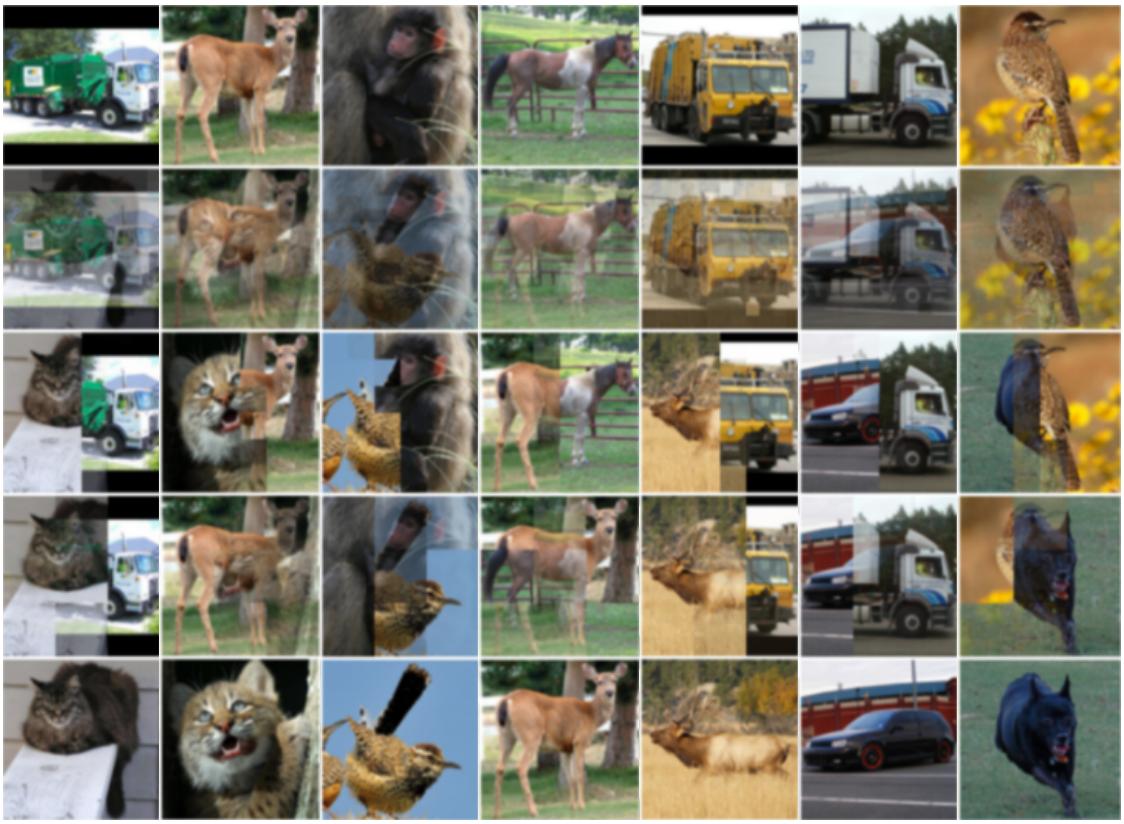

Figure A9: The mixed sample visualization of different feature layers of the STL-10 dataset as input to the mix block, from top to bottom, represents the input image a, the mix visualization from $layer1$ to $layer3$ and the input image b, respectively.

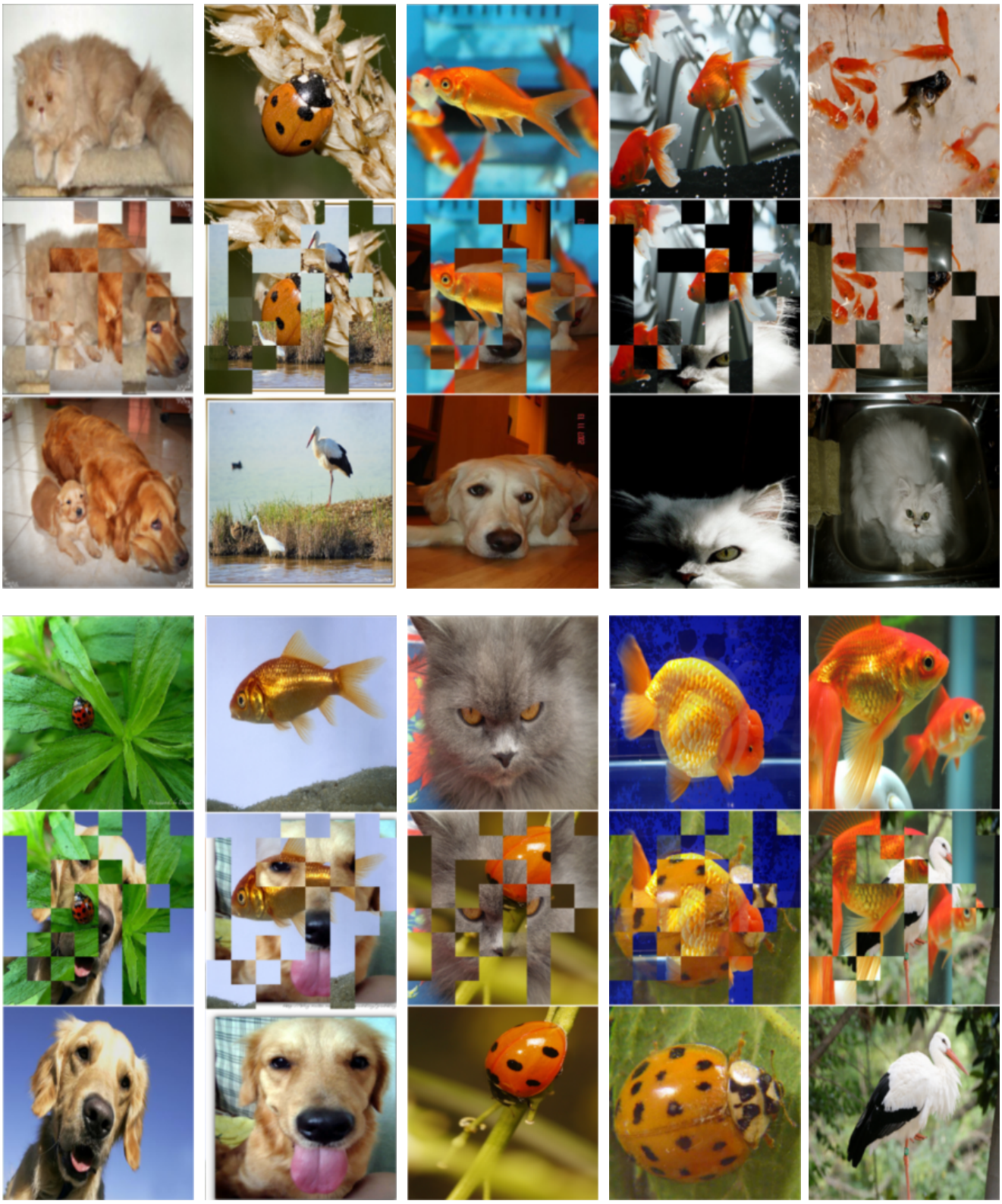

Figure A10: The mixed sample visualization of different feature layers of the ImageNet dataset as input to the mix block, from top to bottom, represents the input image a, the mix visualization from $layer3$ and the input image b, respectively.