Mixup without Hesitation

Hao Yu, Huanyu Wang, and Jianxin Wu(✉)

State Key Laboratory for Novel Software Technology, Nanjing University, China
wujx2001@gmail.com

Abstract. Mixup linearly interpolates pairs of examples to form new samples, which has been shown to be effective in image classification tasks. However, there are two drawbacks in mixup: one is that more training epochs are needed to obtain a well-trained model; the other is that mixup requires tuning a hyper-parameter to gain appropriate capacity. In this paper, we find that mixup constantly explores the representation space, and inspired by the exploration-exploitation dilemma, we propose mixup Without hesitation (mWh), a concise and effective training algorithm. We show that mWh strikes a good balance between exploration and exploitation by gradually replacing mixup with basic data augmentation. It can achieve a strong baseline with less training time than original mixup and without searching for optimal hyper-parameter, i.e., mWh acts as mixup without hesitation.

Keywords: Deep Learning · Mixup · Exploration-Exploitation Dilemma.

1 Introduction

Deep learning has made great breakthroughs in various computer vision problems such as image classification [4] and object detection [11]. However, requiring lots of training data is the well-known drawback of deep learning, and data augmentation methods have partially alleviated this difficulty.

In particular, mixup, proposed by Zhang et al. [18], is based on virtual examples created by linearly interpolating two random samples and their corresponding labels, i.e.,

$$\begin{aligned} \tilde{x} &= \lambda x_i + (1-\lambda) x_j \\ \tilde{y} &= \lambda y_i + (1-\lambda) y_j \end{aligned} \tag{1}$$

where x_i and x_j are two data samples, and y_i and y_j are their labels. The mixing coefficient λ is sampled from a beta distribution $\mathbf{Beta}(\alpha, \alpha)$. Mixup allows deep learning models to train on a large number of virtual examples resulting from the random combination, and the standard cross-entropy loss is calculated on the soft-labels $\tilde{y}$ instead of hard labels. In general, mixup raises the generalization of deep neural networks significantly, especially on small datasets [13].

However, it is also well-known that mixup suffers from slow convergence and requires a sophisticated selection of the hyper-parameter α. In detail,

- Mixup requires more epochs to converge. Since it explores more regions of the data space, longer training is required, e.g., it takes 200 epochs to train ResNet-50 on ImageNet with mixup, but a normal training routine of 90 epochs is sufficient [18]. Note that this observation not only exists in mixup, but can also be found in other data augmentation methods that strongly increase the complexity of training data [2,17].
- Mixup requires an α value to sample mixing coefficients. Different α values usually lead to big differences in model accuracy. Zhang et al. [18] mentioned that mixup improves performance better when $\alpha \in [0.1, 0.4]$, and larger α may cause underfitting. However, α greater than 1 tends to perform better in some cases [13]. In other words, the generalization ability of mixup is heavily affected by hyper-parameter selection, but choosing a suitable α is quite difficult.

In order to solve both difficulties, we propose mWh, which stands for **m**ixup **W**ithout **h**esitation (mWh). Instead of using mixup to augment data throughout the entire model training process, mWh accelerates mixup by periodically turning the mixing operation off, which also makes it robust to the hyper-parameter α. The contributions of mWh are:

- Through carefully designed experiments and observations, we show that mixup attains its accuracy improvement through boldly *exploring* the representation space. Basic data augmentation (e.g., flipping and cropping) focuses more on *exploiting* the space. Hence, mWh strikes a good exploration-exploitation trade-off, and achieves both high accuracy and training speed.
- We gain new benchmarks of image classification consistently. Regardless of whether epochs are doubled or not, mWh performs better than mixup.
- mWh is robust with respect to α. With a default α value, mWh performs consistently well in a variety of computer vision tasks.

2 Related Work

First, we briefly review data augmentation methods and the related works that inspired this paper.

One of the important problems in computer vision is training with a small amount of data, as deep learning models often overfit with small datasets. Data augmentation is a family of techniques to solve this difficulty, and basic data augmentation methods, such as horizontal reflection, rotation and rescaling, have been widely applied to many tasks and often boost the model accuracy. Mixup can be regarded as a kind of data augmentation method and it often enhances the generalization performance of CNNs. Mixup can also ease the over-confident prediction problem for deep neural networks [13]. Similar interpolation can be applied in semi-supervised learning [1] and model adversarial robustness [10] and domain adaptation [16]. Manifold mixup [14] shares similarities with mixup. It trains neural networks on linear combinations of hidden representations of training examples.

Apart from mixup, some novel data augmentation methods have recently been proposed, too. Some data augmentation approaches are based on searching, like AutoAugment [3] and Fast AutoAugment [9]. AutoAugment designs a search space to combine various data augmentation strategies to form a policy, but the whole search algorithm is very computationally demanding. Cutout [5] randomly masks out square regions of an input image during training. GridMask [2] improves the existing information dropping algorithms. Similar to mixup, CutMix [17] also involves two training samples: it cuts one image patch and pastes it to another training image. He et al. [6] analyze the distribution gap between clean and augmented data. They preserve the standard data augmentation and refine the model with 50 epochs after training DNNs with mixup.

Note that these elaborate methods change the original images significantly, and almost always elongate the training process, e.g., in Manifold mixup, Verma et al. [14] trained PreAct ResNet-18 for 2000 epochs in CIFAR-10 and CIFAR-100, but 100 epochs of training will be enough without Manifold mixup. CutMix and GridMask also need careful hyper-parameter selection. Although they often achieve higher accuracy than basic data augmentation, more training epochs lead to significantly inflated financial and environmental costs, which is the common and significant drawback of mixup and these methods. Hence, we propose mWh to solve this dilemma. Its goal is to achieve higher accuracy even without many epochs or hyper-parameter selection.

3 mixup Without hesitation (mWh)

We propose mWh in this section. First, we analyze the effect of mixup training and reveal its property. Then we propose mixup Without hesitation (mWh), a simple plug-and-play training strategy. Finally, we study the role of every stage as well as the influence of hyper-parameters in mWh.

3.1 Observations

We investigate the role mixup plays during training, and demonstrate that with *the combination of mixup and basic data augmentation*, mWh has the potential to retain the accuracy improvement brought by mixup, too.

Training neural networks involves finding minima of a high-dimensional non-convex loss function, which can be visualized as an energy landscape. Since mixup generates more uncertain data, it enlarges the sample representation space significantly, and *explores* more potential energy landscapes, so the optimization method can find a better locally optimal solution. However, the side effect of mixup is that it also brings instability during the training process. We trained PreAct ResNet-18 on CIFAR-10, and as Fig. 1 showed, we have two observations from the curves. The first one is: with mixup, the loss oscillates constantly on the original test dataset, but if the model is trained on the clean data, the curves are smoother. This phenomenon suggests that compared with basic data augmentation, mixup introduces higher uncertainty in the training process. The

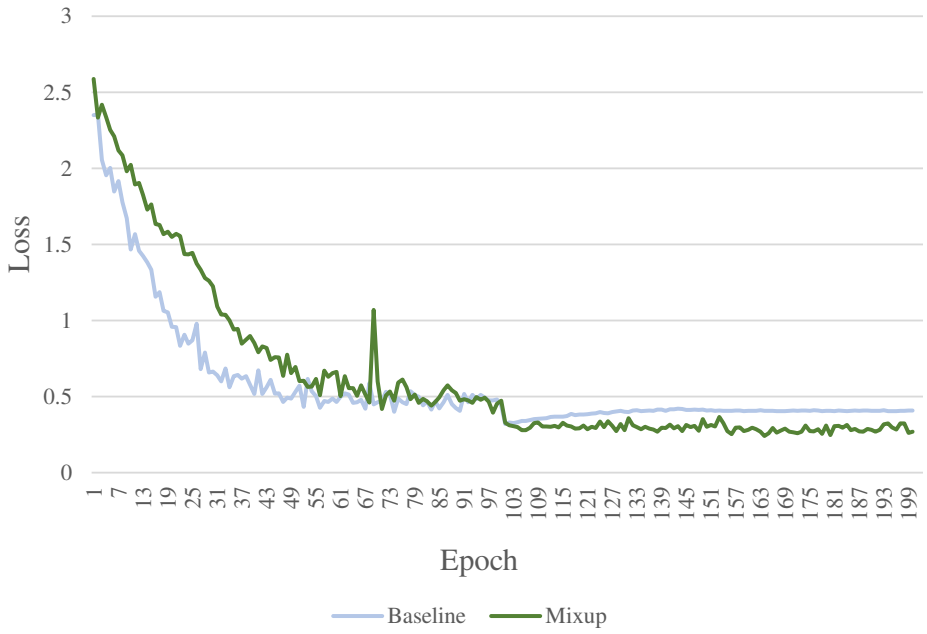


Fig. 1. Cross-entropy loss on the CIFAR-10 *test* set. We used PreAct ResNet-18 and $\alpha = 0.5$.

second one is: in the early training stage, mixup enlarges the test loss, which indicates it focuses on exploring the energy landscapes and will not fall into local optima prematurely.

However, in the later training stages, since the model gets closer to convergence, exploration must not be the main goal. Instead, we should switch to *exploiting* the current state using *basic data augmentation*. Results in Table 1 validate our motivations. We train PreAct ResNet-18 on CIFAR-100 with 100 epochs and set α as 1.0. The result demonstrates that using mixup only in the first 50 epochs is better. Hence, we conjecture that *mixup is effective because it actively explores the search space in the early epochs, while in the later epochs, mixup might be harmful*.

But, if we directly apply basic data augmentation after a model is trained with mixup [6], this refinement operation may end up with overfitting. We trained PreAct ResNet-18 with 200 epochs in CIFAR-10 and Tiny-ImageNet-200 with mixup, and refine the model with 25 epochs without mixup. Learning rate starts at 0.1 and is divided by 10 after 100 and 150 epochs, and we set the learning rate to be the same as that in the final epochs of the last stage during refinement. The results are shown in Table 2. We can observe that accuracy decreased after refinement, which indicates the number of refining epochs is difficult to control, and refining may lead to overfitting. Putting our observations and conjectures together, we propose to *gradually replace mixup with basic data aug-*

Table 1. Accuracy (%) on CIFAR-100. Baseline means we train the model with basic data augmentation. Mixup means we apply mixup throughout the training process. First Half Mixup means the first half of epochs apply mixup but the last do not, and similarly, Second Half Mixup means we only apply mixup in the second half of epochs.

Methods	Top1	Top5
Baseline	74.20	92.53
Mixup	75.25	92.40
First Half Mixup	**75.87**	**93.10**
Second Half Mixup	72.50	91.04

Table 2. Accuracy (%) on PreAct ResNet-18, $\alpha = 0.5$.

Datasets	Mixup	+Refinement
CIFAR-10	**95.46**	95.30
Tiny-ImageNet-200	**60.73**	60.21

mentation such that the learning algorithm gradually switches from exploration to exploitation, which is a good strategy to solve the *exploration-exploitation dilemma* [12] in our context.

3.2 Algorithm

The mWh algorithm is in Algorithm 1. We use a mini-batch instead of an epoch as the unit of execution. We denote the total number of mini-batches as m, and use two hyper-parameters p and q ($0 \leq p < q \leq 1$) to divide the whole training process into three stages, i.e.,

- First stage: from 1 to pm mini-batches, we train with mixup. Note that here we assume pm is an integer.
- Second stage: from $pm + 1$ to qm mini-batches, we alternate between mixup and basic data augmentation. If the last mini-batch does not apply mixup, the next one will, and vice versa.
- Third stage: from $qm + 1$ mini-batches to the end, we run mixup with probability ϵ, where ϵ decreases *linearly* from 1 to 0.

In the first stage, mWh lets the model explore a large portion of the sample representation space by consistently applying mixup.

The second stage is an exploration-exploitation trade-off. We periodically turn mixup on and off to avoid getting trapped in a local optimum prematurely. When we turn mixup off, the model will exploit the limited and promising region of the sample representation space with the hope of accelerating convergences. When we turn mixup on, the model will keep exploring more energy landscapes.

In the third stage, we gradually switch to exploitation, which is inspired by the ϵ-greedy algorithm [12]. We define an exploration rate ϵ that is initially set

Algorithm 1: The mWh Training Algorithm

Input: Training dataset $(\mathcal{X}, \mathcal{Y})$, number of training mini-batches m, two parameters p and q satisfying $(0 \leq p < q \leq 1)$, Beta distribution parameter α for mixup.

for $i = 1$ *to* m **do**
  Draw a mini-batch (x_b, y_b).
  if $i \leq pm$ **then** *// First stage*
    $(\tilde{x}_b,\ \tilde{y}_b) = \text{mixup}(x_b, y_b, \alpha)$
  else if $i \leq qm$ **then** *// Second stage*
    if i is even **then**
      $(\tilde{x}_b,\ \tilde{y}_b) = \text{mixup}(x_b, y_b, \alpha)$
    else
      $(\tilde{x}_b,\ \tilde{y}_b) = \text{basic\_augmentation}(x_b, y_b)$
    end if
  else *// Third stage*
    $\epsilon = \frac{m-i}{m(1-q)}$
    Randomly generate threshold $\theta \in [0, 1]$.
    if $\theta < \epsilon$ **then**
      $(\tilde{x}_b,\ \tilde{y}_b) = \text{mixup}(x_b, y_b, \alpha)$
    else
      $(\tilde{x}_b,\ \tilde{y}_b) = \text{basic\_augmentation}(x_b, y_b)$
    end if
  end if
  Train model with mini-batch $(\tilde{x}_b,\ \tilde{y}_b)$.
end for

to 1. This rate is the probability that our model will use mixup. As ϵ decreases gradually, the model tends to choose exploitation rather than exploration.

Finding suitable values of p and q is essential. Note that we expect mWh to be robust and insensitive to hyper-parameters. Hence, we want to fix p and q in *all* experiments. Here we train ResNet-50 on ImageNet with 100 epochs and study the effect of different p and q. In these experiments, the default learning rate is 0.1 with a linear warmup for the first 5 epochs and divided by 10 after training 30, 60, 90 epochs. We set batch size to 256. In Table 3, we set q as 0.9 and explore the impact of different q. We also fix p as 0.6 and research the effect of q in Table 4. Especially, when q is equal to 1.0, we remove the third stage in mWh, and similarly, when q is 0.6, we apply the ϵ-greedy algorithm after the mini-batches of 60 percent. Based on our experimental results, although choosing different p and q does not have a significant effect on the outcome, 0.6 and 0.9 are a reasonable choice, so we always set $p = 0.6$ and $q = 0.9$.

Now we validate our framework by an ablation study on ImageNet. We train ResNet-50 on ImageNet with 100 epochs and set α as 0.5. All experiments apply mixup in the top 60 percent mini-batches. Table 5 contains several results and we try different strategies in Stage 2 and Stage 3. Different rows represent using different strategies to train the model. In particular, none indicates we apply

Table 3. The influence of p on ImageNet.

p	0.5	0.6	0.7	0.8	0.9
$\alpha = 0.2$	**76.952**	76.948	76.814	76.856	76.894
$\alpha = 0.5$	76.782	76.854	**76.964**	76.754	76.736

Table 4. The influence of q on ImageNet.

q	0.6	0.7	0.8	0.9	1.0
$\alpha = 0.2$	76.854	76.814	76.792	**76.948**	76.742
$\alpha = 0.5$	76.792	**76.942**	76.894	76.854	76.716

Table 5. Accuracy (%) of ResNet-50 trained on ImageNet.

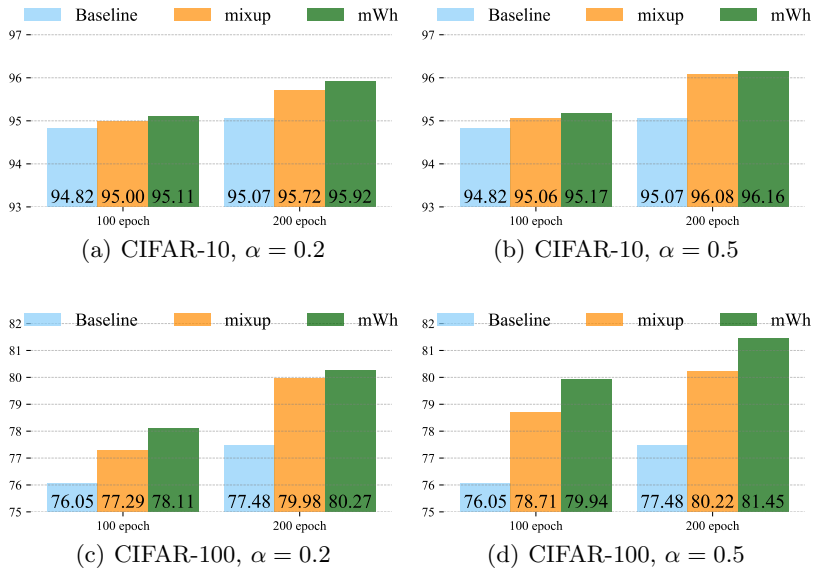
Stage 2	Stage 3	Top1	Top5
None	None	76.756	93.314
mixup	None	76.770	93.478
mixup	mixup	76.212	93.246
mixup	mWh	76.832	**93.504**
mWh	None	76.772	93.428
mWh	mixup	76.388	93.304
mWh	mWh	**76.854**	93.463

basic data augmentation. When we apply mWh at Stage 2, it refers to the alternating of mixup and basic data augmentation between the mini-batches of 60 to 90 percent. Using mWh at Stage 3 means running mixup with a probability of ϵ in the final 10 percent of the mini-batches.

From Table 5, we can find no matter at what stage, applying mWh instead of mixup leads to higher accuracy. Especially, mixup reduces the performance of the model in Stage 3, which coincides well with results in Table 1 and our conjecture (mixup is harmful in later epochs). These results verify the effectiveness of our framework. Note that those results do not mean that our 3 stages setting is the best, maybe four stages or cosine decaying chance strategy performs better in some experimental results. After all, our contribution is a working algorithm that meets our goal (attain mixup accuracy with fewer epochs and insensitive to alpha).

4 Experiments

In this section, we evaluate the performance of mWh. We first conduct experiments to validate the effectiveness of mWh on four benchmark classification datasets. For a fair comparison, mWh used the same random seed as mixup. Then we show its transferability in CutMix [17]. The parameters p and q are always set to 0.6 and 0.9. All our experiments are conducted by PyTorch.



(a) CIFAR-10, $\alpha = 0.2$ (b) CIFAR-10, $\alpha = 0.5$

(c) CIFAR-100, $\alpha = 0.2$ (d) CIFAR-100, $\alpha = 0.5$

Fig. 2. Accuracy (%) on CIFAR-10, CIFAR-100 using mixup and mWh.

4.1 Experiments on Image Classification Tasks

First we will show the results on four small-scale datasets, i.e., CIFAR-10, CIFAR-100 [8] and CUB-200 [15]. Then the results on ImageNet will be presented.

We want to show that without doubled epochs and tuning hyper-parameters we can still get better performance in all datasets, so we follow the setting of Zhang et al. [18] but halve the training epochs. We also set α to 0.2 and 0.5 to illustrate mWh is robust to hyper-parameter selection. To further show the validity of mWh, we double the epochs and report the results, although it is not our primary focus.

Datasets: CIFAR-10 and CIFAR-100 [8] both consist of 50k training and 10k test images at 32x32 resolution. CIFAR-10 contains 5k training and 1k test images per class in a total of 10 classes, and CIFAR-100 has 100 classes containing 600 images each. For the CUB-200, it contains 200 categories of birds with 5,994 training and 5,794 testing images. The large-scale ImageNet ILSVRC-12 dataset consists of 1.28M training and 50K validation images of various resolutions.

Implementation details: For CIFAR-10 and CIFAR-100, to provide a strong baseline we train DenseNet-121 [7] with the mini-batch size of 128 for 100 epochs. Learning rate starts at 0.1 and is divided by 10 after 50 and 75 epochs. Note that we also conduct experiments to double the epoch, i.e., we train the model with 200 epochs, and divide the learning rate by 10 after 100 and 150 epochs.

Table 6. Accuracy (%) on CUB-200. The first two groups of experiments are about training ResNet-18 from scratch and the rest are the fine-tuning experiments.

Method	$\alpha = 0.2$		$\alpha = 0.5$	
	Epochs	Accuracy	Epochs	Accuracy
Baseline	350	64.308	350	64.308
mixup	350	66.672	350	68.347
mWh	350	**67.535**	350	**70.297**
Baseline	175	62.858	175	62.858
mixup	175	**63.704**	175	64.118
mWh	175	**63.704**	175	**65.948**
Baseline	300	77.080	300	77.080
mixup	300	78.650	300	78.391
mWh	300	**79.272**	300	**79.185**
Baseline	150	76.345	150	76.345
mixup	150	78.236	150	78.219
mWh	150	**78.357**	150	**78.840**

For CUB-200, we use ResNet-18 and crop 224*224 patches as input images for training. For fair comparisons, we evaluate the strategy of training from scratch and fining tune. For training from scratch, we set the number of training epochs to be 175 and 300, and initialize the learning rate as 0.1, batch size as 32. A smoother cosine learning rate adjustment is applied. For fine-tuning, we train the model with 150 and 300 epochs. We set the learning rate as 0.001, batch size as 32. We also use a cosine schedule to scale the learning rate. The initialization ResNet-18 model is downloaded from the PyTorch official website.

To provide further evidence about the quality of representations learned with mWh, we evaluate it on ImageNet. We train ResNet-50 from scratch. For faster convergence we use NVIDIA's mixed-precision training code base with batch size 2048. The default learning rate is $0.1 * \frac{\text{batch size}}{256}$ with a linear warmup for the first 5 epochs and divided by 10 after training 30, 60, 90 epochs when training 100 epochs, or after 60, 120 and 180 epochs when training 200 epochs. We first randomly crop a patch from the original image and then resize the patch to the target size (224*224). Finally, the patch is horizontally flipped with a probability of 0.5.

Results: For CIFAR-10 and CIFAR-100, we summarize the results in Fig. 2. mWh consistently outperforms mixup and the baseline. And, mWh with 100 epochs consistently outperforms baseline with even 200 epochs. Note that on CIFAR-100, with a higher α, mWh boosts more accuracy. We think the reason is that the difference between the augmented data and the original data will be greater because of a higher α, so the empirical risk will decrease more after introducing basic data augmentation. This indicates mWh will bring more improvement with larger α, and this situation is particularly noticeable on more complex datasets.

Table 7. Accuracy (%) on ImageNet with ResNet-50.

Method	$\alpha = 0.2$		$\alpha = 0.5$	
	Epochs	Accuracy	Epochs	Accuracy
Baseline	200	76.392	200	76.392
mixup	200	**77.148**	200	77.838
mWh	200	77.098	200	**77.888**
Baseline	100	76.043	100	76.043
mixup	100	76.718	100	76.212
mWh	100	**76.948**	100	**76.854**

Table 6 shows the results on CUB-200, and similar to the previous experiments, we observe mWh is highly competitive when compared with mixup. When jointly applying mixup and basic data augmentation, mWh obtains the lowest Top-1 error in both training from scratch and fining tune.

Table 7 demonstrates the validation accuracy rates on the ImageNet dataset. mWh outperforms (or is on par with) mixup, mWh also exhibits its robustness to the hyper-parameter α. In the 100 epochs case, although mixup is effective when $\alpha = 0.2$, it is not effective when $\alpha = 0.5$ (only improves 0.169% over the baseline). However, mWh is consistently effective for different α values.

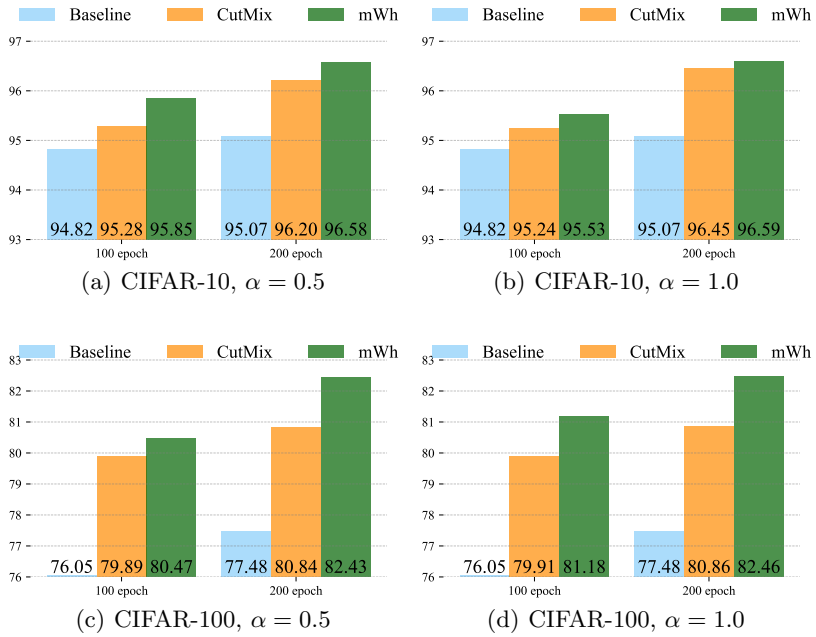
As the results have shown, mWh achieves the state-of-the-art performance when halving the epochs, and without deliberately selecting α we can still gain consistently better performance than mixup. Although mWh's goal is not to gain improvement with more epochs, our results show that mWh performs better than mixup when training time is doubled. These results indicate that without doubling epochs and selecting optimal α, mWh can still perform very well. The fact that with more epochs mWh performs better than mixup is a nice byproduct.

4.2 Transferability in CutMix

In order to provide insights into what makes mWh successful, we further study the transferability of mWh in CutMix. We examine the effect of mWh in CutMix for the image classification tasks.

Implementation details: All of our following experiments share the same setting with previous ones. We train our strategy in CIFAR-10 and CIFAR-100. Note that different from mixup, we select α as 1 instead of 0.2. It is because according to the recommendation of Yun et al. [17], choosing α as 1.0 will achieve better performance. For providing strong baselines we set α to 1.0 and 0.5.

Results: The results of CutMix with mWh in CIFAR-10 and CIFAR-100 are shown in Fig. 3. mWh also brings considerable improvement, which proves the validity of our algorithm.



(a) CIFAR-10, $\alpha = 0.5$ (b) CIFAR-10, $\alpha = 1.0$

(c) CIFAR-100, $\alpha = 0.5$ (d) CIFAR-100, $\alpha = 1.0$

Fig. 3. Accuracy (%) on CIFAR-10, CIFAR-100 using CutMix and mWh in CutMix.

5 Discussion and Conclusion

In this paper, we proposed mixup Without hesitation (mWh), a simple but general training policy for effective training. We apply the strategy of reintroducing basic data augmentation to balance exploration and exploitation. Experimental results showed that mWh improves the convergence rate of various dataset instances and is robust to the hyper-parameter selection. It also gains remarkable improvement in different tasks and models compared with the baseline.

Many data augmentation algorithms used in computer vision have similar features as mixup. Therefore, One interesting future work is to extend the proposed algorithm to other augmentation algorithms.

Acknowledgements. This research was partially supported by the National Natural Science Foundation of China (61772256).

References

1. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.A.: MixMatch: A Holistic Approach to Semi-Supervised Learning. In: Advances in Neural Information Processing Systems 32, pp. 5049–5059 (2019)

2. Chen, P.: GridMask Data Augmentation. arXiv preprint arXiv:2001.04086 (2020)
3. Cubuk, E.D., Zoph, B., Mane, D., Vasudevan, V., Le, Q.V.: AutoAugment: Learning Augmentation Strategies from Data. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 113–123 (2019)
4. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 248–255 (2009)
5. DeVries, T., Taylor, G.W.: Improved Regularization of Convolutional Neural Networks with Cutout. arXiv preprint arXiv:1708.04552 (2017)
6. He, Z., Xie, L., Chen, X., Zhang, Y., Wang, Y., Tian, Q.: Data Augmentation Revisited: Rethinking the Distribution Gap between Clean and Augmented Data. arXiv preprint arXiv:1909.09148 (2019)
7. Huang, G., Liu, Z., Van Der Maaten, L., Weinberger, K.Q.: Densely Connected Convolutional Networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). pp. 4700–4708 (2017)
8. Krizhevsky, A., Hinton, G.: Learning Multiple Layers of Features from Tiny Images. Tech. rep., University of Toronto (2009)
9. Lim, S., Kim, I., Kim, T., Kim, C., Kim, S.: Fast AutoAugment. In: Advances in Neural Information Processing Systems 32. pp. 6662–6672 (2019)
10. Pang, T., Xu, K., Zhu, J.: Mixup inference: Better exploiting mixup to defend adversarial attacks. In: International Conference on Learning Representations, ICLR (2019)
11. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: Advances in Neural Information Processing Systems 28, pp. 91–99 (2015)
12. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction, 2nd edition. MIT press (2018)
13. Thulasidasan, S., Chennupati, G., Bilmes, J.A., Bhattacharya, T., Michalak, S.: On Mixup Training: Improved Calibration and Predictive Uncertainty for Deep Neural Networks. In: Advances in Neural Information Processing Systems 32. pp. 13888–13899 (2019)
14. Verma, V., Lamb, A., Beckham, C., Najafi, A., Mitliagkas, I., Courville, A., Lopez-Paz, D., Bengio, Y.: Manifold Mixup: Better Representations by Interpolating Hidden States. In: International Conference on Machine Learning. pp. 6438–6447 (2019)
15. Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., Perona, P.: Caltech-UCSD birds 200. Tech. rep., California Institute of Technology (2010)
16. Xu, M., Zhang, J., Ni, B., Li, T., Wang, C., Tian, Q., Zhang, W.: Adversarial Domain Adaptation with Domain Mixup. In: Proceedings of the AAAI Conference on Artificial Intelligence. pp. 6502–6509 (2020)
17. 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 IEEE International Conference on Computer Vision. pp. 6023–6032 (2019)
18. Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. In: International Conference on Learning Representations, ICLR (2018)