AlignMix: Improving representation by interpolating aligned features

Shashanka Venkataramanan Yannis Avrithis Ewa Kijak Laurent Amsaleg

Inria, Univ Rennes, CNRS, IRISA

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. Many recent mixup methods focus on cutting and pasting two or more objects into one image, which is more about efficient processing than interpolation. However, how to best interpolate images is not well defined. In this sense, mixup has been connected to autoencoders, because often autoencoders "interpolate well", for instance generating an image that continuously deforms into another.

In this work, we revisit mixup from the interpolation perspective and introduce AlignMix, 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 gives rise to a situation where mixup retains mostly the geometry or pose of one image and the texture of the other, connecting it to style transfer. More than that, we show that an autoencoder can still improve representation learning under mixup, without the classifier ever seeing decoded images. AlignMix outperforms state-of-the-art mixup methods on five different benchmarks.

1. Introduction

Data augmentation [8, 31, 39] is a powerful regularization method that increases the amount and diversity of data, be it labeled or unlabeled [14]. It improves the generalization performance and helps learning invariance [44] 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 [61] and sensitivity to adversarial examples [48].

Mixup operates on two or more examples at a time, *interpolating* between them in the input [63] or feature [52] space, while also interpolating between target labels for image classification. This allows for instance to flatten class representations as well as reduce overly confident incorrect predictions and smooth decision boundaries far away from training data. However, input mixup images are overlays and tend to be unnatural [59], which raises the question: *what is a good interpolation of images?*

Figure 1: Different mixup methods. Our AlignMix retains the pose from image 1 and texture from image 2, which is different from overlay (Input and Manifold mixup) or combination of two objects (CutMix). Manifold mixup and AlignMix for visualization only, not used at training.

Often, *autoencoders* "interpolate well" [4]: Mixing in the latent space and decoding may reveal that an autoencoder has uncovered some structure about the data and captured it in the latent space. For instance, one image may deform into another, in a continuous way. However, improving such properties beyond tiny datasets is hard, while learning on generated images is not very successful [33], presumably because of their of low quality.

Interestingly, recent mixup methods focus of combining two [27, 59] or more [26] objects from different images into one, in the input space. However, this can hardly be seen as interpolation. Just before pooling, a convolutional network will extract features separately for each object. Then, pool-

ing amounts to learning that all objects are present. Hence, the argument remains that such methods make efficient use of training pixels, which is not about interpolating.

In this work, motivated by the idea of deformation as a natural way of interpolating images, we investigate geometric *alignment* for mixup. 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.

We further improve representation learning by using an autoencoder in the mixup process. Interestingly, decoding the mixed examples of aligned tensors reveals that we retain the *geometry* or *pose* of the image where we keep the coordinates and the *appearance* or *texture* of the other. This is similar to *style transfer* [15] or *image-to-image translation* [32]. Figure 1 illustrates that our method, called *AlignMix*, retains the *pose* from image 1 and *texture* from image 2, which is different than overlay or combination of two objects. However, we *do not* decode and we are not concerned about the quality of generated images. We are not concerned about the interpolation properties of the autoencoder either, because of our explicit tensor alignment.

We make the following contributions:

1. We introduce a novel mixup operation, called *AlignMix*, advocating interpolation of local structure over packing objects into images (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* [9].
2. We show that a *vanilla autoencoder* can further improve representation learning under mixup training, without the classifier ever seeing decoded clean or mixed images (subsection 3.1).
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 Zhang *et al*. [63], concurrently with similar methods [25, 50], introduce *mixup*, augmenting data by linear interpolation between two examples. While [63] includes attempts to apply mixup on intermediate representations of the network, it is Verma *et al*. [52] who make this work, introducing *manifold mixup*. Without alignment, the result is an overlay of either images [63] or features [52]. Guo *et al*. [20] eliminate "manifold intrusion"—mixed data conflicting with true data. Nonlinear mixing over random image regions is an alternative, *e.g*. from masking square regions [12] to cutting a rectangular region from one image and pasting it onto another [59], as well as several variants using arbitrary regions [22, 47, 49]. Instead of choosing regions at random, *saliency* can be used to locate objects from different images and fit them in one [26, 27, 40, 51].

Cutting and/or pasting regions may improve object localization [12, 59], while the use of saliency certainly improves efficiency by packing more than one object in the same image [26, 27]. However, we argue than none of the ideas in [26, 27, 40, 51, 59] is about *interpolation*, which is the underlying principle of mixup. We combine regional operation with linear interpolation by aligning feature tensors and then interpolating matching features separately.

Autoencoders Traversing along the manifold of representations obtained from deeper layers of the network more likely results in finding realistic examples than in input space [3]. Moreover, autoencoders often "interpolate well" [4], indicating that they may better disentangle the underlying factors of variation. One of the first applications of this insight for data augmentation is the use of transformations in the feature space, including interpolation and extrapolation, then decoding into images [11].

Many efforts have followed on mixing latent representations of autoencoders to generate realistic images for data augmentation. They can be successful as adversarial defenses [37], for example. However, this approach is more expensive, requiring three networks (encoding, decoding, classifying) and more complex, often also requiring an adversarial discriminator [2, 34]. They also perform poorly compared to standard mixup on large datasets [34], presumably due to the low quality of generated images at high resolution. One interesting alternative is Automix [66], which employs a U-Net rather than an autoencoder, mixing at several layers. Still, it is limited to small datasets and provides little improvement over manifold mixup [52].

Most state of the art mixup methods still operate on image space. We revisit mixup in the feature space with the aid of an autoencoder, without particular efforts to improve its interpolation properties or the quality of generated images. In fact, we show that a vanilla autoencoder improves representation learning when used with mixup, without the classifier seeing any decoded images.

Alignment Local correspondences from intra-class alignment of feature tensors have been used in *image registration* [7, 35], *optical flow* [55], *semantic alignment* [21, 41] and *image retrieval* [45]. 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 [13] and DeepEMD [62]. The *earth mover's distance* (EMD) [42], or

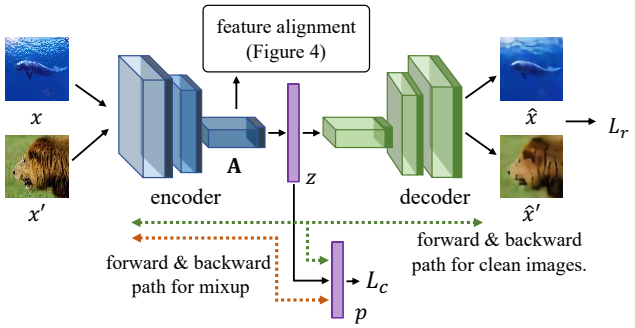

Figure 2: *Autoencoder mixup*. An image x is mapped to $c \times w \times h$ *feature tensor* $\mathbf{A} := E(x)$ and *latent vector* $z := e(\mathbf{A}) \in \mathbb{R}^d$, then decoded into image $\hat{x}$. The latent vector z is classified as $p := g(z)$. During training, we interpolate two images x, x', their feature tensors $\mathbf{A}, \mathbf{A}'$ or their latent vectors z, z' before classifying (7), while a reconstruction loss (1) applies only to clean examples. Interpolation of *aligned* tensors is shown in Figure 4.

Wasserstein metric, is an instance of *optimal transport* [53], addressed by linear programming. To accelerate, Cuturi [9] computes optimal matching by *Sinkhorn distance* with *entropic regularization*. This distance is widely applied between distributions in generative models [16, 38].

EMD has been used for mixup in the input space, for instance *point mixup* for 3D point clouds [5] and OptTransMix for images [66], 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 [9].

3. AlignMix

3.1. Latent mixup using autoencoder

Autoencoder Consider an autoencoder (AE) architecture with a classifier in the latent space, as shown in Figure 2. 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. The *encoder* consists of two stages. The first is $E : \mathcal{X} \to \mathbb{R}^{c \times w \times h}$, which maps x to feature tensor $\mathbf{A} := E(x)$, where c is the number of channels and $w \times h$ is the spatial resolution. The second is $e : \mathbb{R}^{c \times w \times h} \to \mathbb{R}^d$, which maps tensor $\mathbf{A}$ to latent vector $z := e(\mathbf{A})$. The *decoder* $D : \mathbb{R}^d \to \mathcal{X}$ maps z back to the image space, reconstructing image $\hat{x} := D(z)$. Finally, the *classifier* $g : \mathbb{R}^d \to \mathbb{R}^k$ maps the latent vector z to the vector $p := g(z)$ of probabilities over classes.

When x is a *clean* example, that is, without mixup, we apply a *reconstruction* loss L_r between x and $\hat{x}$ and a *classification* loss L_c between the prediction p and the label y:

$$L_r(x, \hat{x}) + L_c(g(e(E(x))), y), \tag{1}$$

where $L_r(x, x') := \|x - x'\|^2$ is the squared Euclidean distance and $L_c(p, y) := -\sum_{i=1}^k y_i \log p_i$ is cross-entropy.

Mixup We follow [52] in mixing the representations from different layers of the network, focusing on the deepest layers at or near the latent space. 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 $\mathrm{Beta}(\alpha, \alpha)$ [63] and then we interpolate labels y, y' linearly by the *standard* mixup operator

$$\mathrm{mix}_\lambda(y, y') := \lambda y + (1 - \lambda) y' \tag{2}$$

and inputs x, x' by the generic formula

$$\mathrm{Mix}_\lambda^{f_1, f_2}(x, x') := f_2(\mathrm{Mix}_\lambda(f_1(x), f_1(x')), \tag{3}$$

where Mix_λ is a mixup operator to be defined. This generic formula allows interpolation of the input, feature or latent representations by decomposing the encoder mapping $e \circ E$ as $f_2 \circ f_1$ according to

$$\text{input } (x): \quad f_1 := \mathrm{id}, f_2 := e \circ E \tag{4}$$

$$\text{feature } (\mathbf{A}): \quad f_1 := E, f_2 := e \tag{5}$$

$$\text{latent } (z): \quad f_1 := e \circ E, f_2 := \mathrm{id}, \tag{6}$$

where id is the identity mapping. For (4) and (6), we define Mix_λ in (3) as standard mixup mix_λ (2), like *input* [63] and *manifold mixup* [52], respectively; while for (5), we define Mix_λ as discussed in subsection 3.2.

The loss function for the *mixed* examples is

$$L_c(g(\mathrm{Mix}_\lambda^{f_1, f_2}(x, x'))), \mathrm{mix}_\lambda(y, y')). \tag{7}$$

We do not apply the reconstruction loss on mixed examples, because it is not clear what would be the input image x and target image $\hat{x}$ in (1).

Optimization We are given a training set $\{(x_i, y_i)\}_{i=1}^N$ of N examples, where image $x_i \in \mathcal{X}$ and label $y_i \in Y$. We process data in mini-batches, where in each mini-batch we either use clean images only and apply both the reconstruction and classification loss (1) per example (x_i, y_i), or we mix over pairs and apply the classification loss (7) per pair $((x_i, y_i), (x_j, y_j))$. As shown in Figure 2, the reconstruction loss L_r results in updating the parameters of E, e, D, while the classification loss L_c in updating the parameters of E, e, g. We alternate between the two and learn all parameters jointly.

Example The decoder with its reconstruction loss is meant to generate images that "interpolate well". For instance, we may expect images generated from $\mathrm{Mix}_\lambda^{f_1, f_2}(x, x')$ to look *natural* and, by varying λ from 1

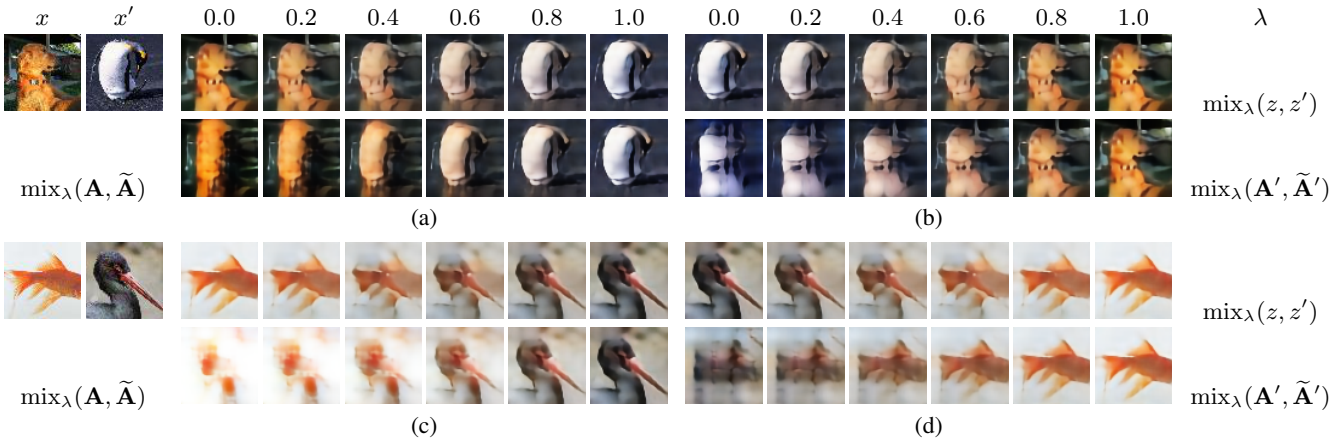

Figure 3: *Decoding of interpolated examples*. For different $\lambda \in [0, 1]$, we interpolate the latent vectors z, z' (3) (top) or aligned feature tensors $\mathbf{A}, \mathbf{A}'$ (bottom) of two images x, x' and then we generate a new image by decoding the resulting latent vector through the decoder D. (a), (c) We align $\mathbf{A}$ to $\mathbf{A}'$ and mix with (13). (b), (d) We align $\mathbf{A}'$ to $\mathbf{A}$ and mix with (14). Only meant for illustration: No decoded images are seen by the classifier at training.

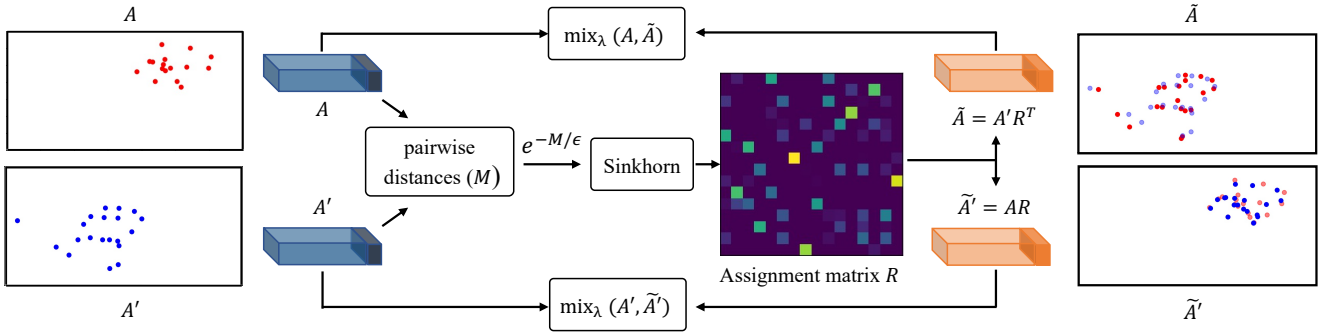

Figure 4: *Feature tensor alignment and interpolation*. Cost matrix M contains pairwise distances of feature vectors in tensors $\mathbf{A}, \mathbf{A}'$. Assignment matrix R is obtained by Sinkhorn-Knopp [29] on similarity matrix $e^{-M/\epsilon}$. $\mathbf{A}$ is aligned to $\mathbf{A}'$ according 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 $\mathbf{A}', \widetilde{\mathbf{A}}'$. Original $\mathbf{A}, \mathbf{A}'$ on the left (toy example of 16 points in 2D) shown semi-transparent on the right for reference.

to 0, to give the impression of an image continuously *deforming* from x to x'. *Does this happen in practice?*

To investigate this, we mix the latent vectors of two images x, x' by (3) and generate a new image by decoding, for different λ. As shown in Figure 3, this approach does *not* interpolate well. Instead, mixing appears to be element-wise as in input mixup [63] and the generated image appears to be an overlay of x, x'. Rather than attempting to improve the autoencoder to learn how to interpolate, we focus on interpolating the feature tensors and explicitly *align* them, as discussed next.

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 are close to the classifier—the second encoder e is small. Importantly, we are not attempting to pack two objects into one image [27], 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.

Alignment is based on *optimal transport* theory [53] and *Sinkhorn distance* (SD) [9] in particular. Let $\mathbf{A} := E(x), \mathbf{A}' := E(x')$ 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 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:

$$m_{ij} := \left\|a_i - a'_j\right\|^2 \tag{8}$$

for $i, j \in \{1, \dots, r\}$.

What we are looking for is a *transport plan*, that is, a $r \times r$ matrix $P \in U_r$, where

$$U_r := \{P \in \mathbb{R}_+^{r \times r} : P\mathbf{1} = P^\top \mathbf{1} = \mathbf{1}/r\} \tag{9}$$

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:

$$P^* = \arg\min_{P \in U_r} \langle P, M \rangle - \epsilon H(P), \tag{10}$$

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 [29], *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:

$$\widetilde{A} := A'R^\top \tag{11}$$

$$\widetilde{A}' := AR. \tag{12}$$

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 $\widetilde{\mathbf{A}}$ 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 $\mathbf{A}$:

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* $\mathbf{A}'$. As shown in Figure 4 (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 $\mathbf{A}$:

$$\mathrm{mix}_\lambda(\mathbf{A}, \widetilde{\mathbf{A}}). \tag{13}$$

Symmetrically, we can also *align* $\mathbf{A}'$ *to* $\mathbf{A}$ and interpolate between $\widetilde{\mathbf{A}}'$ and $\mathbf{A}'$:

$$\mathrm{mix}_\lambda(\mathbf{A}', \widetilde{\mathbf{A}}'). \tag{14}$$

When mixing feature tensors with alignment (5), we define Mix_λ in (3) as the mapping of $(\mathbf{A}, \mathbf{A}')$ to either (13) or (14), chosen at random.

Example In Figure 3, we visualize images generated from the decoder after mixing the aligned feature tensors. Interestingly, by aligning $\mathbf{A}$ to $\mathbf{A}'$ and mixing using (13) 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 'dog' and x' is 'penguin', 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 (14) with $\lambda = 0$, the generated image retains the pose of x and the texture of x'.

4. Experimental Results

Architecture We use a residual network as the stage 1 encoder E and a residual generator [18] as the decoder D. The encoder and decoder have the same architecture. The output $\mathbf{A}$ of the encoder is a $c \times 4 \times 4$ tensor, where c depends on the encoder. This is followed by a fully-connected layer as stage 2 encoder e for $z \in \mathbb{R}^c$ and another fully-connected layer as classifier g.

Mini-batch sampling For a given mini-batch during training, we mix either x, $\mathbf{A}$ (using either (13) or (14) for alignment) or z, or use clean images without mixup. We choose between the five cases uniformly at random.

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

4.1. Image classification and robustness

We use PreActResnet18 [23] (R-18) and WRN16-8 [60] as the encoder E on CIFAR-10 and CIFAR-100 datasets [30]. Using our experimental settings (in supplementary material), we reproduce the state-of-the-art (SOTA) mixup methods: Baseline network (without mixup), Input mixup [63], Manifold mixup [52], CutMix [59], PuzzleMix [27] and Co-Mixup [26] using official code provided by the authors. We do not compare AlignMix with AutoMix [66], since its experimental settings are different from ours and there is no available code.

In addition, we use R-18 as E on TinyImagenet [57] (TI), following the experimental settings of [26], and Resnet-50 (R-50) as E on ImageNet [43], following the training protocol of [27]. Using top-1 error (%) as evaluation metric, we show the effectiveness of AlignMix on image classification and robustness to FGSM [17] and PGD [36] attacks.

Image classification As shown in Table 1, AlignMix is on par or outperforms the SOTA methods by achieving the lowest top-1 error, especially on large datasets. On CIFAR-

DATASET	CIFAR-10		CIFAR-100		TI	IMNET
NETWORK	R-18	W16-8	R-18	W16-8	R-18	R-50
Baseline	5.19	5.11	23.24	20.63	43.40	23.68
Input [63]	4.03	3.98	20.21	19.88	43.48	22.58
CutMix [59]	3.27	3.54	19.37	19.71	43.11	21.40
Manifold [52]	2.95	3.56	19.80	19.23	40.76	22.50
PuzzleMix [27]	2.93	**2.99**	20.01	19.25	36.52	21.24
Co-Mixup [26]	2.89	3.04	19.81	19.57	35.85	–
AlignMix (ours)	**2.83**	3.15	**17.82**	**18.09**	**32.73**	**18.83**
Gain	**+0.06**	**-0.16**	**+1.55**	**+1.14**	**+3.12**	**+2.41**

Table 1: *Image classification* top-1 error (%): lower is better. Blue: second best. Gain: reduction of error. TI: Tiny-Imagenet, ImNet: ImageNet; R: PreActResnet, W: WRN.

10, AlignMix is on par with Co-Mixup and Puzzlemix with R-18 and WRN16-8. On CIFAR-100, it outperforms CutMix and Manifold mixup by 1.55% and 1.14% with R-18 and WRN16-8, respectively. On TI, using the top-1 error (%) reported for competitors by [26], AlignMix outperforms Co-Mixup by 3.12% using R-18. On Imagenet, using the top-1 error (%) reported for competitors by [27], AlignMix outperforms PuzzleMix by 2.41%. Importantly, while the overall improvement by SOTA methods on ImageNet over Baseline is around 2%, AlignMix improves SOTA by another 2.5%.

Robustness to FGSM and PGD attacks Following the evaluation protocol of [27], 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 on TI for FGSM are as reported in [27].

As shown in Table 2, AlignMix is more robust comparing to SOTA methods. While AlignMix is on par with PuzzleMix and Co-Mixup on CIFAR-10 image classification, it outperforms Co-Mixup and PuzzleMix by 8.06% and 8.98% in terms of robustness to FGSM attacks. There is also significant gain of robustness to FGSM on Tiny-ImageNet and to the stronger PGD on CIFAR-100.

4.2. 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 [24], *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 AlignMix with SOTA methods trained using R-18 on CIFAR-100 as discussed in section 4.1. At inference, ID examples are test images from CIFAR-100, while OOD examples are test images from LSUN (crop) [58], iSUN [56] and Tiny-ImageNet (crop); where crop denotes that the OOD examples are center-cropped to 32×32 to match the resolution of ID images [59]. Following [24], 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 3(left), AlignMix 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 [59]. More results are in supplementary material.

Calibration According to [10], 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.

We evaluate the calibration of AlignMix against SOTA methods in terms of *expected calibration error* (ECE) [19] using R-18 on CIFAR-100. As shown in Figure 5, while SOTA methods are under-confident compared to Baseline, AlignMix results in the best calibration among all competitors. Table 3(right) confirms this result, indicating that AlignMix has the lowest ECE.

4.3. 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 AlignMix using the same procedure as for image classification. At inference, following [59], we compute a saliency map using CAM [65], binarize it using a threshold of 0.15 and take the bounding box of the mask. We use VGG-GAP [46] and Resnet-50 [23] as pretrained on Imagenet [43] and we fine-tune them on CUB200-2011 [54]. Using top-1 localization accuracy with IoU threshold of 0.5, we compare AlignMix with baseline CAM (without mixup), Input mixup [63], CutOut [12] and CutMix [59].

According to Table 4, AlignMix outperforms Input mixup, CutOut [12] and CutMix by 11.98%, 8.88% and 1.18% respectively using VGG-GAP and by 8.5%, 5.02% and 3% respectively using Resnet-50. It also outperforms dedicated WSOL methods ACoL [64] and ADL [6], which

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 [63]	78.42	79.21	81.30	67.33	88.68	99.77	99.43	99.96	99.37
CutMix [59]	77.72	78.33	86.96	60.16	88.68	99.82	98.10	98.67	97.98
Manifold [52]	77.63	76.11	80.29	56.45	89.25	97.22	98.49	99.66	98.43
PuzzleMix [27]	41.11	50.73	78.70	57.77	83.91	97.73	97.00	96.42	95.28
Co-Mixup [26]	40.19	48.93	77.61	56.59	–	97.59	96.19	95.35	94.23
AlignMix (ours)	**32.13**	**44.86**	**76.40**	**55.44**	**78.98**	**97.16**	**95.32**	**93.69**	**92.23**
Gain	**+8.06**	**+4.07**	**+1.21**	**+1.01**	**+4.93**	**+0.06**	**+0.87**	**+1.66**	**+2.00**

Table 2: *Robustness to FGSM & PGD attacks*. Top-1 error (%): lower is better. Blue: second best. Gain: reduction of error. TI: TinyImagenet. R: PreActResnet, W: WRN.

Task	Out-Of-Distribution Detection												Calibration
Dataset	LSUN (crop)				iSUN				TI (crop)				Cifar-100
Metric	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	Det Acc	AuROC	AuPR (ID)	AuPR (OOD)	ECE
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	10.25
Input [63]	57.5	59.3	61.4	55.2	59.6	63.0	60.2	63.4	58.7	62.8	63.0	62.1	18.50
Cutmix [59]	63.8	63.1	61.9	63.4	67.0	76.3	81.0	77.7	70.4	84.3	87.1	80.6	7.60
Manifold [52]	58.9	60.3	57.8	59.5	64.7	73.1	80.7	76.0	67.4	69.9	69.3	70.5	18.41
PuzzleMix [27]	64.3	69.1	80.6	73.7	73.9	77.2	79.3	71.1	71.8	76.2	78.2	81.9	8.22
Co-Mixup [26]	70.4	75.6	82.3	70.3	68.6	80.1	82.5	75.4	71.5	84.8	86.1	80.5	5.83
AlignMix (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**	**5.06**
Gain	**+6.5**	**+7.9**	**+4.4**	**+5.7**	**+1.7**	**+4.0**	**+3.4**	**+4.0**	**+7.9**	**+3.2**	**+2.6**	**+3.8**	**+0.77**

Table 3: *Overconfidence: OOD detection* (left) and *calibration* (right) using PreActResnet18. Det Acc (detection accuracy), AuROC, AuPR (ID) and AuPR (OOD): higher is better; ECE: expected calibration error: lower is better. Blue: second best. Gain: increase in performance. TI: TinyImagenet.

Network	VGG-GAP	ResNet-50
ACoL [64]	45.92	–
ADL [6]	52.36	–
Baseline CAM [65]	37.12	49.41
Input [63]	41.73	49.30
Cutout [12]	44.83	52.78
CutMix [59]	52.53	54.81
AlignMix (ours)	**53.71**	**57.80**
Gain	**+1.18**	**+2.99**

Table 4: *Weakly-supervised object localization* on CUB200-2011. Top-1 localization accuracy (%): higher is better. Blue: second best. Gain: increase in accuracy.

focus on learning spatially dispersed representations. Qualitative localization results shown in Figure 6 illustrate that AlignMix encodes semantically discriminative representations, resulting in better localization performance.

4.4. Ablation study

All ablations are performed on CIFAR-100 using R-18 as stage 1 encoder E with feature tensor $\mathbf{A}$ being $512 \times 4 \times 4$ and latent vector $z \in \mathbb{R}^{512}$. We study the effect of mixing different layers (x, $\mathbf{A}$ or z), aligning $\mathbf{A}$ or not before mixing, the resolution used when aligning, as well as the autoencoder architecture. The latter includes a *vanilla autoencoder* (AlignMix/AE), a *variational autoencoder* [28] (AlignMix/VAE) and *no decoder* (AlignMix/–). We report top-1 accuracy (%). All results are shown in Table 5.

Layers In general, we may mix any layer in $\{x, \mathbf{A}, z\}$ in a given iteration. We ablate the effect of allowing only a particular subset of layers. In general, $z \in \mathbb{R}^{512}$ is a vector. Here, we also consider the case where z is a $128 \times 2 \times 2$ tensor, denoted as $\mathbf{Z}$ and obtained from $\mathbf{A}$ by a convolutional layer of kernel size 2×2 and stride 2. In AlignMix/AE ar-

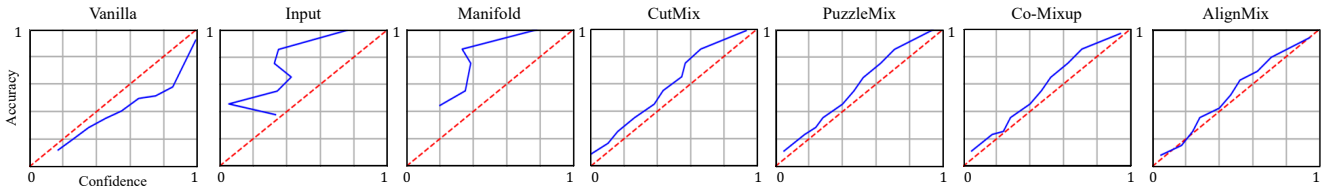

Figure 5: *Calibration* plots on CIFAR-100 using PreActResnet18: near diagonal is better. Baseline is clearly overconfident while Input and Manifold mixup are clearly under-confident. AlignMix has the best calibrated predictions.

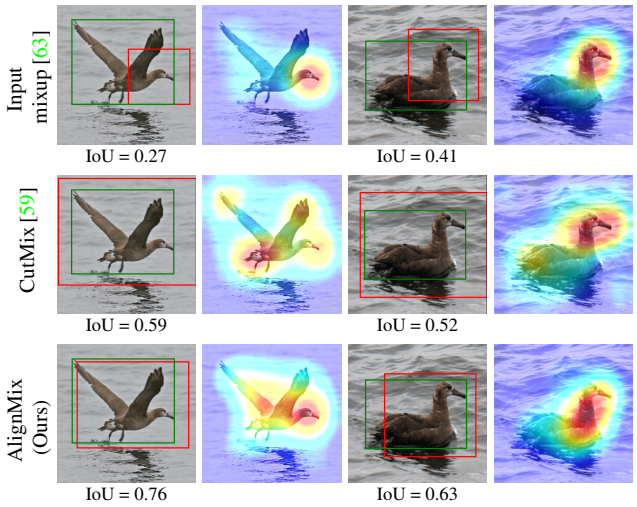

Figure 6: *Localization examples* using ResNet-50 on CUB200-2011. Red boxes: predicted; green: ground truth.

METHOD/ARCH	LAYERS	UNALIGNED	ALIGNED
Baseline		76.76	–
Manifold [52]		80.20	-
CutMix [59]		80.63	-
AlignMix/AE	$\{x, z\}$	81.92	-
	$\{x, \mathbf{A}\}$	81.78	81.85
	$\{x, \mathbf{Z}\}$	80.80	81.54
	$\{x, \mathbf{A}, z\}$	81.61	**82.18**
AlignMix/AE	$\{x, \mathbf{A}_{2\times2}, z\}$	81.47	81.20
	$\{x, \mathbf{A}_{4\times4}, z\}$	81.61	82.18
	$\{x, \mathbf{A}_{8\times8}, z\}$	80.49	**82.20**
AlignMix/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**
AlignMix/–	$\{x, z\}$	80.81	–
	$\{x, \mathbf{A}\}$	80.34	81.61
	$\{x, \mathbf{A}, \mathbf{Z}\}$	80.46	81.36
	$\{x, \mathbf{A}, z\}$	80.33	**81.77**

Table 5: *Ablation study* using R-18 on CIFAR-100. Top-1 classification accuracy (%): higher is better. Arch: autoencoder architecture. AE: vanilla; VAE: variational [28]; –: no decoder. Layer x, $\mathbf{A}$, z: (4), (5), (6).

chitecture, among different choices of unaligned layer sets, mixing from $\{x, z\}$ results in the highest classification accuracy. Furthermore, AlignMix/AE outperforms Baseline and the best performing competitor CutMix for all choices of layer sets, even when features are unaligned.

Tensor alignment We ablate the effect of aligning feature tensor $\mathbf{A}$ or not before mixing it, by using standard mixup (3) or (13), (14), respectively. In AlignMix/AE architecture, we observe that aligning $\mathbf{A}$ before mixing improves classification accuracy significantly. It is important to note that when z is a vector, we do not align it. However, when it is a tensor $\mathbf{Z}$, aligning it improves significantly. Overall, AlignMix/AE works the best when $x, \mathbf{A}, z$ are mixed, with $\mathbf{A}$ being aligned. This setting outperforms CutMix by 1.55%. Mixing $\mathbf{A}$ only helps when it is aligned; otherwise, it is preferable to just mix z.

Alignment resolution We ablate the effect of aligning $\mathbf{A}$ at different spatial resolutions. The default is 4×4, denoted as $\mathbf{A}_{4\times4}$. Here, we investigate 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 investigate two more autoencoder architectures, AlignMix/VAE and AlignMix/–. The former has two vectors $\mu, \sigma \in \mathbb{R}^{512}$ instead of z, representing mean and standard deviation, respectively. We also investigate $128 \times 2 \times 2$ tensors, denoted as $\mathbf{M}, \mathbf{\Sigma}$. In both cases, the two variables are mixed simultaneously. As for AlignMix/AE, we investigate different combinations of layers with or without alignment. Both architectures are inferior to AlignMix/AE. However, their best setting still outperforms Baseline and Cutmix. All three architecture work best when $x, \mathbf{A}, z$ are mixed. Alignment improves consistently on all three architectures.

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. There is a clear message in favor of alignment, rather than packing, of objects.

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.

References

[1] David Alvarez-Melis and Tommi S Jaakkola. Gromov-wasserstein alignment of word embedding spaces. In *Conference on Empirical Methods in Natural Language Processing*, 2018. 5

[2] Christopher Beckham, Sina Honari, Vikas Verma, Alex Lamb, Farnoosh Ghadiri, R Devon Hjelm, Yoshua Bengio, and Christopher Pal. On adversarial mixup resynthesis. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. 2

[3] Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In *International Conference on Machine Learning (ICML)*, pages 552–560, 2013. 2

[4] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543*, 2018. 1, 2

[5] Yunlu Chen, Vincent Tao Hu, Efstratios Gavves, Thomas Mensink, Pascal Mettes, Pengwan Yang, and Cees GM Snoek. Pointmixup: Augmentation for point clouds. *ECCV*, 2020. 3

[6] Junsuk Choe and Hyunjung Shim. Attention-based dropout layer for weakly supervised object localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2219–2228, 2019. 6, 7

[7] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 30, 2016. 2

[8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. AutoAugment: Learning augmentation strategies from data. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 113–123, 2019. 1

[9] Marco Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 2, page 4, 2013. 2, 3, 4

[10] Morris H DeGroot and Stephen E Fienberg. The comparison and evaluation of forecasters. *Journal of the Royal Statistical Society: Series D (The Statistician)*, 32(1-2):12–22, 1983. 6

[11] Terrance DeVries and Graham W Taylor. Dataset augmentation in feature space. *arXiv preprint arXiv:1702.05538*, 2017. 2

[12] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 2, 6, 7

[13] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 33, 2020. 2

[14] Alexey Dosovitskiy, Jost Tobias Springenberg, and Thomas Brox. Unsupervised feature learning by augmenting single images. *arXiv preprint arXiv:1312.5242*, 2013. 1

[15] Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. Image style transfer using convolutional neural networks. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 2

[16] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, pages 1608–1617. PMLR, 2018. 3

[17] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *International Conference on Learning Representations (ICLR)*, 2015. 5

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

[19] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, pages 1321–1330. PMLR, 2017. 6

[20] Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *AAAI Conference on Artificial Intelligence*, volume 33, pages 3714–3722, 2019. 2

[21] Kai Han, Rafael S Rezende, Bumsub Ham, Kwan-Yee K Wong, Minsu Cho, Cordelia Schmid, and Jean Ponce. Scnet: Learning semantic correspondence. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1831–1840, 2017. 2

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

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. 5, 6

[24] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. *International Conference on Learning Representations (ICLR)*, 2017. 6, 2

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

[26] Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-mixup: Saliency guided joint mixup with supermodular diversity. In *International Conference on Learning*

Representations (ICLR), 2021. 1, 2, 5, 6, 7

[27] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning (ICML)*, pages 5275–5285. PMLR, 2020. 1, 2, 4, 5, 6, 7

[28] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 7, 8

[29] Philip A Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 2008. 4, 5, 1

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

[31] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems (NeurIPS)*, 25:1097–1105, 2012. 1

[32] Ming-Yu Liu, Xun Huang, Arun Mallya, Tero Karras, Timo Aila, Jaakko Lehtinen, and Jan Kautz. Few-shot unsupervised image-to-image translation. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 2

[33] 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 *International Conference on Pattern Recognition (ICPR)*, 2018. 1

[34] 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 *International Conference on Pattern Recognition (ICPR)*, pages 728–733. IEEE, 2018. 2

[35] Jonathan Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In *International Conference on Neural Information Processing Systems (NIPS)*, volume 1, page 1601–1609, 2014. 2

[36] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *International Conference on Learning Representations (ICLR)*, 2018. 5

[37] Puneet Mangla, Vedant Singh, Shreyas Jayant Havaldar, and Vineeth N Balasubramanian. Varmixup: Exploiting the latent space for robust training and inference. *arXiv preprint arXiv:2003.06566*, 2020. 2

[38] 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*, pages 733–743. PMLR, 2020. 3

[39] Mattis Paulin, Jérôme Revaud, Zaid Harchaoui, Florent Perronnin, and Cordelia Schmid. Transformation pursuit for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3646–3653, 2014. 1

[40] 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

[41] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. End-to-end weakly-supervised semantic alignment. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 6917–6925, 2018. 2

[42] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000. 2

[43] 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. *International journal of computer vision*, 115(3):211–252, 2015. 5, 6

[44] 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*, pages 239–274. Springer, 1998. 1

[45] Oriane Siméoni, Yannis Avrithis, and Ondrej Chum. Local features and visual words emerge in activations. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 11651–11660, 2019. 2

[46] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations (ICLR)*, 2015. 6

[47] Cecilia Summers and Michael J Dinneen. Improved mixed-example data augmentation. In *Winter Conference on Applications of Computer Vision (WACV)*, 2019. 2

[48] Christian Szegedy, Wojciech Zaremba, Ilya Sutskever, Joan Bruna, Dumitru Erhan, Ian Goodfellow, and Rob Fergus. Intriguing properties of neural networks. In *International Conference on Learning Representations (ICLR)*, 2014. 1

[49] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *Asian Conference on Machine Learning (ACML)*, 2018. 2

[50] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition. In *International Conference on Learning Representations (ICLR)*, 2018. 2

[51] 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 *International Conference on Machine Learning (ICML)*, 2021. 2

[52] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning (ICML)*, pages 6438–6447, 2019. 1, 2, 3, 5, 6, 7, 8

[53] Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008. 3, 4

[54] 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. 6

[55] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 1385–1392, 2013. 2

[56] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3485–3492. IEEE, 2010. 6

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

[58] 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, 2

[59] 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 *IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6023–6032, 2019. 1, 2, 5, 6, 7, 8

[60] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In *The British Machine Vision Conference (BMVC)*, 2016. 5

[61] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations (ICLR)*, 2017. 1

[62] Chi Zhang, Yujun Cai, Guosheng Lin, and Chunhua Shen. Deepemd: Few-shot image classification with differentiable earth mover's distance and structured classifiers. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 12203–12213, 2020. 2

[63] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018. 1, 2, 3, 4, 5, 6, 7, 8

[64] Xiaolin Zhang, Yunchao Wei, Jiashi Feng, Yi Yang, and Thomas S Huang. Adversarial complementary learning for weakly supervised object localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1325–1334, 2018. 6, 7

[65] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2921–2929, 2016. 6, 7, 1

[66] Jianchao Zhu, Liangliang Shi, Junchi Yan, and Hongyuan Zha. Automix: Mixup networks for sample interpolation via cooperative barycenter learning. In *European Conference on Computer Vision (ECCV)*, pages 633–649. Springer, 2020. 2, 3, 5

A. Algorithm

AlignMix is summarized in algorithm 1. For each mini-batch, we uniformly draw at random a choice (line 2) over clean examples or mixup on input (x), latent vectors (z), or feature tensors ($\mathbf{A}$, using either (13) or (14) for mixing). We use both reconstruction and classification loss (1) for clean examples (line 7). For mixup, following [52], 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 (3),(4) (line 12) and latent vectors z, z' by (3),(6) (line 14). Feature tensors $\mathbf{A}$ and $\mathbf{A}'$ are first aligned and then mixed by (3),(13) ($\mathbf{A}$ aligns to $\mathbf{A}'$) or (3),(14) ($\mathbf{A}'$ aligns to $\mathbf{A}$) (lines 17,26). We use only classification loss (7) for mixed examples (line 27).

In computing loss derivatives, we backpropagate through latent vectors z, z' or feature tensors $\mathbf{A}, \mathbf{A}'$ but not through the transport plan P^* (line 23). Hence, although the Sinkhorn-Knopp algorithm [29] is differentiable, its iterations take place only in the forward pass. Importantly, AlignMix is easy to implement and does not require sophisticated optimization like [26, 27].

Algorithm 1: AlignMix

Input: encoders E, 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* $\sim \text{unif}\{\textit{clean}, \textit{input}, \textit{latent}, \textit{feat}, \textit{feat}'\}$ ▷ 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 $z \leftarrow e(E(x)), \hat{z} \leftarrow D(z)$ ▷ encode/decode
7 $\ell_i \leftarrow L_r(x, \hat{z}) + L_c(g(z), y)$ ▷ clean loss (1)
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 [63]
12 $z \leftarrow e(E(\text{mix}_\lambda(x, x')))$ ▷ (3),(4)
13 **else if** *mode* $=$ *latent* **then** ▷ as in [52]
14 $z \leftarrow \text{mix}_\lambda(e(E((x)), e(E(x')))$ ▷ (3),(6)
15 **else** ▷ *mode* $\in \{\textit{feat}, \textit{feat}'\}$
16 **if** *mode* $=$ *feat*$'$ **then** ▷ choose (14) over (13)
17 SWAP (x, x'), SWAP (y, y')
18 $\mathbf{A} \leftarrow E(x)$, $\mathbf{A}' \leftarrow E(x')$ ▷ feature tensors
19 $A \leftarrow \text{RESHAPE}_{c \times r}(\mathbf{A})$ ▷ to matrix
20 $A' \leftarrow \text{RESHAPE}_{c \times r}(\mathbf{A}')$
21 $M \leftarrow \text{DIST}(A, A')$ ▷ pairwise distances (8)
22 $P^* \leftarrow \text{SINKHORN}(e^{-M/\epsilon})$ ▷ tran. plan (10)
23 $R \leftarrow \text{DETACH}(rP^*)$ ▷ assignments
24 $\widetilde{A} \leftarrow A'R^\top$ ▷ alignment (11)
25 $\widetilde{\mathbf{A}} \leftarrow \text{RESHAPE}_{c \times w \times h}(\widetilde{A})$ ▷ to tensor
26 $z \leftarrow e(\text{mix}_\lambda(\mathbf{A}, \widetilde{\mathbf{A}}))$ ▷ (3),(13)
27 $\ell_i \leftarrow L_c(g(z), \text{mix}_\lambda(y, y'))$ ▷ mixed loss (7)

B. Hyperparameter settings

CIFAR-10/ CIFAR-100 We train AlignMix 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 [29] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

TinyImagenet We follow the training protocol of Kim *et al*. [27], training R-18 as stage-1 encoder E 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 [29] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

ImageNet We follow the training protocol of Kim *et al*. [27], where training R-50 as E 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 [29] 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*. [26].

CUB200-2011 For weakly-supervised object localization (WSOL), we use VGG-GAP and R-50 pretrained on ImageNet as E. The training strategy for WSOL is the same as image classification and the network is trained *without bounding box information*. In R-50, following [59], 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 [65]. The classifier is modified to have 200 classes instead of 1000.

For fair comparisons with [59], during training, we resize the input image to 256×256 and randomly crop the resized image to 224×224. During testing, we directly resize to 224×224. We train the network for 600 epochs using

NETWORK	RESNET-50
Baseline	24.03
Input [63]	22.97
Manifold [52]	23.30
CutMix [59]	22.92
PuzzleMiz [27]	22.49
Co-Mixup [26]	22.39
AlignMix (ours)	**20.76**
Gain	**+1.63**

Table 6: *Image classification* on ImageNet for 100 epochs using ResNet-50. Top-1 error (%): lower is better. Blue: second best. Gain: reduction of error.

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 [63]	61.5	66.5	66.4	65.8	59.6	63.8	63.0	63.4
Cutmix [59]	71.3	77.4	79.1	75.5	69.1	79.4	79.8	73.3
Manifold [52]	67.8	78.9	76.3	71.3	62.5	77.8	76.8	72.2
PuzzleMix [27]	74.9	79.9	84.0	77.5	73.9	77.3	80.6	71.9
Co-Mixup [26]	73.8	82.6	86.8	76.9	68.1	78.9	82.5	74.2
AlignMix (ours)	**77.0**	**85.8**	**87.9**	**83.7**	**76.2**	**84.8**	**87.2**	**82.3**
Gain	**+2.1**	**+3.2**	**+1.1**	**+6.2**	**+2.3**	**+5.4**	**+4.7**	**+8.1**
NOISE	UNIFORM				GAUSSIAN			
Baseline	58.3	75.3	75.0	69.0	60.8	64.3	62.9	63.9
Input [63]	50.0	67.9	71.8	71.7	60.2	65.0	63.1	64.1
Cutmix [59]	74.8	80.0	84.9	72.4	75.7	79.0	84.0	70.9
Manifold [52]	69.8	75.9	83.2	71.9	70.8	78.8	81.3	71.6
PuzzleMix [27]	78.6	85.2	86.0	74.4	78.5	85.1	85.9	74.3
Co-Mixup [26]	80.4	87.6	87.4	75.2	81.6	78.6	89.5	74.2
AlignMix (ours)	**88.0**	**90.6**	**94.0**	**80.8**	**86.0**	**87.2**	**91.9**	**75.6**
Gain	**+7.6**	**+3.0**	**+6.6**	**+5.6**	**+4.4**	**+2.1**	**+2.4**	**+1.3**

Table 7: *OOD 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.

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 [26], Table 6 reports classification performance when training for 100 epochs on ImageNet. Using the top-1 error (%) reported for competitors by [26], AlignMix outperforms all methods, including Co-Mixup [26]. Importantly, while the overall improvement by SOTA methods over Baseline is around 1.64%, AlignMix improves SOTA by another 1.63%.

Out-of-distribution detection We compare AlignMix with SOTA methods, training R-18 on CIFAR-100 as discussed in section 4.1. At inference, ID examples are test images from CIFAR-100, while OOD examples are test images from LSUN [58] and Tiny-ImageNet, resizing OOD examples to 32×32 to match the resolution of ID images [59]. We also use test images from CIFAR-100 with 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 [24], 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 7, AlignMix outperforms SOTA methods under all metrics by a large margin, indicating that it is better in reducing over-confident predictions.