AlignMixup: Improving Representations By Interpolating Aligned Features

Shashanka Venkataramanan, Ewa Kijak, Laurent Amsaleg, Yannis Avrithis

AlignMixup: Improving Representations By Interpolating Aligned Features

Shashanka Venkataramanan[1] Ewa Kijak[1] Laurent Amsaleg[1] Yannis Avrithis[2]
[1]Inria, Univ Rennes, CNRS, IRISA [2] Athena RC

Abstract

Mixup is a powerful data augmentation method that interpolates between two or more examples in the input or feature space and between the corresponding target labels. However, how to best interpolate images is not well defined. Recent mixup methods overlay or cut-and-paste two or more objects into one image, which needs care in selecting regions. Mixup has also been connected to autoencoders, because often autoencoders generate an image that continuously deforms into another. However, such images are typically of low quality.

In this work, we revisit mixup from the deformation *perspective and introduce AlignMixup, where we geometrically align two images in the feature space. The correspondences allow us to interpolate between two sets of features, while keeping the locations of one set. Interestingly, this retains mostly the geometry or pose of one image and the appearance or texture of the other. We also show that an autoencoder can still improve representation learning under mixup, without the classifier ever seeing decoded images. AlignMixup outperforms state-of-the-art mixup methods on five different benchmarks. Code available at* https://github.com/shashankvkt/AlignMixup_CVPR22.git

1. Introduction

Data augmentation [10, 36, 43] is a powerful regularization method that increases the amount and diversity of data, be it labeled or unlabeled [16]. It improves the generalization performance and helps learning invariance [49] at almost no cost, because the same example can be transformed in different ways over epochs. However, by operating on one image at a time and limiting to label-preserving transformations, it has limited chances of exploring beyond the image manifold. Hence, it is of little help in combating memorization of training data [67] and sensitivity to adversarial examples [53].

Mixup operates on two or more examples at a time, *interpolating* between them in the input space [69] or feature space [58], while also interpolating between target labels for image classification. This flattens class representations [58], reduces overly confident incorrect predictions, and smoothens decision boundaries far away from training data. However, input mixup images are overlays and tend to be unnatural [65]. Interestingly, recent mixup methods focus of combining two [32, 65] or more [31] objects from different images into one in the input space, making efficient use of training pixels. However, randomness in the patch selection and thereby label mixing may mislead the classifier to learn uninformative features [57], which raises the question: *what is a good interpolation of images?*

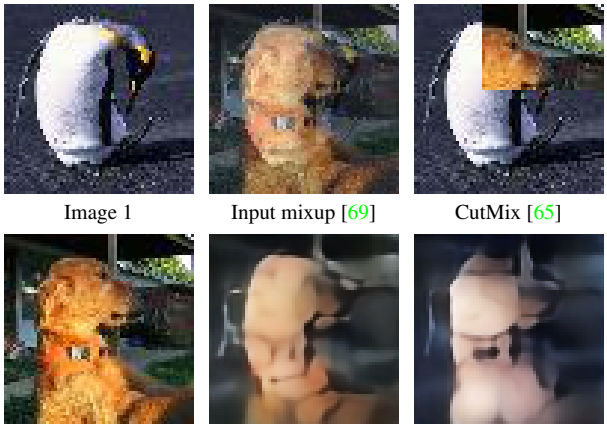

Figure 1. Different mixup methods. AlignMixup retains the pose of image 2 and the texture of image 1. This different from overlay (Input and Manifold mixup) or combination of two objects (CutMix). Manifold mixup and AlignMixup visualized by a decoder (subsection 3.3) that is not used at training.

Bengio *et al.* [3] show that traversing along the manifold of representations obtained from deeper layers of the network more likely results in finding realistic examples. This is because the interpolated points smoothly traverse the underlying manifold of the data, capturing salient characteristics of the two images. Furthermore, [4] show the ability of autoencoders to capture semantic correspondences obtained by decoding mixed latent codes. This is because the autoencoder may disentangle the underlying factors of variation. Efforts have followed on mixing latent representations of autoencoders to generate realistic images for data aug-

mentation. However, these approaches are more expensive, requiring three networks (encoder, decoder, classifier) [4] and more complex, often also requiring an adversarial discriminator [2, 39]. More importantly, they perform poorly compared to standard input mixup on large datasets [39], due to the low quality of generated images.

In this work, we are motivated by the idea of *deformation* as a natural way of interpolating images, where one image may deform into another, in a continuous way. Contrary to previous efforts, we do not interpolate directly in the input space, we do not limit to vectors as latent codes and we do not decode. We rather investigate geometric *alignment* for mixup, based on explicit semantic correspondences in the feature space. In particular, we explicitly align the feature tensors of two images, resulting in soft correspondences. The tensors can be seen as sets of features with coordinates. Hence, each feature in one set can be interpolated with few features in the other.

By choosing to keep the coordinates of one set or the other, we define an *asymmetric* operation. What we obtain is one object continuously morphing, rather than two objects in one image. Interestingly, observing this asymmetric morphing reveals that we retain the *geometry* or *pose* of the image where we keep the coordinates and the *appearance* or *texture* of the other. Figure 1 illustrates that our method, *AlignMixup*, retains the *pose* of image 2 and the *texture* of image 1, which is different from existing mixup methods. Note that, as in manifold mixup, we *do not* decode, hence we are not concerned about the quality of generated images.

We make the following contributions:

1. We introduce a novel mixup operation, called *AlignMixup*, advocating interpolation of local structure in the feature space (subsection 3.2). Feature tensors are ideal for alignment, giving rise to semantic correspondences and being of low resolution. Alignment is efficient by using *Sinkhorn distance* [11].
2. We also show that a *vanilla autoencoder* can further improve representation learning under mixup training, without the classifier seeing decoded clean or mixed images (section 4).
3. We set a new state-of-the-art on *image classification*, *robustness to adversarial attacks*, *calibration*, *weakly-supervised localization* and *out-of-distribution detection* against more sophisticated mixup operations on several networks and datasets (section 4).

2. Related Work

Mixup [69], concurrently with similar methods [30, 56], introduce *mixup*, augmenting data by linear interpolation between two examples. While [69] apply mixup on intermediate representations, it is [58] who make this work, introducing *manifold mixup*. Without alignment, the result is an overlay of either images [69] or features [58]. [23] eliminate "manifold intrusion"—mixed data conflicting with true data. Unlike manifold mixup, AlignMixup interpolates feature tensors from deeper layers after aligning them.

Nonlinear mixing over random image regions is an alternative, *e.g.* from masking square regions [14] to cutting a rectangular region from one image and pasting it onto another [65], as well as several variants using arbitrary regions [25, 52, 54]. Instead of choosing regions at random, *saliency* can be used to locate objects from different images and fit them in one [31, 32, 44, 57]. Exploiting the knowledge of a teacher network to mix images based on saliency has been proposed in [12]. Instead of combining more than one objects in an image, AlignMixup attempts to deform one object into another.

Another alternative is Automix [72], which employs a U-Net rather than an autoencoder, mixing at several layers. It is limited to small datasets and provides little improvement over manifold mixup [58]. StyleMix and StyleCutMix [28] interpolate content and style between two images, using AdaIN [29], a style transfer autoencoder network. By contrast, AlignMixup aligns feature tensors and interpolates matching features directly, without using any additional network.

Alignment Local correspondences from intra-class alignment of feature tensors have been used in *image registration* [9, 40], *optical flow* [61], *semantic alignment* [24, 46] and *image retrieval* [50]. Here, we mostly use *inter-class* alignment. In *few-shot learning*, local correspondences between query and support images are important in finding attention maps, used *e.g.* by CrossTransformers [15] and DeepEMD [68]. The *earth mover's distance* (EMD) [47], or *Wasserstein metric*, is an instance of *optimal transport* [59], addressed by linear programming. To accelerate, [11] computes optimal matching by *Sinkhorn distance* with *entropic regularization*. This distance is widely applied between distributions in generative models [18, 42].

EMD has been used for mixup in the input space, for instance *point mixup* for 3D point clouds [6] and OptTransMix for images [72], which is the closest to our work. However, aligning coordinates only applies to images with clean background. We rather *align tensors in the feature space*, which is generic. We do so using the Sinkhorn distance, which is orders of magnitude faster than EMD [11].

3. AlignMixup

3.1. Preliminaries

Problem formulation Let (x, y) be an image $x \in \mathcal{X}$ with its one-hot encoded class label $y \in Y$, where $\mathcal{X}$ is the input image space, $Y = [0, 1]^k$ and k is the number of classes. An *encoder network* $F : \mathcal{X} \to \mathbb{R}^{c \times w \times h}$ maps x to feature tensor $\mathbf{A} = F(x)$, where c is the number of channels and

$w \times h$ is the spatial resolution. A *classifier* $g : \mathbb{R}^{c\times w\times h} \to \mathbb{R}^k$ then maps $\mathbf{A}$ to the vector $p = g(\mathbf{A})$ of probabilities over classes.

Mixup We follow [58] in mixing the representations from different layers of the network, focusing on the deepest layers near the classifier. We are given two labeled images $(x, y), (x', y') \in \mathcal{X} \times Y$. We draw an *interpolation factor* $\lambda \in [0, 1]$ from $\mathrm{Beta}(\alpha, \alpha)$ [69] and then we interpolate labels y, y' linearly by the *standard* mixup operator

$$\mathrm{mix}_\lambda(y, y') := \lambda y + (1 - \lambda)y' \tag{1}$$

and inputs x, x' by the generic formula

$$\mathrm{Mix}_\lambda^{f_1,f_2}(x, x') := f_2(\mathrm{Mix}_\lambda(f_1(x), f_1(x')), \tag{2}$$

where Mix_λ is a mixup operator to be defined. This generic formula allows interpolation of the input or feature as $f_2 \circ f_1$ according to

$$\text{input } (x): \quad f_1 := \mathrm{id}, f_2 := F \tag{3}$$
$$\text{feature } (\mathbf{A}): \quad f_1 := F, f_2 := \mathrm{id}, \tag{4}$$

where id is the identity mapping. For (3), we define Mix_λ in (2) as standard mixup mix_λ (1), like [69]; while for (4), we define Mix_λ as discussed in subsection 3.2.

By default, we train the encoder network and the classifier by using a classification loss L_c on the output of the classifier g for mixed examples along with the corresponding mixed labels:

$$L_c(g(\mathrm{Mix}_\lambda^{f_1,f_2}(x, x')), \mathrm{mix}_\lambda(y, y')), \tag{5}$$

where $L_c(p, y) := -\sum_{i=1}^k y_i \log p_i$ is the standard cross-entropy loss. More options using an autoencoder architecture are investigated in section 4.

3.2. Interpolation of aligned feature tensors

Alignment Alignment refers to finding a geometric correspondence between image elements before interpolation. The feature tensor is ideal for this purpose, because its spatial resolution is low, reducing the optimization cost, and allows for semantic correspondence, because features close to the classifier are small. Importantly, we are not attempting to combine two or more objects into one image [32], but put two objects in correspondence and then interpolate into one. We make no assumptions on the structure of input images in terms of objects and we use no ground truth correspondences.

Our feature tensor alignment is based on *optimal transport* theory [59] and *Sinkhorn distance* (SD) [11] in particular. Let $\mathbf{A} := F(x), \mathbf{A}' := F(x')$ be the $c \times w \times h$ feature tensors of images $x, x' \in \mathcal{X}$. We reshape them to $c \times r$ matrices A, A' by flattening the spatial dimensions, where $r := hw$. Then, every column $a_j, a'_j \in \mathbb{R}^c$ of A, A' for $j = 1, \ldots, r$ is a feature vector representing corresponding to a spatial position in the original image x, x'. Let M be the $r \times r$ *cost matrix* with its elements being the pairwise distances of these vectors:

$$m_{ij} := \left\|a_i - a'_j\right\|^2 \tag{6}$$

for $i, j \in \{1, \ldots, r\}$. We are looking for a *transport plan*, that is, a $r \times r$ matrix $P \in U_r$, where

$$U_r := \{P \in \mathbb{R}_+^{r\times r} : P\mathbf{1} = P^\top\mathbf{1} = \mathbf{1}/r\} \tag{7}$$

and $\mathbf{1}$ is an all-ones vector in $\mathbb{R}^r$. That is, P is non-negative with row-wise and column-wise sum $1/r$, representing a joint probability over spatial positions of $\mathbf{A}, \mathbf{A}'$ with uniform marginals. It is chosen to minimize the expected pairwise distance of their features, as expressed by the linear cost function $\langle P, M\rangle$, under an entropic regularizer:

$$P^* = \arg\min_{P\in U_r} \langle P, M\rangle - \epsilon H(P), \tag{8}$$

where $H(P) := -\sum_{ij} p_{ij} \log p_{ij}$ is the entropy of P, $\langle\cdot,\cdot\rangle$ is Frobenius inner product and ϵ is a regularization coefficient. The optimal solution P^* is unique and can be found by forming the $r \times r$ *similarity* matrix $e^{-M/\epsilon}$ and then applying the Sinkhorn-Knopp algorithm [34], *i.e.*, iteratively normalizing rows and columns. A small ϵ leads to sparser P, which improves one-to-one matching but makes the optimization harder [1], while a large ϵ leads to denser P, causing more correspondences and poor matching.

Interpolation The *assignment matrix* $R := rP^*$ is a doubly stochastic $r \times r$ matrix whose element r_{ij} expresses the probability that column a_i of A corresponds to column a'_j of A'. Thus, we align A and A' as follows:

$$\widetilde{A} := A'R^\top \tag{9}$$
$$\widetilde{A}' := AR. \tag{10}$$

Here, column $\widetilde{a}_i$ of $c \times r$ matrix $\widetilde{A}$ is a convex combination of columns of A' that corresponds to the same column a_i of A. We reshape $\widetilde{A}$ back to $c \times w \times h$ tensor $\widetilde{\mathbf{A}}$ by expanding spatial dimensions and we say that $\widetilde{\mathbf{A}}$ represents $\mathbf{A}$ *aligned to* $\mathbf{A}'$. We then interpolate between $\widetilde{\mathbf{A}}$ and the original feature tensor $\mathbf{A}$:

$$\mathrm{mix}_\lambda(\mathbf{A}, \widetilde{\mathbf{A}}). \tag{11}$$

As shown in Figure 2 (toy example, top right), $\widetilde{\mathbf{A}}$ is geometrically close to $\mathbf{A}$. The correspondence with $\mathbf{A}'$ and the geometric proximity to $\mathbf{A}$ makes $\widetilde{\mathbf{A}}$ appropriate for interpolation with $\mathbf{A}$. Symmetrically, we can also *align* $\mathbf{A}'$ *to* $\mathbf{A}$ and interpolate between $\widetilde{\mathbf{A}}'$ and $\mathbf{A}'$:

$$\mathrm{mix}_\lambda(\mathbf{A}', \widetilde{\mathbf{A}}'). \tag{12}$$

When mixing feature tensors with alignment (4), we define Mix_λ in (2) as the mapping of $(\mathbf{A}, \mathbf{A}')$ to either (11) or (12), chosen at random.

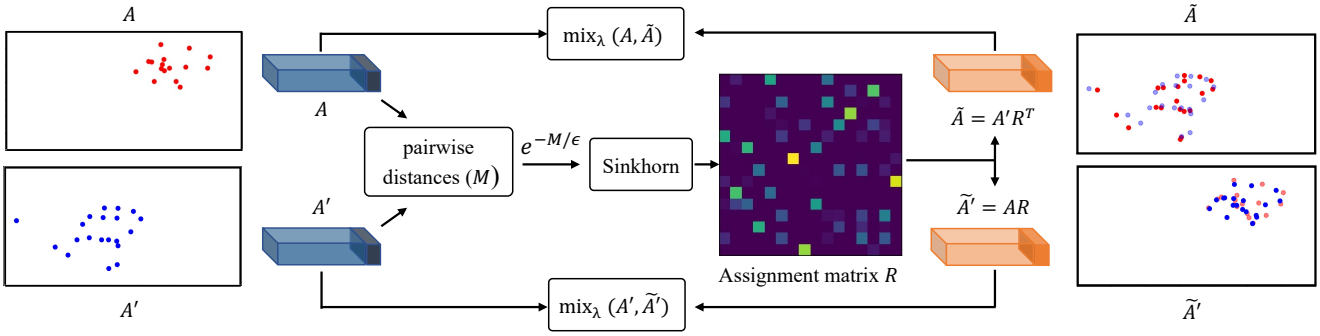

Figure 2. *Feature tensor alignment and interpolation.* Cost matrix M contains pairwise distances of feature vectors in tensors $\mathbf{A}, \mathbf{A}'$. Assignment matrix R is obtained by Sinkhorn-Knopp [34] on similarity matrix $e^{-M/\epsilon}$. $\mathbf{A}$ is aligned to $\mathbf{A}'$ according to R, giving rise to $\widetilde{\mathbf{A}}$. We then interpolate between $\mathbf{A}, \widetilde{\mathbf{A}}$. Symmetrically, we can align $\mathbf{A}'$ to $\mathbf{A}$ and interpolate between $\mathbf{A}', \widetilde{\mathbf{A}}'$. $\mathbf{A}, \mathbf{A}'$ on the left (toy example of 16 points in 2D) shown semi-transparent on the right for reference.

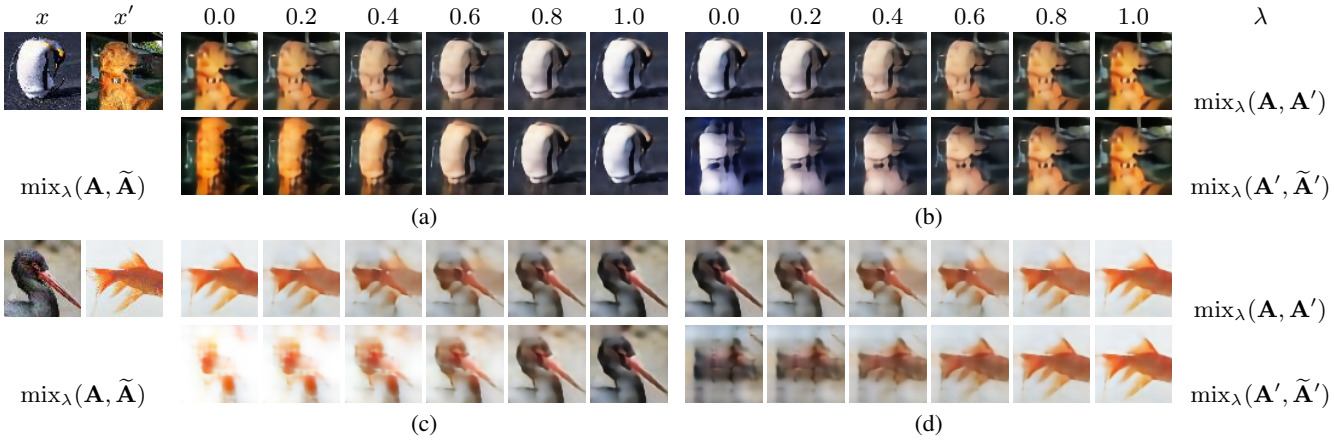

Figure 3. *Visualizing alignment.* For different $\lambda \in [0, 1]$, we interpolate feature tensors $\mathbf{A}, \mathbf{A}'$ without alignment (top) or aligned feature tensors (bottom) of two images x, x' and then we generate a new image by decoding the resulting embedding through the decoder D. (a), (c) We align $\mathbf{A}$ to $\mathbf{A}'$ and mix with (11). (b), (d) We align $\mathbf{A}'$ to $\mathbf{A}$ and mix with (12). Only meant for illustration: No decoded images are seen by the classifier at training.

3.3. Visualization and discussion

Decoder We use a decoder to study images generated with or without feature alignment. Let $f : \mathbb{R}^{c \times w \times h} \to \mathbb{R}^d$ be a FC layer mapping tensor $\mathbf{A}$ to embedding $e = f(A)$. We use $f \circ F$ as an encoder and a *decoder* $D : \mathbb{R}^d \to \mathcal{X}$ mapping e back to the image space, reconstructing image $\hat{x} = D(e)$. The autoencoder is trained using only clean images (without mixup) using *reconstruction* loss L_r between x and $\hat{x}$, where $L_r(x, x') := \|x - x'\|^2$ is the squared Euclidean distance. We use generated images only for visualization purposes below, but we also use the decoder optionally during AlignMixup training in section 4.

Discussion For different $\lambda \in [0, 1]$, we interpolate the feature tensors $\mathbf{A}, \mathbf{A}'$ of x, x' without or with alignment, using (11) or (12), and we generate a new image by decoding the resulting embedding through the decoder D.

In Figure 3, we visualize such generated images. Interestingly, by aligning $\mathbf{A}$ to $\mathbf{A}'$ and mixing using (11) with $\lambda = 0$, the generated image retains the pose of x and the texture of x'. In Figure 3(a) in particular, when x is 'penguin' and x' is 'dog', the generated image retains the pose of the penguin, while the texture of the dog aligns to the body of the penguin. Similarly, in Figure 3(c), the texture from the goldfish is aligned to that of the stork, while the pose of the stork is retained. Vice versa, as shown in Figure 3(b,d), by aligning $\mathbf{A}'$ to $\mathbf{A}$ and mixing using (12) with $\lambda = 0$, the generated image retains the pose of x' and the texture of x. By contrast, the image generated from unaligned features appears to be an overlay.

Randomly sampling several values of $\lambda \in [0, 1]$ during training generates an abundance of samples, capturing texture from one image and the pose from another. This allows the model to explore beyond the image manifold, thereby

DATASET	CIFAR-10		CIFAR-100		TI
NETWORK	R-18	W16-8	R-18	W16-8	R-18
Baseline	5.19	5.11	23.24	20.63	43.40*
Input [69]	4.03	3.98	20.21	19.88	43.48*
CutMix [65]	3.27	3.54	19.37	19.71	43.11*
Manifold [58]	2.95	3.56	19.80	19.23	40.76*
PuzzleMix [32]	2.93	**2.99**	20.01	19.25	36.52*
Co-Mixup [31]	2.89	3.04	19.81	19.57	35.85*
SaliencyMix [57]	2.99	3.53	19.69	19.59	34.81
StyleMix [28]	3.76	3.89	20.04	20.45	36.13
StyleCutMix [28]	3.06	3.12	19.34	19.28	34.49
AlignMixup (ours)	2.95	3.09	18.29	18.77	33.13
AlignMixup/AE (ours)	**2.83**	3.15	**17.82**	**18.09**	**32.73**
Gain	**+0.06**	**-0.10**	**+1.52**	**+1.14**	**+1.76**

Table 1. *Image classification* top-1 error (%) on CIFAR-10/100 and TI (TinyImagenet). Top-1 error (%): lower is better. Blue: second best. R: PreActResnet, W: WRN. $*$: reported by [31].

METHOD	PARAM.	MSEC/BATCH	TOP-1 ERROR
Baseline	25M	418	23.68
Input† [69]	25M	436	22.58
CutMix† [65]	25M	427	21.40
Manifold† [58]	25M	441	22.50
PuzzleMix† [32]	25M	846	21.24
Co-Mixup* [31]	25M	1022	–
SaliencyMix* [57]	25M	462	21.26
StyleMix* [28]	25M	828	-
StyleCutMix* [28]	25M	912	-
AlignMixup (ours)	25M	450	20.68
AlignMixup/AE (ours)	35M	688	**18.83**
Gain			**+2.41**

Table 2. *Image classification* top-1 error (%) and *computational analysis* on ImageNet using Resnet-50 for 300 epochs. Lower is better. Blue: second best. $*$: reported by authors; †: reported by PuzzleMix.

improving its generalization and enhancing its performance across multiple benchmarks, as discussed in section 4.

4. Experiments

4.1. Implementation details

Architecture We use a residual network as our encoder F. The output $\mathbf{A}$ is a $c \times 4 \times 4$ tensor. This is followed by a fully-connected layer as classifier g.

Autoencoder In Figure 3, we have used a decoder to visualize the effect of feature tensor alignment. In our experiments, we also use a decoder optionally during training of AlignMixup, to investigate its effect on representation learning under mixup. This results in a vanilla autoencoder architecture, which we denote as AlignMixup/AE. We use a residual generator [21] as the decoder D. The encoder and decoder have the same architecture.

Training We train AlignMixup using only the classification loss L_c (5) on mixed examples. For a given mini-batch during training, we mix either x or $\mathbf{A}$ (using either (11) or (12) with alignment). We choose between the three cases uniformly at random. For AlignMixup/AE, we either use the reconstruction loss L_r on clean examples, training the encoder and decoder, or the classification loss L_c (5) on mixed examples, training the encoder and classifier. This gives rise to a fourth case and we choose uniformly at random. The algorithm is in the supplementary material.

Hyperparameters The hyperparameters used for different datasets are reported in the supplementary material.

4.2. Image classification and robustness

We use PreActResnet18 [26] (R-18) and WRN16-8 [66] as the backbone architecture on CIFAR-10 and CIFAR-100 datasets [35]. Using the experimental settings of Manifold mixup [58] (in supplementary material), we reproduce the state-of-the-art (SOTA) mixup methods: baseline network (without mixup), Input mixup [69], Manifold mixup [58], CutMix [65], PuzzleMix [32], Co-Mixup [31], SaliencyMix [57], StyleMix [28] and StyleCutMix [28] using official code provided by the authors. We do not compare AlignMixup with AutoMix [72] and Re-Mix [5], since its experimental settings are different from ours and there is no available code.

In addition, we use R-18 as the backbone network on TinyImagenet [63] (TI) and reproduce SaliencyMix [57], StyleMix [28] and StyleCutMix [28] following the experimental settings of [32], and Resnet-50 (R-50) on ImageNet [48], following the training protocol of [32]. Using top-1 error (%) as evaluation metric, we show the effectiveness of AlignMixup on image classification and robustness to FGSM [19] and PGD [41] attacks.

Image classification As shown in Table 1, AlignMixup and AlignMixup/AE is on par or outperforms the SOTA methods by achieving the lowest top-1 error, especially on large datasets. On CIFAR-10, AlignMixup and AlignMixup/AE is on par with Co-Mixup and Puzzlemix with R-18 and WRN16-8. On CIFAR-100, AlignMixup outperforms StyleCutMix and Manifold mixup by 1.05% and 0.46% with R-18 and WRN16-8, respectively. On TI, AlignMixup outperforms Co-Mixup by 2.72% using R-18. From Table 2, AlignMixup/AE outperforms PuzzleMix by 2.41% on ImageNet. While the overall improvement by SOTA methods on ImageNet over Baseline is around 2%, AlignMixup/AE improves SOTA by another 2.5%.

Computational complexity Table 2 shows the computational analysis of AlignMixup training as compared with baseline and SOTA mixup methods on ImageNet, in terms of number of parameters and msec/batch on a NVIDIA RTX

ATTACK	FGSM					PGD			
DATASET	CIFAR-10		CIFAR-100		TI	CIFAR-10		CIFAR-100	
NETWORK	R-18	W16-8	R-18	W16-8	R-18	R-18	W16-8	R-18	W16-8
Baseline	89.41	88.02	87.12	72.81	91.85	99.99	99.94	99.97	99.99
Input [69]	78.42	79.21	81.30	67.33	88.68	99.77	99.43	99.96	99.37
CutMix [65]	77.72	78.33	86.96	60.16	88.68	99.82	98.10	98.67	97.98
Manifold [58]	77.63	76.11	80.29	56.45	89.25	97.22	98.49	99.66	98.43
PuzzleMix [32]	57.11	60.73	78.70	57.77	83.91	97.73	97.00	96.42	95.28
Co-Mixup [31]	60.19	58.93	77.61	56.59	–	97.59	96.19	95.35	94.23
SaliencyMix [57]	57.43	68.10	77.79	58.10	81.16	97.51	97.04	95.68	93.76
StyleMix [28]	79.54	71.05	80.54	67.94	84.93	98.23	97.46	98.39	98.24
StyleCutMix [28]	58.79	56.12	77.49	56.83	80.59	97.87	96.70	91.88	93.78
AlignMixup (ours)	54.83	56.20	**74.18**	**55.05**	**78.83**	**95.42**	96.71	**90.40**	**92.16**
AlignMixup/AE (ours)	**52.13**	**54.86**	76.40	55.44	78.98	97.16	**95.32**	91.69	92.23
Gain	**+4.98**	**+1.26**	**+3.31**	**+1.40**	**+1.76**	**+1.80**	**+0.87**	**+1.48**	**+1.60**

Table 3. *Robustness to FGSM & PGD attacks*. Top-1 error (%): lower is better. Blue: second best. Gain: reduction of error. TI: TinyImagenet. R: PreActResnet, W: WRN.

2080 TI GPU. AlignMixup has nearly the same computational overhead as Manifold mixup while achieving 1.82% increase of accuracy. While SOTA methods like Co-Mixup and PuzzleMix are computationally more expensive than AlignMixup by 1.8× and 2.3× respectively, they are outperformed by AlignMixup by 0.6% on average. AlignMixup/AE brings a further 1.85% gain in accuracy over AlignMixup. It is important to note that 40% increase in number of parameters of AlignMixup/AE is due to the residual decoder, which is only used in one out of five cases on clean images without mixup. Computational complexity during inference is the same for all methods.

Challenges From Table 1, we observe that AlignMixup achieves SoTA top-1 error on CIFAR-10 and CIFAR-100. These results are computed using 2000 epochs following the experimental settings of [58], which also achieves its best performance at 2000 epochs. While baseline mixup methods [28, 31, 32, 57, 65, 69] perform best at 300 epochs, they do not benefit from long training time. Unlike these methods, which perform mixup in the image space, Manifold mixup [58] and AlignMixup performs mixup in the feature space. We hypothesize that this takes longer training time until the network learns some meaningful representations. It is even more challenging in our case, since we mix features at deeper layers comparing with Manifold mixup. Empirically, when trained for 2000 epochs instead of 300 epochs, the top-1 error drops from 21.64 → 19.80 for Manifold mixup and from 21.38 → 18.29 for AlignMixup.

Robustness to FGSM and PGD attacks Following the evaluation protocol of [32], we use $8/255$ l_∞ ϵ-ball for FGSM and $4/255$ l_∞ ϵ-ball with step size $2/255$ for PGD. We reproduce the results of competitors for FGSM and PGD on CIFAR-10 and CIFAR-100; results of baseline, Input, Manifold, Cutmix and Puzzlemix on TI for FGSM are as reported in [32] and reproduced for SaliencyMix, StyleMix and StyleCutMix.

As shown in Table 3, AlignMixup is more robust comparing to SOTA methods. While AlignMixup is on par with PuzzleMix and Co-Mixup on CIFAR-10 image classification, it outperforms Co-Mixup and PuzzleMix by 5.36% and 2.28% in terms of robustness to FGSM attacks. There is also significant gain of robustness to FGSM on TinyImageNet and to the stronger PGD on CIFAR-100.

4.3. Overconfidence

Deep neural networks tend to be overconfident about incorrect predictions far away from the training data and mixup helps combat this problem. Two standard benchmarks to evaluate this improvement are their ability to detect *out-of-distribution* data and their *calibration*, *i.e.*, the discrepancy between accuracy and confidence.

Out-of-distribution detection According to [27], *in-distribution* (ID) refers to a test example drawn from the same distribution which the network is trained on, while a sample drawn from any other distribution is *out-of-distribution* (OOD). At inference, given a mixture of ID and OOD examples, the network assigns probabilities to the known classes by softmax. An example is then classified as OOD if the maximum class probability is below a certain threshold, else ID. A well-calibrated network should be able to assign a higher probability to ID than OOD examples, making it easier to distinguish the two distributions.

We compare AlignMixup with SOTA methods trained using R-18 on CIFAR-100 as discussed in subsection 4.2. At inference, ID examples are test images from CIFAR-100, while OOD examples are test images from LSUN (crop) [64], iSUN [62] and Tiny-ImageNet (crop); where crop denotes that the OOD examples are center-cropped to

Task	Out-Of-Distribution Detection											
Dataset	LSUN (crop)				iSUN				TI (crop)			
Metric	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)
Baseline	54.0	47.1	54.5	45.6	66.5	72.3	74.5	69.2	61.2	64.8	67.8	60.6
Input [69]	57.5	59.3	61.4	55.2	59.6	63.0	60.2	63.4	58.7	62.8	63.0	62.1
Cutmix [65]	63.8	63.1	61.9	63.4	67.0	76.3	81.0	77.7	70.4	84.3	87.1	80.6
Manifold [58]	58.9	60.3	57.8	59.5	64.7	73.1	80.7	76.0	67.4	69.9	69.3	70.5
PuzzleMix [32]	64.3	69.1	80.6	73.7	73.9	77.2	79.3	71.1	71.8	76.2	78.2	81.9
Co-Mixup [31]	70.4	75.6	82.3	70.3	68.6	80.1	82.5	75.4	71.5	84.8	86.1	80.5
SaliencyMix [57]	68.5	79.7	82.2	64.4	65.6	76.9	78.3	79.8	73.3	83.7	87.0	82.0
StyleMix [28]	62.3	64.2	70.9	63.9	61.6	68.4	67.6	60.3	67.8	73.9	71.5	78.4
StyleCutMix [28]	70.8	78.6	83.7	74.9	70.6	82.4	83.7	76.5	75.3	82.6	82.9	78.4
AlignMixup (ours)	74.2	79.9	84.1	75.1	72.8	83.2	84.1	80.3	77.2	85.0	87.8	85.0
AlignMixup/AE (ours)	**76.9**	**83.5**	**86.7**	**79.4**	**75.6**	**84.1**	**85.9**	**81.7**	**79.7**	**88.0**	**89.7**	**85.7**
Gain	**+6.1**	**+3.8**	**+3.0**	**+4.5**	**+1.7**	**+1.7**	**+2.2**	**+1.9**	**+4.4**	**+3.2**	**+2.6**	**+3.8**

Table 4. *Out-of-distribution detection* using PreActResnet18. Det Acc (detection accuracy), AuROC, AuPR (ID) and AuPR (OOD): higher is better; Blue: second best. Gain: increase in performance. TI: TinyImagenet. Additional results are in the supplementary material.

Metric	Top-1 loc.		MaxBoxAcc-v2	
Network	VGG-GAP	ResNet-50	VGG-GAP	ResNet-50
ACoL [70]	45.9	–	57.4	–
ADL [8]	52.4	–	61.3	58.4
Baseline CAM [71]	37.1	49.4	59.0	59.7
Input [69]	41.7	49.3	57.1	60.6
CutMix [65]	52.5	54.8	62.6	64.8
AlignMixup (ours)	**53.1**	**56.2**	**63.8**	**65.4**
Gain	**+0.6**	**+1.4**	**+1.2**	**+0.6**

Table 5. *Weakly-supervised object localization* on CUB200-2011. Top-1 loc.: Top-1 localization accuracy (%), MaxBoxAcc-v2: Maximal box accuracy [7]. Higher is better. Blue: second best. Gain: increase of accuracy.

32×32 to match the resolution of ID images [65]. Following [27], we measure *detection accuracy* (Det Acc) using a threshold of 0.5, *area under ROC curve* (AuROC) and *area under precision-recall curve* (AuPR).

As shown in Table 4, AlignMixup outperforms SOTA methods under all metrics by a large margin, indicating that it is better in reducing over-confident predictions. We further observe that Input mixup is inferior to Baseline, which is consistent with the findings of [65]. More results are given in the supplementary material.

Calibration According to [13], calibration measures the discrepancy between the accuracy and confidence level of a network's predictions. A poorly calibrated network may make incorrect predictions with high confidence. In the supplementary, we compare AlignMixup with SOTA methods using calibration plots and quantitative experiments.

4.4. Weakly-supervised object localization (WSOL)

WSOL aims to localize an object of interest using only class labels *without bounding boxes* at training. WSOL works by extracting visually discriminative cues to guide the classifier to focus on salient regions in the image.

We train AlignMixup using the same procedure as for image classification. At inference, following [65], we compute a saliency map using CAM [71], binarize it using a threshold of 0.15 and take the bounding box of the mask. We use VGG-GAP [51] and Resnet-50 [26] as pretrained on Imagenet [48] and we fine-tune them on CUB200-2011 [60]. We follow the evaluation protocol by [7] and use top-1 localization accuracy with IoU threshold of 0.5 and Maximal Box Accuracy (MaxBoxAcc-v2) to compare AlignMixup with baseline CAM (without mixup), Input mixup [69], CutOut [14] and CutMix [65].

According to Table 5, AlignMixup outperforms Input mixup, CutOut and CutMix by 11.4%, 7.3% and 0.6% respectively using VGG-GAP and by 6.9%, 3.8% and 1.4% respectively using Resnet-50 in terms of top-1 localization accuracy. Furthermore, AlignMixup outperforms CutMix by 1.2% and 0.6% using VGG-GAP and Resnet-50 respectively in terms of MaxBoxAcc-v2. It also outperforms dedicated WSOL methods ACoL [70] and ADL [8], which focus on learning spatially dispersed representations. Qualitative localization results are given in the supplementary material.

4.5. Ablation study

All ablations are performed on CIFAR-100 using R-18 as encoder F with feature tensor $\mathbf{A}$ being $512 \times 4 \times 4$. We study the effect of mixing at different layers (x, $\mathbf{A}$), by aligning $\mathbf{A}$ or not before mixing, as well as using a decoder D in an different autoencoder architectures. We report top-1 accu-

METHOD/ARCH	LAYERS	UNALIGNED	ALIGNED
Baseline		76.76	–
Manifold [58]		80.20	-
StyleCutMix [28]		80.66	-
AlignMixup	$\{x, e\}$	80.81	–
	$\{\mathbf{A}\}$	79.07	80.28
	$\{e\}$	78.71	-
	$\{x, \mathbf{A}\}$	80.34	81.71
	$\{x, \mathbf{A}, \mathbf{E}\}$	80.46	81.36
	$\{x, \mathbf{A}, e\}$	80.33	**81.92**
AlignMixup/AE	$\{x, e\}$	81.92	-
	$\{\mathbf{A}\}$	79.39	81.04
	$\{e\}$	79.49	-
	$\{x, \mathbf{A}\}$	81.78	81.85
	$\{x, \mathbf{E}\}$	80.80	81.54
	$\{x, \mathbf{A}, e\}$	81.61	**82.18**
AlignMixup/AE	$\{x, \mathbf{A}_{2\times2}, e\}$	81.47	81.20
	$\{x, \mathbf{A}_{4\times4}, e\}$	81.61	82.18
	$\{x, \mathbf{A}_{8\times8}, e\}$	80.49	**82.20**
AlignMixup/VAE	$\{x, (\mu, \sigma)\}$	81.81	–
	$\{x, \mathbf{A}\}$	81.35	81.85
	$\{x, (\mathbf{M}, \mathbf{\Sigma})\}$	80.45	81.10
	$\{x, \mathbf{A}, (\mu, \sigma)\}$	81.00	**81.89**

Table 6. *Ablations* using R-18 on CIFAR-100. Top-1 classification accuracy (%): higher is better. Arch: autoencoder architecture. AE: vanilla; VAE: variational [33].

racy (%). All results are in Table 6. The ablation showing the effect of the number of iterations in Sinkhorn-Knopp algorithm is summarized in the supplementary material.

Layers We study the choice of layers to mix, regardless of feature alignment. According to (2), we may mix at any of two layers, represented by $\{x, \mathbf{A}\}$. To investigate more diverse cases, we introduce an additional layer f to the encoder of AlignMixup and mix its output, which acts as the latent space of AlignMixup/AE. f could be a FC layer, which outputs a vector $e \in \mathbb{R}^{512}$, or a convolutional layer of kernel size 2×2 and stride 2, producing a $128 \times 2 \times 2$ tensor $\mathbf{E}$. In both cases, mixup is also represented by (2), where now $f_1 := f \circ F$ and $f_2 := \mathrm{id}$. Mixing is now chosen from $\{x, \mathbf{A}, e\}$ or $\{x, \mathbf{A}, \mathbf{E}\}$. In AlignMixup (no decoder), among different choices of unaligned layer sets, mixing $\{x, e\}$ results in the highest classification accuracy. Furthermore, AlignMixup/AE outperforms baseline and the best performing competitor StyleCutMix for all choices of layers, even when features are unaligned, showing a motivation to use the decoder.

Tensor alignment We investigate the effect of aligning feature tensors or not before mixing it, by using standard mixup (2) or (11), (12), respectively. It is important to note that when e is a vector, we do not align it. In AlignMixup, we observe that aligning tensors $\mathbf{A}$ and $\mathbf{E}$ before mixing improves classification accuracy significantly. Furthermore, we observe that using an additional FC layer for e brings only minor improvement ($81.71 \rightarrow 81.92$), meaning that the major improvement comes from alignment. Overall, AlignMixup/AE works the best when $x, \mathbf{A}, e$ are mixed, with $\mathbf{A}$ being aligned. It outperforms StyleCutMix by 1.52%.

Alignment resolution Given the best settings of AlignMixup/AE, we investigate the effect of aligning $\mathbf{A}$ at different spatial resolutions. The default is 4×4, denoted as $\mathbf{A}_{4\times4}$. We also experiment 2×2 ($\mathbf{A}_{2\times2}$), obtained by average pooling, and 8×8 ($\mathbf{A}_{8\times8}$), by removing downsampling from the last convolutional layer. The accuracy of 8×8 is only slightly better than 4×4 by 0.02%, while being computationally more expensive. Thus, we choose 4×4 as the default. By contrast, aligning at 2×2 is worse than not aligning at all. This may be due to soft correspondences causing loss of information by averaging.

Autoencoder architecture We compare AlignMixup with two autoencoder architectures: the *vanilla autoencoder* (AlignMixup/AE), and a *variational autoencoder* [33] (AlignMixup/VAE). The latter has two vectors $\mu, \sigma \in \mathbb{R}^{512}$ instead of e, representing mean and standard deviation, respectively. We also investigate $128 \times 2 \times 2$ tensors, denoted as $\mathbf{M}, \mathbf{\Sigma}$ where the two variables are mixed simultaneously. As for AlignMixup and AlignMixup/AE, we investigate different combinations of layers with or without alignment. All three architectures work best when $x, \mathbf{A}, e$ are mixed. Alignment improves consistently on all three architectures. Both AlignMixup and AlignMixup/VAE are inferior to AlignMixup/AE. However, their best settings still outperform Baseline and StyleCutMix.

5. Conclusion

We have shown that mixup of a combination of input and latent representations is a simple and very effective pairwise data augmentation method. The gain is most prominent on large datasets and in combating overconfidence in predictions, as indicated by out-of-distribution detection. Interpolation of feature tensors boosts performance significantly, but only if they are aligned.

Our work is a compromise between a "good" handcrafted interpolation in the image space and a fully learned one in the latent space. A challenge is to make progress in the latter direction without compromising speed and simplicity, which would affect wide applicability.

6. Acknowledgement

This work was in part supported by the ANR-19-CE23-0028 MEERQAT project and was performed using the HPC resources from GENCI-IDRIS Grant 2021 AD011012528. This work was partially done while Yannis was at Inria. We also thank Konstantinos Tertikas for his amazing help with adapting AlignMixup to transformers.

References

[1] David Alvarez-Melis and Tommi S Jaakkola. Gromov-wasserstein alignment of word embedding spaces. In *EMNLP*, 2018. 3

[2] Christopher Beckham, Sina Honari, Vikas Verma, Alex Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Christopher Pal. On adversarial mixup resynthesis. In *NeurIPS*, 2019. 2

[3] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In *ICML*, 2013. 1

[4] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. In *ICLR*, 2019. 1, 2

[5] Jie Cao, Luanxuan Hou, Ming-Hsuan Yang, Ran He, and Zhenan Sun. Remix: Towards image-to-image translation with limited data. In *CVPR*, 2021. 5

[6] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. In *ECCV*, 2020. 2

[7] Junsuk Choe, Seong Joon Oh, Seungho Lee, Sanghyuk Chun, Zeynep Akata, and Hyunjung Shim. Evaluating weakly supervised object localization methods right. In *CVPR*, 2020. 7

[8] Junsuk Choe and Hyunjung Shim. Attention-based dropout layer for weakly supervised object localization. In *CVPR*, 2019. 7

[9] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *NeurIPS*, 2016. 2

[10] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning augmentation strategies from data. In *CVPR*, 2019. 1

[11] Marco Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. In *NeurIPS*, 2013. 2, 3

[12] Ali Dabouei, Sobhan Soleymani, Fariborz Taherkhani, and Nasser M Nasrabadi. Supermix: Supervising the mixing data augmentation. In *CVPR*, 2021. 2

[13] Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 1983. 7

[14] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2, 7

[15] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. In *NeurIPS*, 2020. 2

[16] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Unsupervised feature learning by augmenting single images. In *ICLR Workshops*, 2014. 1

[17] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 13

[18] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *AISTATS*, 2018. 2

[19] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015. 5

[20] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: A vision transformer in convnet's clothing for faster inference. In *ICCV*, 2021. 12

[21] Ishaan Gulrajani, Faruk Ahmed, Martin Arjovsky, Vincent Dumoulin, and Aaron Courville. Improved training of wasserstein gans. In *ICLR*, 2018. 5

[22] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 12

[23] Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *AAAI*, 2019. 2

[24] Kai Han, Rafael S Rezende, Bumsub Ham, Kwan-Yee K Wong, Minsu Cho, Cordelia Schmid, and Jean Ponce. Scnet: Learning semantic correspondence. In *ICCV*, 2017. 2

[25] 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*, 2020. 2

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 7

[27] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017. 6, 7, 12

[28] Minui Hong, Jinwoo Choi, and Gunhee Kim. Stylemix: Separating content and style for enhanced data augmentation. In *CVPR*, 2021. 2, 5, 6, 7, 8, 12, 13

[29] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 2

[30] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018. 2

[31] Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-mixup: Saliency guided joint mixup with supermodular diversity. In *ICLR*, 2021. 1, 2, 5, 6, 7, 11, 12, 13

[32] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*, 2020. 1, 2, 3, 5, 6, 7, 11, 12, 13

[33] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014. 8

[34] Philip A Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 2008. 3, 4, 11

[35] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009. 5

[36] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NIPS*, 2012. 1

[37] 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 *ECCV*, 2014. 13

[38] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016.

13

[39] Xiaofeng Liu, Yang Zou, Lingsheng Kong, Zhihui Diao, Junliang Yan, Jun Wang, Site Li, Ping Jia, and Jane You. Data augmentation via latent space interpolation for image classification. In *ICPR*, 2018. 2

[40] Jonathan Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In *NIPS*, 2014. 2

[41] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *ICLR*, 2018. 5

[42] Giorgio Patrini, Rianne van den Berg, Patrick Forre, Marcello Carioni, Samarth Bhargav, Max Welling, Tim Genewein, and Frank Nielsen. Sinkhorn autoencoders. In *Uncertainty in Artificial Intelligence*, 2020. 2

[43] Mattis Paulin, Jérôme Revaud, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Transformation pursuit for image classification. In *CVPR*, 2014. 1

[44] 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. 2

[45] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 13

[46] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. End-to-end weakly-supervised semantic alignment. In *CVPR*, 2018. 2

[47] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *IJCV*, 2000. 2

[48] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *IJCV*, 2015. 5, 7

[49] Patrice Y Simard, Yann A LeCun, John S Denker, and Bernard Victorri. Transformation invariance in pattern recognition—tangent distance and tangent propagation. In *Neural networks: tricks of the trade*. 1998. 1

[50] Oriane Siméoni, Yannis Avrithis, and Ondrej Chum. Local features and visual words emerge in activations. In *CVPR*, 2019. 2

[51] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 7

[52] Cecilia Summers and Michael J Dinneen. Improved mixed-example data augmentation. In *WACV*, 2019. 2

[53] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *ICLR*, 2014. 1

[54] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *ACML*, 2018. 2

[55] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, 2019. 12

[56] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition. In *ICLR*, 2018. 2

[57] A F M Uddin, Mst. Monira, Wheemyung Shin, TaeChoong Chung, and Sung-Ho Bae. SaliencyMix: A saliency guided data augmentation strategy for better regularization. In *ICML*, 2021. 1, 2, 5, 6, 7, 12, 13

[58] 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 *ICML*, 2019. 1, 2, 3, 5, 6, 7, 8, 11, 12, 13

[59] Cédric Villani. *Optimal transport: old and new*. Springer Science & Business Media, 2008. 2, 3

[60] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. Technical Report CNS-TR-2011-001, California Institute of Technology, 2011. 7

[61] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 2

[62] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 6

[63] Leon Yao and John Miller. Tiny imagenet classification with convolutional neural networks. Technical report, Standford University, 2015. 5

[64] Fisher Yu, Ari Seff, Yinda Zhang, Shuran Song, Thomas Funkhouser, and Jianxiong Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*, 2015. 6, 12

[65] 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 *ICCV*, 2019. 1, 2, 5, 6, 7, 11, 12, 13

[66] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *BMVC*, 2016. 5

[67] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *ICLR*, 2017. 1

[68] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *CVPR*, 2020. 2

[69] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 1, 2, 3, 5, 6, 7, 11, 12, 13

[70] Xiaolin Zhang, Yunchao Wei, Jiashi Feng, Yi Yang, and Thomas S Huang. Adversarial complementary learning for weakly supervised object localization. In *CVPR*, 2018. 7

[71] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 7, 11

[72] Jianchao Zhu, Liangliang Shi, Junchi Yan, and Hongyuan Zha. Automix: Mixup networks for sample interpolation via cooperative barycenter learning. In *ECCV*, 2020. 2, 5

A. Algorithm

AlignMixup and AlignMixup/AE are summarized in algorithm 1. By default (AlignMixup), for each mini-batch, we uniformly draw at random one among three choices (line 2) over mixup on input (x) or feature tensors ($\mathbf{A}$, using either (11) or (12) for mixing). For AlignMixup/AE, there is a fourth choice where we only use reconstruction loss on clean examples (line 7).

For mixup, we use only classification loss (5) (line 24). Following [58], we form, for each example (x, y) in the mini-batch, a paired example (x', y') from the same mini-batch regardless of class labels, by randomly permuting the indices (lines 1,10). Inputs x, x' are mixed by (2),(3) (line 12). Feature tensors $\mathbf{A}$ and $\mathbf{A}'$ are first aligned and then mixed by (2),(11) ($\mathbf{A}$ aligns to $\mathbf{A}'$) or (2),(12) ($\mathbf{A}'$ aligns to $\mathbf{A}$) (lines 14,23).

Algorithm 1: AlignMixup/AE (parts involved in the AE variant indicated in blue)

Input: encoders F; embedding e, decoder D; classifier g
Input: mini-batch $B := \{(x_i, y_i)\}_{i=1}^{b}$
Output: loss values $L := \{\ell_i\}_{i=1}^{b}$

1 $\pi \sim \text{unif}(S_b)$ ▷ random permutation of $\{1, \ldots, b\}$
2 *mode* $\sim \text{unif}\{\textit{clean}, \textit{input}, \textit{feat}, \textit{feat}'\}$ ▷ mixup?
3 **for** $i \in \{1, \ldots, b\}$ **do**
4 $(x, y) \leftarrow (x_i, y_i)$ ▷ current example
5 **if** *mode* $=$ *clean* **then** ▷ no mixup
6 $\hat{x} \leftarrow D(e(F(x)))$ ▷ encode/decode
7 $\ell_i \leftarrow L_r(x, \hat{x})$ ▷ reconstruction loss
8 **else** ▷ mixup
9 $\lambda \sim \text{Beta}(\alpha, \alpha)$ ▷ interpolation factor
10 $(x', y') \leftarrow (x_{\pi(i)}, y_{\pi(i)})$ ▷ paired example
11 **if** *mode* $=$ *input* **then** ▷ as in [69]
12 *out* $\leftarrow F(\text{mix}_\lambda(x, x'))$
▷ (2),(3) **else** ▷ *mode* $\in \{\textit{feat}, \textit{feat}'\}$
13 **if** *mode* $=$ *feat*′ **then** ▷ choose (12) over (11)
14 SWAP (x, x'), SWAP (y, y')
15 $\mathbf{A} \leftarrow F(x)$, $\mathbf{A}' \leftarrow F(x')$ ▷ feature tensors
16 $A \leftarrow \text{RESHAPE}_{c \times r}(\mathbf{A})$ ▷ to matrix
17 $A' \leftarrow \text{RESHAPE}_{c \times r}(\mathbf{A}')$
18 $M \leftarrow \text{DIST}(A, A')$ ▷ pairwise distances (6)
19 $P^* \leftarrow \text{SINKHORN}(\exp(-M/\epsilon))$ ▷ tran. plan (8)
20 $R \leftarrow \text{DETACH}(rP^*)$ ▷ assignments
21 $\widetilde{A} \leftarrow A'R^\top$ ▷ alignment (9)
22 $\widetilde{\mathbf{A}} \leftarrow \text{RESHAPE}_{c \times w \times h}(\widetilde{A})$ ▷ to tensor
23 *out* $\leftarrow f(\text{mix}_\lambda(\mathbf{A}, \widetilde{\mathbf{A}}))$ ▷ (2),(11)
24 $\ell_i \leftarrow L_c(g(out), \text{mix}_\lambda(y, y'))$ ▷ classification loss (5)

In computing loss derivatives, we backpropagate through feature tensors $\mathbf{A}, \mathbf{A}'$ but not through the transport plan P^* (line 20). Hence, although the Sinkhorn-Knopp algorithm [34] is differentiable, its iterations take place only in the forward pass. Importantly, AlignMixup is easy to implement and does not require sophisticated optimization like [31, 32].

B. Hyperparameter settings

CIFAR-10/CIFAR-100 We train AlignMixup using SGD for 2000 epochs with an initial learning rate of 0.1, decayed by a factor 0.1 every 500 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 128. The interpolation factor is drawn from $\text{Beta}(\alpha, \alpha)$ where $\alpha = 2.0$. Using these settings, we reproduce the results of SOTA mixup methods for image classification, robustness to FGSM and PGD attacks, calibration and out-of-distribution detection. For alignment, we apply the Sinkhorn-Knopp algorithm [34] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

TinyImagenet We follow the training protocol of Kim *et al*. [32], training R-18 as stage-1 encoder F using SGD for 1200 epochs. We set the initial learning rate to 0.1 and decay it by 0.1 at 600 and 900 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 128 on 2 GPUs. The interpolation factor is drawn from $\text{Beta}(\alpha, \alpha)$ where $\alpha = 2.0$. For alignment, we apply the Sinkhorn-Knopp algorithm [34] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

ImageNet We follow the training protocol of Kim *et al*. [32], where training R-50 as F using SGD for 300 epochs. The initial learning rate of the classifier and the remaining layers is set to 0.1 and 0.01, respectively. We decay the learning rate by 0.1 at 100 and 200 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 100 on 4 GPUs. The interpolation factor is drawn from $\text{Beta}(\alpha, \alpha)$ where $\alpha = 2.0$. For alignment, we apply the Sinkhorn-Knopp algorithm [34] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

We also train R-50 on ImageNet for 100 epochs, following the training protocol described in Kim *et al*. [31].

CUB200-2011 For weakly-supervised object localization (WSOL), we use VGG-GAP and R-50 pretrained on ImageNet as F. The training strategy for WSOL is the same as image classification and the network is trained *without bounding box information*. In R-50, following [65], we modify the last residual block (`layer 4`) to have stride 2 instead of 1, resulting in a feature map of spatial resolution 14×14. The modified architecture of VGG-GAP is the same as described in [71]. The classifier is modified to have 200 classes instead of 1000.

For fair comparisons with [65], during training, we resize the input image to 256×256 and randomly crop the resized image to 224×224. During testing, we directly re-

NETWORK	RESNET-50
Baseline	24.03
Input [69]	22.97
Manifold [58]	23.30
CutMix [65]	22.92
PuzzleMix [32]	22.49
Co-Mixup [31]	22.39
StyleMix [28]	24.06
StyleCutMix [28]	22.71
AlignMixup (ours)	**22.0**
Gain	**+0.39**

Table 7. *Image classification* on ImageNet for 100 epochs using ResNet-50. Top-1 error (%): lower is better. Blue: second best. Gain: reduction of error.

size to 224×224. We train the network for 600 epochs using SGD. For R-50, the initial learning rate of the classifier and the remaining layers is set to 0.01 and 0.001, respectively. For VGG, the initial learning rate of the classifier and the remaining layers is set to 0.001 and 0.0001, respectively. We decay the learning rate by 0.1 every 150 epochs. The momentum is set to 0.9 with weight decay of 0.0001 and batch size of 16.

C. Additional experiments

ImageNet classification Following the training protocol of [31], Table 7 reports classification performance when training for 100 epochs on ImageNet. Using the top-1 error (%) reported for competitors by [31], AlignMixup outperforms all methods, including Co-Mixup [31]. Importantly, while the overall improvement by SOTA methods over Baseline is around 1.64%, AlignMixup improves SOTA by another 0.4%.

Experiments using transformers We apply mixup to LeViT-128S [20] on ImageNet for 100 epochs. For AlignMixup, we align the feature tensors in the last layer of the convolution stem. The top-1 accuracy is: baseline 67.4%, input mixup 68.3%, manifold mixup 67.8%, CutMix 68.7%, AlignMixup 69.9%. Thus, we outperform input mixup and CutMix by **1.6%** and **1.2%** respectively, which in turn outperform the baseline by **0.9%** and **1.3%** respectively. This means that the improvement brought by mixing is roughly doubled.

Out-of-distribution detection We compare AlignMixup with SOTA methods, training R-18 on CIFAR-100 as discussed in subsection 4.2. At inference, ID examples are test images from CIFAR-100, while OOD examples are test images from LSUN [64] and Tiny-ImageNet, resizing OOD examples to 32×32 to match the resolution of ID images [65]. We also use test images from CIFAR-100 with

DATASET	LSUN (RESIZE)				TI (RESIZE)			
METRIC	DET ACC	AU ROC	AUPR (ID)	AUPR (OOD)	DET ACC	AU ROC	AUPR (ID)	AUPR (OOD)
Baseline	67.6	73.3	76.6	68.9	65.1	70.6	73.1	67.1
Input [69]	61.5	66.5	66.4	65.8	59.6	63.8	63.0	63.4
Cutmix [65]	71.3	77.4	79.1	75.5	69.1	79.4	79.8	73.3
Manifold [58]	67.8	78.9	76.3	71.3	62.5	77.8	76.8	72.2
PuzzleMix [32]	74.9	79.9	84.0	77.5	73.9	77.3	80.6	71.9
Co-Mixup [31]	73.8	82.6	86.8	76.9	68.1	78.9	82.5	74.2
SaliencyMix [57]	75.8	79.7	82.2	84.4	75.3	81.2	83.8	79.5
StyleMix [28]	73.0	74.6	72.4	73.4	72.9	79.5	78.2	74.6
StyleCutMix [28]	74.3	83.1	86.9	78.9	73.8	80.9	83.1	76.3
AlignMixup (ours)	76.1	84.3	87.1	**85.8**	74.7	82.6	86.1	80.9
AlignMixup/AE (ours)	**77.0**	**85.8**	**87.9**	83.7	**76.2**	**84.8**	**87.2**	**82.3**
Gain	**+2.1**	**+2.7**	**+1.0**	**+1.4**	**+0.9**	**+3.6**	**+3.4**	**+2.8**
NOISE	UNIFORM				GAUSSIAN			
Baseline	58.3	75.3	75.0	69.0	60.8	64.3	62.9	63.9
Input [69]	50.0	67.9	71.8	71.7	60.2	65.0	63.1	64.1
Cutmix [65]	74.8	80.0	84.9	72.4	75.7	79.0	84.0	70.9
Manifold [58]	69.8	75.9	83.2	71.9	70.8	78.8	81.3	71.6
PuzzleMix [32]	78.6	85.2	86.0	74.4	78.5	85.1	85.9	74.3
Co-Mixup [31]	80.4	87.6	87.4	75.2	81.6	78.6	89.5	74.2
SaliencyMix [57]	83.1	87.4	89.1	76.6	82.4	85.4	81.1	**81.3**
StyleMix [28]	75.3	71.8	77.8	65.5	78.0	75.2	84.3	71.0
StyleCutMix [28]	84.5	83.2	88.6	78.3	84.8	81.9	83.3	73.9
AlignMixup (ours)	86.9	89.1	93.6	77.7	**86.7**	**87.9**	91.8	77.4
AlignMixup/AE (ours)	**88.0**	**90.6**	**94.0**	**80.8**	86.0	87.2	**91.9**	75.6
Gain	**+3.5**	**+3.0**	**+4.9**	**+2.5**	**+1.9**	**+2.8**	**+2.4**	**-3.9**

Table 8. *Out-of-distribution detection* on different datasets (top) and under different noise (bottom) using PreActResnet18. Det Acc (detection accuracy), AuROC, AuPR (ID) and AuPR (OOD): higher is better. Blue: second best. Gain: increase in performance. TI: TinyImagenet.

Uniform and Gaussian noise as OOD samples. Uniform is drawn from $\mathcal{U}(0,1)$ and Gaussian from $\mathcal{N}(\mu,\sigma)$ with $\mu = \sigma = 0.5$. All SOTA mixup methods are reproduced using the same experimental settings. Following [27], we measure *detection accuracy* (Det Acc) using a threshold of 0.5, *area under ROC curve* (AuROC) and *area under precision-recall curve* (AuPR).

As shown in Table 8, AlignMixup outperforms SOTA methods under all metrics by a large margin, indicating that it is better in reducing over-confident predictions.

Calibration We compare AlignMixup with SOTA methods , training R-18 on CIFAR-100 as discussed in subsection 4.2. All SOTA mixup methods are reproduced using the same experimental settings. We compare qualitatively by plotting accuracy *vs*. confidence. As shown in Figure 4, while Baseline is clearly overconfident and Input and Manifold mixup are clearly under-confident, AlignMixup results in the best calibration among all competitors. We also compare quantitatively, measuring the *expected calibration error* (ECE) [22] and *overconfidence error* (OE) [55]. As shown in Table 9, AlignMixup outperforms SOTA methods by achieveing lower ECE and OE, indicating that it is better calibrated.

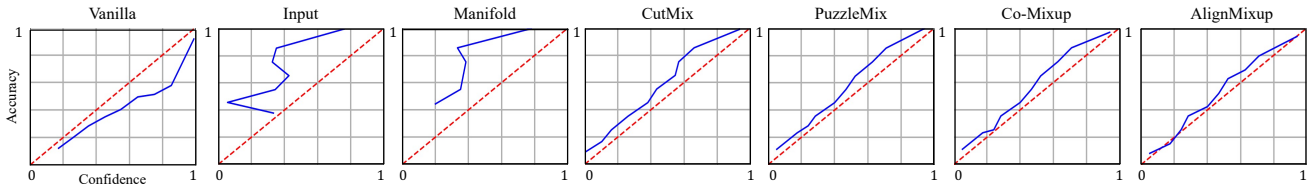

Figure 4. *Calibration plots* on CIFAR-100 using PreActResnet18: near diagonal is better. We plot accuracy *vs*. confidence, that is, probability for the predicted class.

METRIC	ECE	OE
Baseline	10.25	1.11
Input [69]	18.50	1.42
CutMix [65]	7.60	1.05
Manifold [58]	18.41	0.79
PuzzleMix [32]	8.22	0.61
Co-Mixup [31]	5.83	0.55
SaliencyMix [57]	5.89	0.59
StyleMix [28]	11.43	1.31
StyleCutMix [28]	9.30	0.87
AlignMixup (ours)	5.78	**0.41**
AlignMixup/AE (ours)	**5.06**	0.48
Gain	**+0.77**	**+0.14**

Table 9. *Calibration* using PreActResnet18 on CIFAR-100. ECE: expected calibration error; OE: overconfidence error. Lower is better. Blue: second best. Gain: reduction of error.

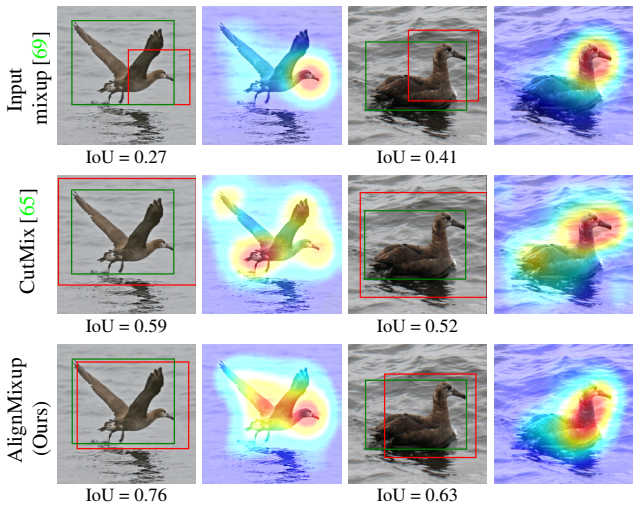

Figure 5. *Localization examples* using ResNet-50 on CUB200-2011. Red boxes: predicted; green: ground truth.

Qualitative results of WSOL Qualitative localization results shown in Figure 5 indicate that AlignMixup encodes semantically discriminative representations, resulting in better localization performance.

Object detection Following the settings of CutMix [65], we use Resnet-50 pretrained on ImageNet using AlignMixup as the backbone of SSD [38] and Faster R-CNN [45] detectors and fine-tune it on Pascal VOC07 [17] and MS-COCO [37] respectively. AlignMixup outperforms CutMix mAP by **0.8%** (77.6 $\rightarrow$ 78.4) on Pascal VOC07 and **0.7%** (35.16 $\rightarrow$ 35.84) on MS-COCO.

D. Additional ablations

ITERATIONS (i)	0	10	20	50	100	200	500	1000
AlignMixup	80.98	80.96	81.31	81.42	81.71	81.50	81.34	81.28

Table 10. *Ablation* of the number of iterations in Sinkhorn-Knopp algorithm using R-18 on CIFAR-100. Top-1 classification accuracy(%): higher is better.

Iterations in Sinkhorn-Knopp The default number of iterations for the Sinkhorn-Knopp algorithm in solving (8) is $i = 100$. Here, we investigate more choices, as shown in Table 10. The case of $i = 0$ is similar to cross-attention. In this case, we only normalize either the rows or columns in (7) once, such that $P\mathbf{1} = \mathbf{1}/r$ (when $\mathbf{A}$ aligned to $\mathbf{A}'$) or $P^\top \mathbf{1} = \mathbf{1}/r$ (when $\mathbf{A}'$ aligned to $\mathbf{A}$). We observe that while AlignMixup outperforms the best baseline–StyleCutMix (80.66)–in all cases, it performs best for $i = 100$ iterations.