A Appendix

A.1 More Implementation Details

Dataset information. We briefly introduce image datasets used in Section 4. (1) Small scale classification benchmarks: CIFAR-10/100 [23] contains 50,000 training images and 10,000 test images in 32×32 resolutions, with 10 and 100 classes settings. (2) Large scale classification benchmarks: ImageNet-1k (IN-1k) [24] contrains 1,281,167 training images and 50,000 validation images of 1000 classes. Tiny-ImageNet (Tiny) [5] is a rescale version of ImageNet-1k, which has 10,000 training images and 10,000 validation images of 200 classes in 64×64 resolutions. (3) Small-scale fine-grained classification scenarios: CUB-200-2011 (CUB) [51] contains 11,788 images from 200 wild bird species for fine-grained classification. FGVC-Aircraft (Aircraft) [34] contains 10,000 images of 100 classes of aircrafts. (4) Large-scale fine-grained classification datasets iNaturalist2017/2018 (iNat2017/2018) [20] contains a total of 5,089 categories with 579,184 training images and 95,986 validation images. (5) Scenic classification dataset Place205 [66] contains around 2,500,000 images from 205 common scene categories. Notice that we use modified structures [17] of ResNet and ResNeXt for CIFAR-10/100 and Tiny experiments, *i.e.*, replacing the 7×7 convolution and MaxPooling by a 3×3 convolution, while using normal structures.

Training settings. Detailed training settings of PyTorch [35], DeiT [47], and RSB A2/A3 [54] on ImageNet-1k are provided in Table 10. Notice that we replace the step learning rate decay by Cosine Scheduler [32] and remove `ColorJitter` and `PCA lighting` in PyTorch training setting for better performances.

Reproduction settings. We adopt OpenMixup[3] implemented in PyTorch [35] as the open-source codebase, where we implement AutoMix and reproduce most comparison methods (Mixup [64], CutMix [61], ManifoldMix [50], PuzzleMix [22], SaliencyMix [48], FMix [15], and ResizeMix [37]). Notice that *optimization-based* methods adopt a consistent α for all datasets, PuzzleMix adopts $\alpha = 1$, Co-Mixup and AutoMix adopts $\alpha = 2$. *Hand-crafted* methods use dataset-specific hyper-parameter settings as follows: For CIFAR-10/100, Mixup and ResizeMix use $\alpha = 1$, and CutMix, FMix and SaliencyMix use $\alpha = 0.2$, and ManifoldMix uses $\alpha = 2$, respectively. For Tiny-ImageNet, ImageNet-1k, iNat2017/2018, and Place205 using PyTorch-style training setting, ManifoldMix uses $\alpha = 0.2$ and the rest methods adopt $\alpha = 1$ for median and large backbones (*e.g.*, ResNet-50), while all these methods use $\alpha = 0.2$ for ResNet-18. For ImageNet-1k using DeiT and RSB A2/A3 settings, all these methods use $\alpha = 0.2$. For small-scale fine-grained datasets (CUB-200 and Aircraft), SaliencyMix and FMix use $\alpha = 0.2$, and ManifoldMix uses $\alpha = 0.5$, while the rest use $\alpha = 1$. As for other methods, we reproduce results of AugMix [19], Co-Mixup [21], and SuperMix [8] with their official implementations.

[3] https://github.com/Westlake-AI/openmixup

Table 10. Ingredients and hyper-parameters used for ImageNet-1k training settings.

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

A.2 More Experiments and Ablations

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

Table 11 and Table 12 report results on more practical training settings: RSB and DeiT denote *randomly combining Mixup and CutMix* which produces com-

Table 11. Top-1 accuracy (%)↑ on ImageNet-1k based on various ConvNets using RSB A2/A3 training settings.

	R-50	EfficientNet B0		MobileNet.V2	
Methods	A3	A2	A3	A2	A3
RSB	78.08	77.26	74.02	72.87	69.86
MixUp	77.66	77.19	73.87	72.78	70.17
CutMix	77.62	77.24	73.46	72.23	69.62
ManifoldMix	77.78	77.22	73.83	72.34	70.05
SaliencyMix	77.93	77.67	73.42	72.07	69.69
FMix*	77.76	77.33	73.71	72.79	70.10
PuzzleMix	78.02	77.35	74.10	72.85	70.04
ResizeMix*	77.85	77.27	73.67	72.50	69.94
AutoMix	**78.44**	**77.58**	**74.61**	**73.19**	**71.16**
Gain	**+0.36**	**+0.23**	**+0.51**	**+0.32**	**+0.99**

Table 12. Top-1 accuracy (%)↑ on ImageNet-1k based on ViTs and ConvNeXt using DeiT training settings.

Methods	DeiT-S	Swin-T	ConvNeXt-T
DeiT	79.80	81.28	82.10
MixUp	79.65	80.71	80.71
CutMix	79.78	80.83	81.38
AttentiveMix	77.63	77.27	78.19
SaliencyMix	79.32	80.68	80.94
FMix*	79.41	80.37	80.17
PuzzleMix	79.84	81.03	81.48
ResizeMix*	79.93	80.94	81.42
TransMix†	80.70	**81.80**	-
AutoMix	**80.78**	**81.80**	**82.28**
Gain	**+0.08**	+0.00	**+0.18**

petitive performs as previous state-of-the-art methods (*e.g.*, PuzzleMix), while AutoMix still brings significantly gains over the original RSB (+0.32∼1.30%) and DeiT (+0.18∼0.98%). It worth notice that previous mixup variants yield little performance gain when training with light ConvNets (*e.g.*, R-18 in Table 2, EfficientNet B0 and MobileNet.V2 (1×) in Table 11), while AutoMix achieves stable performance gains on these backbones. Moreover, AutoMix achieves competitive performances than the recently proposed Transformer-based mixup method, TransMix.

Hyperparameters for AutoMix. We further analyze the hyper-parameter setting for AutoMix with extra ablation studies conducted on Tiny-ImageNet and ImageNet-1k with various network architectures. As the same conclusion we provided in main body of experiment, the result in Figure 10 also recommends the choice of $l = 3$, which reflects the hyper-parameter robustness of AutoMix.

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

Backbone	R-18				RX-50			
Epoch	200ep	400ep	800ep	1200ep	200ep	400ep	800ep	1200ep
Vanilla	94.87	95.10	95.50	95.59	95.92	95.81	96.23	96.26
MixUp	95.70	96.55	96.62	96.84	96.88	97.19	97.30	97.33
CutMix	96.11	96.13	96.68	96.56	96.78	96.54	96.60	96.35
ManifoldMix	96.04	96.57	96.71	97.02	96.97	97.39	97.33	97.36
SaliencyMix	96.05	96.42	96.20	96.18	96.65	96.89	96.70	96.60
FMix*	96.17	96.53	96.18	96.01	96.72	96.76	96.76	96.10
PuzzleMix	96.42	96.87	97.10	97.03	97.05	97.24	97.27	97.34
ResizeMix*	96.16	96.91	96.76	97.04	97.02	97.38	97.21	97.36
AutoMix	**96.59**	**97.08**	**97.34**	**97.30**	**97.19**	**97.42**	**97.65**	**97.51**
Gain	+0.17	+0.17	+0.24	+0.26	+0.14	+0.04	+0.32	+0.15

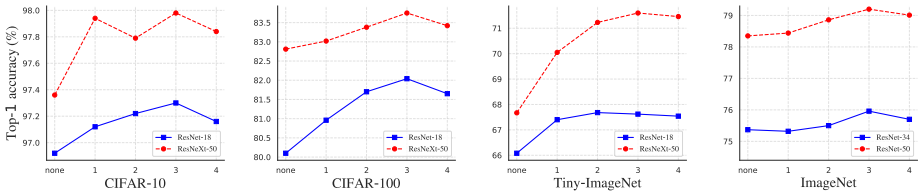

Fig. 10. Top-1 accuracy ablation study on feature layer l.

Table 14. Top-1 accuracy (%)↑ on CIFAR-100 based on ResNet-18 and ResNeXt-50 (32x4d) trained with various epochs.

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

A.3 Architecture of AutoMix

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

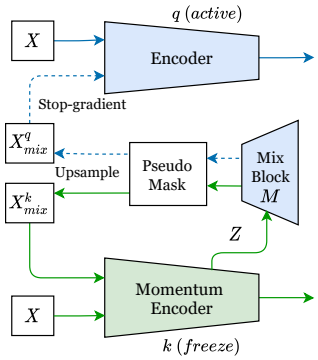

Fig. 11. The network architecture of AutoMix. The parameters in blue modules (active) are updated by backpropagation while the green (freeze) using momentum update in Equation 12.

A.4 Algorithm of AutoMix

We provide the pseudo code of AutoMix in Pytorch style:

Algorithm 1 Pseudocode AutoMix in Pytorch style.

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

f_k.params = f_q.params # initialize
for x, y in loader: # load a minibatch

    # two different permutations of data pairs
    x_q, x_k = x[index_q],x[index_k]
    y_q, y_k = y[index_q],y[index_k]
    lat_f = f_k(x) # hidden representation and logits: NxCxWxH

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

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

    loss.backward()
    update(f_q.params, M.params) # SGD update (q and M)
    f_k_params = m*f_k+(1-m)*f_q.params # momentum update
```

Fig. 12. Visualization of mixed samples on ImageNet-1k. The upper part presents the plot of mixed samples from AutoMix ($l = 3$) for $\lambda = 0.5$; the lower shows the mixed samples when different λ values are taken.

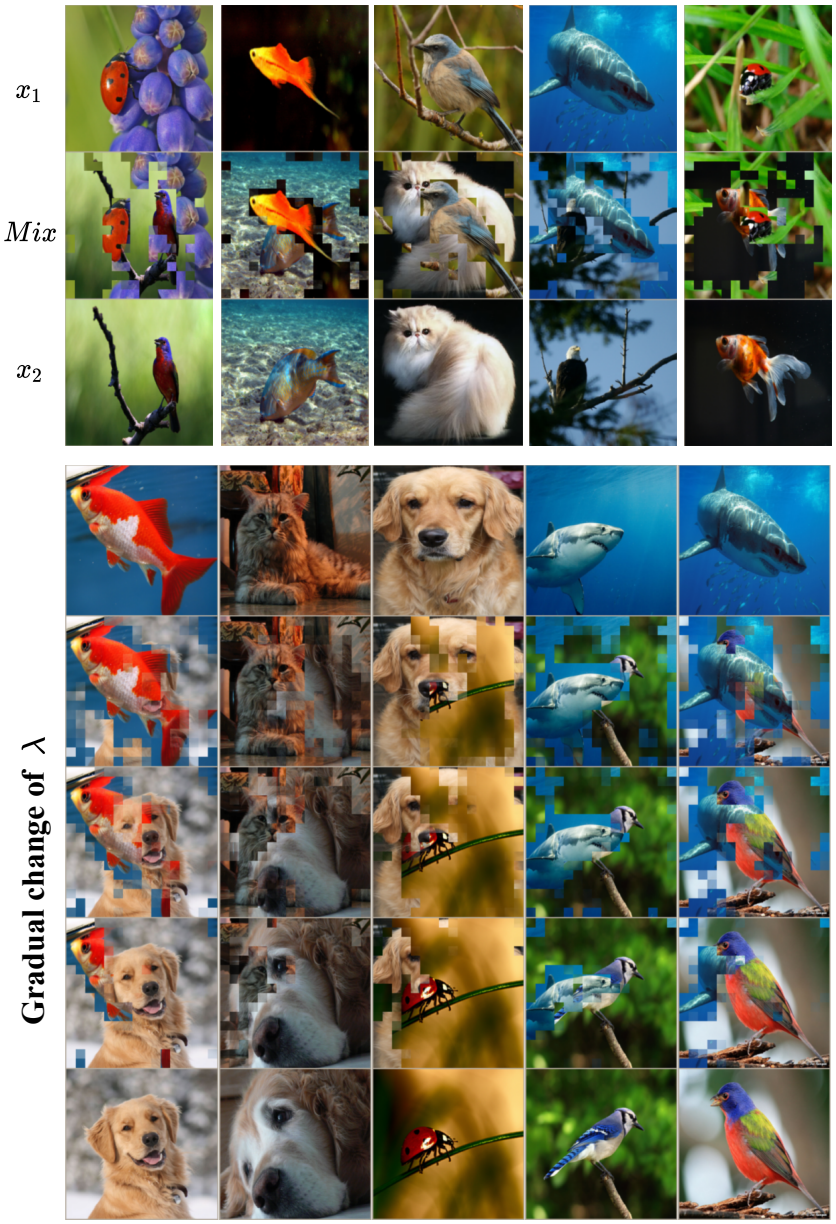

Fig. 13. Visualization of mixed samples on ImageNet-1k. The upper part presents the plot of mixed samples from AutoMix ($l = 3$) for $\lambda = 0.5$; the lower offers the mixed samples when different λ values are taken.