# 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] IARAI, 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* [27, 33, 7] is a powerful regularization method that increases the amount and diversity of data, be it labeled or unlabeled [12]. It improves the generalization performance and helps learning invariance [38] 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.

*Mixup* operates on two or more examples at a time, *interpolating* between them in the input space [51] or feature space [45], while also interpolating between target labels for image classification. This flattens class representations [45], 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 [49]. Interestingly, recent mixup methods focus of combining two [49, 25] or more [24] 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 [44], which raises the question: *what is a good interpolation of images?*

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

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

the underlying factors of variation. Efforts have followed on mixing latent representations of autoencoders to generate realistic images for data augmentation. However, these approaches are more expensive, requiring three networks (encoder, decoder, classifier) [4] and more complex, often also requiring an adversarial discriminator [2, 30]. More importantly, they perform poorly compared to standard input mixup on large datasets [30], 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. **??** 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.

## 2 AlignMixup

### 2.1 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 [45] 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 $\mathrm{Beta}(\alpha, \alpha)$ [51] 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 $\mathrm{Mix}_\lambda$ 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 $\mathrm{id}$ is the identity mapping. For (3), we define $\mathrm{Mix}_\lambda$ in (2) as standard mixup $\mathrm{mix}_\lambda$ (1), like [51]; while for (4), we define $\mathrm{Mix}_\lambda$ as discussed in subsection 2.2.

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 3.

### 2.2 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,

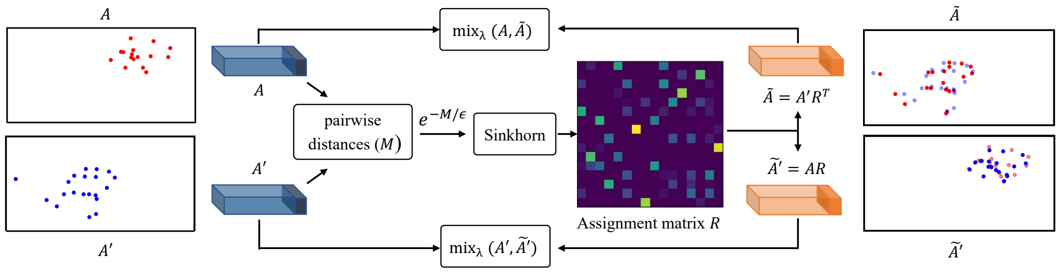


Figure 1: *Feature tensor alignment and interpolation*. Cost matrix $M$ contains pairwise distances of feature vectors in tensors $\mathbf{A}, \mathbf{A}'$. Assignment matrix $R$ is obtained by Sinkhorn-Knopp [26] on similarity matrix $e^{-M/\epsilon}$. $\mathbf{A}$ is aligned to $\mathbf{A}'$ according 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.

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 [25], 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 [46] and *Sinkhorn distance* (SD) [8] 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 $j = 1, \dots, r$ 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 $i, j \in \{1, \dots, r\}$. 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 $\epsilon$ 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 [26], *i.e.*, iteratively normalizing rows and columns. A small $\epsilon$ leads to sparser $P$, which improves one-to-one matching but makes the optimization harder [1], while a large $\epsilon$ 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 $\mathbf{A}$:

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


Figure 2: *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.

As shown in Figure 1 (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 $\mathbf{A}'$:

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

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

### 2.3 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 3.

**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 2, 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 2(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 2(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 2(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 improving its generalization and enhancing its performance across multiple benchmarks, as discussed in section 3.

| 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 [51] | 4.03 | 3.98 | 20.21 | 19.88 | 43.48* |
| CutMix [49] | 3.27 | 3.54 | 19.37 | 19.71 | 43.11* |
| Manifold [45] | 2.95 | 3.56 | 19.80 | 19.23 | 40.76* |
| PuzzleMix [25] | 2.93 | **2.99** | 20.01 | 19.25 | 36.52* |
| Co-Mixup [24] | **2.89** | 3.04 | 19.81 | 19.57 | 35.85* |
| SaliencyMix [44] | 2.99 | 3.53 | 19.69 | 19.59 | 34.81 |
| StyleMix [21] | 3.76 | 3.89 | 20.04 | 20.45 | 36.13 |
| StyleCutMix [21] | 3.06 | 3.12 | 19.34 | 19.28 | 34.49 |
| AlignMixup (ours) | 2.95 | 3.09 | **18.29** | **18.77** | **33.13** |

(a) *Image classification* top-1 error (%). TI (TinyImagenet). R: PreActResnet, W: WRN.

| Method | param. | msec/batch | top-1 error |
|---|---|---|---|
| Baseline | 25M | 418 | 23.68 |
| Input† [51] | 25M | 436 | 22.58 |
| CutMix† [49] | 25M | 427 | 21.40 |
| Manifold† [45] | 25M | 441 | 22.50 |
| PuzzleMix† [25] | 25M | 846 | 21.24 |
| Co-Mixup* [24] | 25M | 1022 | – |
| SaliencyMix* [44] | 25M | 462 | 21.26 |
| StyleMix* [21] | 25M | 828 | - |
| StyleCutMix* [21] | 25M | 912 | - |
| AlignMixup (ours) | 25M | 450 | **20.68** |

(b) *Image classification and training speed* on ImageNet. Top-1 error (%): lower is better. Speed: images/sec ($\times 10^3$): higher is better.

Table 1: *Image classification and training speed.*

| 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 [51] | 78.42 | 79.21 | 81.30 | 67.33 | 88.68 | 99.77 | 99.43 | 99.96 | 99.37 |
| CutMix [49] | 77.72 | 78.33 | 86.96 | 60.16 | 88.68 | 99.82 | 98.10 | 98.67 | 97.98 |
| Manifold [45] | 77.63 | 76.11 | 80.29 | 56.45 | 89.25 | 97.22 | 98.49 | 99.66 | 98.43 |
| PuzzleMix [25] | 57.11 | 60.73 | 78.70 | 57.77 | 83.91 | 97.73 | 97.00 | 96.42 | 95.28 |
| Co-Mixup [24] | 60.19 | 58.93 | 77.61 | 56.59 | – | 97.59 | **96.19** | 95.35 | 94.23 |
| SaliencyMix [44] | 57.43 | 68.10 | 77.79 | 58.10 | 81.16 | 97.51 | 97.04 | 95.68 | 93.76 |
| StyleMix [21] | 79.54 | 71.05 | 80.54 | 67.94 | 84.93 | 98.23 | 97.46 | 98.39 | 98.24 |
| StyleCutMix [21] | 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** |

Table 2: *Robustness to FGSM & PGD attacks*. Top-1 error (%): lower is better.TI: TinyImagenet.

## 3 Experiments

**Image classification** As shown in Table 1(a), AlignMixup is on par or outperforms the SOTA methods by achieving the lowest top-1 error, especially on large datasets. On CIFAR-10, AlignMixup is on par with Co-Mixup and Puzzlemix with R-18 and WRN16-8. On CIFAR-100, AlignMixup outperforms Manifold mixup by 1.51% and 0.46% with R-18 and WRN16-8, respectively. On TI, AlignMixup outperforms Co-Mixup by 2.72% using R-18. From Table 1(b), AlignMixup outperforms PuzzleMix by 0.56% on ImageNet.

**Robustness to FGSM and PGD attacks** Following the evaluation protocol of [25] to evaluate AlignMixup robustness to FGSM and PGD attacks As shown in Table 2, 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 Tiny-ImageNet and to the stronger PGD on CIFAR-100.

## 4 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" hand-crafted 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 *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] 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. 8

[6] Christopher B Choy, JunYoung Gwak, Silvio Savarese, and Manmohan Chandraker. Universal correspondence network. In *NeurIPS*, 2016. 8

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

[8] Marco Cuturi. Sinkhorn distances: lightspeed computation of optimal transport. In *NeurIPS*, 2013. 3, 8

[9] Ali Dabouei, Sobhan Soleymani, Fariborz Taherkhani, and Nasser M Nasrabadi. Supermix: Supervising the mixing data augmentation. In *CVPR*, 2021. 8

[10] Terrance DeVries and Graham W Taylor. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017. 8

[11] Carl Doersch, Ankush Gupta, and Andrew Zisserman. Crosstransformers: spatially-aware few-shot transfer. In *NeurIPS*, 2020. 8

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

[13] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *IJCV*, 2010. 11

[14] Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *AISTATS*, 2018. 8

[15] 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. 11

[16] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017. 11

[17] Hongyu Guo, Yongyi Mao, and Richong Zhang. Mixup as locally linear out-of-manifold regularization. In *AAAI*, 2019. 8

[18] 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. 8

[19] 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. 8

[20] Dan Hendrycks and Kevin Gimpel. A baseline for detecting misclassified and out-of-distribution examples in neural networks. In *ICLR*, 2017. 11

[21] Minui Hong, Jinwoo Choi, and Gunhee Kim. Stylemix: Separating content and style for enhanced data augmentation. In *CVPR*, 2021. 5, 8, 10, 11

[22] Xun Huang and Serge Belongie. Arbitrary style transfer in real-time with adaptive instance normalization. In *ICCV*, 2017. 8

[23] Hiroshi Inoue. Data augmentation by pairing samples for images classification. *arXiv preprint arXiv:1801.02929*, 2018. 8

[24] Jang-Hyun Kim, Wonho Choo, Hosan Jeong, and Hyun Oh Song. Co-mixup: Saliency guided joint mixup with supermodular diversity. In *ICLR*, 2021. 1, 5, 8, 9, 10, 11

[25] Jang-Hyun Kim, Wonho Choo, and Hyun Oh Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *ICML*, 2020. 1, 3, 5, 8, 9, 10, 11

[26] Philip A Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 2008. 3, 8, 9

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

[28] 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. 11

[29] 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. 11

[30] 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

[31] Jonathan Long, Ning Zhang, and Trevor Darrell. Do convnets learn correspondence? In *NIPS*, 2014. 8

[32] 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. 8

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

[34] 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. 8

[35] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, 2015. 11

[36] Ignacio Rocco, Relja Arandjelović, and Josef Sivic. End-to-end weakly-supervised semantic alignment. In *CVPR*, 2018. 8

[37] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover's distance as a metric for image retrieval. *IJCV*, 2000. 8

[38] 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

[39] Oriane Siméoni, Yannis Avrithis, and Ondrej Chum. Local features and visual words emerge in activations. In *CVPR*, 2019. 8

[40] Cecilia Summers and Michael J Dinneen. Improved mixed-example data augmentation. In *WACV*, 2019. 8

[41] Ryo Takahashi, Takashi Matsubara, and Kuniaki Uehara. Ricap: Random image cropping and patching data augmentation for deep cnns. In *ACML*, 2018. 8

[42] 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. 11

[43] Yuji Tokozume, Yoshitaka Ushiku, and Tatsuya Harada. Learning from between-class examples for deep sound recognition. In *ICLR*, 2018. 8

[44] 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, 5, 8, 10, 11

[45] 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, 5, 8, 10, 11

[46] Cédric Villani. *Optimal transport: old and new*. Springer Science & Business Media, 2008. 3, 8

[47] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *ICCV*, 2013. 8

[48] 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. 11

[49] 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, 5, 8, 9, 10, 11, 12

[50] 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. 8

[51] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 1, 2, 5, 8, 9, 10, 11, 12

[52] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. In *CVPR*, 2016. 10

[53] Jianchao Zhu, Liangliang Shi, Junchi Yan, and Hongyuan Zha. Automix: Mixup networks for sample interpolation via cooperative barycenter learning. In *ECCV*, 2020. 8

# A Related Work

**Mixup** [51], concurrently with similar methods [23, 43], introduce *mixup*, augmenting data by linear interpolation between two examples. While [51] apply mixup on intermediate representations, it is [45] who make this work, introducing *manifold mixup*. Without alignment, the result is an overlay of either images [51] or features [45]. [17] 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 [10] to cutting a rectangular region from one image and pasting it onto another [49], as well as several variants using arbitrary regions [41, 40, 19]. Instead of choosing regions at random, *saliency* can be used to locate objects from different images and fit them in one [44, 34, 25, 24]. Exploiting the knowledge of a teacher network to mix images based on saliency has been proposed in [9]. Instead of combining more than one objects in an image, AlignMixup attempts to deform one object into another.

Another alternative is Automix [53], 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 [45]. StyleMix and StyleCutMix [21] interpolate content and style between two images, using AdaIN [22], 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* [6, 31], *optical flow* [47], *semantic alignment* [36, 18] and *image retrieval* [39]. 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 [11] and DeepEMD [50]. The *earth mover's distance* (EMD) [37], or *Wasserstein metric*, is an instance of *optimal transport* [46], addressed by linear programming. To accelerate, [8] computes optimal matching by *Sinkhorn distance* with *entropic regularization*. This distance is widely applied between distributions in generative models [14, 32].

EMD has been used for mixup in the input space, for instance *point mixup* for 3D point clouds [5] and OptTransMix for images [53], 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 [8].

# B 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 [45], 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).

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 [26] is differentiable, its iterations take place only in the forward pass. Importantly, AlignMixup is easy to implement and does not require sophisticated optimization like [25, 24].

# C 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 $\mathrm{Beta}(\alpha, \alpha)$

**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 := {ℓ_i}_{i=1}^b
π ~ unif(S_b)                                          ▷ random permutation of {1, ..., b}
mode ~ unif{clean, input, feat, feat'}                 ▷ mixup?
for i ∈ {1, ..., b} do
    (x, y) ← (x_i, y_i)                                ▷ current example
    if mode = clean then                               ▷ no mixup
        x̂ ← D(e(F(x)))                                 ▷ encode/decode
        ℓ_i ← L_r(x, x̂)                                ▷ reconstruction loss
    else                                               ▷ mixup
        λ ~ Beta(α, α)                                 ▷ interpolation factor
        (x', y') ← (x_{π(i)}, y_{π(i)})                ▷ paired example
        if mode = input then                           ▷ as in [51]
            out ← F(mix_λ(x, x'))                      ▷ (2),(3)
        else                                           ▷ mode ∈ {feat, feat'}
            if mode = feat' then                       ▷ choose (12) over (11)
                SWAP (x, x'), SWAP (y, y')
            A ← F(x) , A' ← F(x')                      ▷ feature tensors
            A ← RESHAPE_{c×r}(A)                       ▷ to matrix
            A' ← RESHAPE_{c×r}(A')
            M ← DIST(A, A')                            ▷ pairwise distances (6)
            P* ← SINKHORN(exp(−M/ε))                   ▷ tran. plan (8)
            R ← DETACH(rP*)                            ▷ assignments
            Ã ← A'R^⊤                                  ▷ alignment (9)
            Ã ← RESHAPE_{c×w×h}(Ã)                     ▷ to tensor
            out ← f(mix_λ(A, Ã))                       ▷ (2),(11)
        ℓ_i ← L_c(g(out), mix_λ(y, y'))                ▷ classification loss (5)
```

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 [26] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

**TinyImagenet** We follow the training protocol of Kim *et al*. [25], 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 $\mathrm{Beta}(\alpha, \alpha)$ where $\alpha = 2.0$. For alignment, we apply the Sinkhorn-Knopp algorithm [26] for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

**ImageNet** We follow the training protocol of Kim *et al*. [25], 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 $\mathrm{Beta}(\alpha, \alpha)$ where $\alpha = 2.0$. For alignment, we apply the Sinkhorn-Knopp algorithm [26] 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*. [24].

**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 [49], we modify the last residual block (`layer 4`) to have stride 2 instead of 1, resulting in a feature map of spatial

| Network | ResNet-50 |
|---|---|
| Baseline | 24.03 |
| Input [51] | 22.97 |
| Manifold [45] | 23.30 |
| CutMix [49] | 22.92 |
| PuzzleMix [25] | 22.49 |
| Co-Mixup [24] | 22.39 |
| StyleMix [21] | 24.06 |
| StyleCutMix [21] | 22.71 |
| AlignMixup (ours) | **22.0** |
| Gain | **+0.39** |

Table 3: *Image classification* on ImageNet for 100 epochs using ResNet-50. Top-1 error (%): lower is better. Blue: second best. Gain: reduction of error.

| 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 [51] | 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 [49] | 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 [45] | 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 [25] | 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 [24] | 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 [44] | 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 [21] | 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 [21] | 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.

resolution $14 \times 14$. The modified architecture of VGG-GAP is the same as described in [52]. The classifier is modified to have 200 classes instead of 1000.

For fair comparisons with [49], during training, we resize the input image to $256 \times 256$ and randomly crop the resized image to $224 \times 224$. During testing, we directly resize to $224 \times 224$. 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.

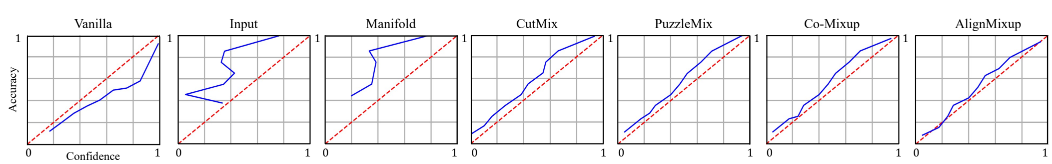


Figure 3: *Calibration plots* on CIFAR-100 using PreActResnet18: near diagonal is better. We plot accuracy *vs*. confidence, that is, probability for the predicted class.

## D Additional experiments

**ImageNet classification** Following the training protocol of [24], Table 3 reports classification performance when training for 100 epochs on ImageNet. Using the top-1 error (%) reported for

| METRIC | ECE | OE |
|---|---|---|
| Baseline | 10.25 | 1.11 |
| Input [51] | 18.50 | 1.42 |
| CutMix [49] | 7.60 | 1.05 |
| Manifold [45] | 18.41 | 0.79 |
| PuzzleMix [25] | 8.22 | 0.61 |
| Co-Mixup [24] | 5.83 | 0.55 |
| SaliencyMix [44] | 5.89 | 0.59 |
| StyleMix [21] | 11.43 | 1.31 |
| StyleCutMix [21] | 9.30 | 0.87 |
| AlignMixup (ours) | 5.78 | **0.41** |
| AlignMixup/AE (ours) | **5.06** | 0.48 |
| Gain | **+0.77** | **+0.14** |

Table 5: *Calibration* using PreActResnet18 on CIFAR-100. ECE: expected calibration error; OE: overconfidence error. Lower is better. Blue: second best. Gain: reduction of error.

competitors by [24], AlignMixup outperforms all methods, including Co-Mixup [24]. 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 [15] 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. At inference, ID examples are test images from CIFAR-100, while OOD examples are test images from LSUN [48] and Tiny-ImageNet, resizing OOD examples to $32 \times 32$ to match the resolution of ID images [49]. 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 [20], 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.

**Calibration** We compare AlignMixup with SOTA methods , training R-18 on CIFAR-100. All SOTA mixup methods are reproduced using the same experimental settings. We compare qualitatively by plotting accuracy *vs*. confidence. As shown in Figure 3, while Baseline is clearly over-confident 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) [16] and *overconfidence error* (OE) [42]. As shown in Table 5, AlignMixup outperforms SOTA methods by achieveing lower ECE and OE, indicating that it is better calibrated.

**Qualitative results of WSOL** Qualitative localization results shown in Figure 4 indicate that AlignMixup encodes semantically discriminative representations, resulting in better localization performance.

**Object detection** Following the settings of CutMix [49], we use Resnet-50 pretrained on ImageNet using AlignMixup as the backbone of SSD [29] and Faster R-CNN [35] detectors and fine-tune it on Pascal VOC07 [13] and MS-COCO [28] 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.

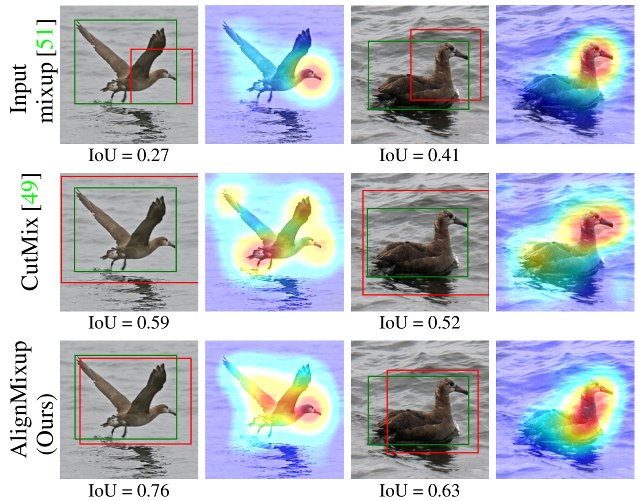


Figure 4: *Localization examples* using ResNet-50 on CUB200-2011. Red boxes: predicted; green: ground truth.

## E 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 6: *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 6. 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.