ALIGNMIX: IMPROVING REPRESENTATIONS BY INTERPOLATING ALIGNED FEATURES

Anonymous authors
Paper under double-blind review

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 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 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. AlignMix outperforms state-of-the-art mixup methods on five different benchmarks.

1 INTRODUCTION

Data augmentation (Krizhevsky et al., 2012; Paulin et al., 2014; Cubuk et al., 2019) is a powerful regularization method that increases the amount and diversity of data, be it labeled or unlabeled (Dosovitskiy et al., 2013). It improves the generalization performance and helps learning invariance (Simard et al., 1998) 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 (Zhang et al., 2017) and sensitivity to adversarial examples (Szegedy et al., 2014).

Mixup operates on two or more examples at a time, *interpolating* between them in the input space (Zhang et al., 2018a) or feature space (Verma et al., 2019), while also interpolating between target labels for image classification. This flattens class representations (Verma et al., 2019), 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 (Yun et al., 2019). Interestingly, recent mixup methods focus of combining two (Yun et al., 2019; Kim et al., 2020) or more (Kim et al., 2021) 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 (Uddin et al., 2021), which raises the question: *what is a good interpolation of images?*

Bengio et al. (2013) 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, Berthelot et al. (2018) show the ability of autoencoders to capture semantic correspondences obtained by decoding mixed latent codes. This is because the autoencoder may disentangle the underlying factors of variation. Efforts have followed on mixing latent representations of autoencoders to generate realistic images for data augmentation. However, these approaches are more expensive, requiring three networks (encoder, decoder, classifier) (Berthelot

Image 1 | Image 2 | Input mixup | CutMix | Manifold mixup | AlignMix (Ours)

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

et al., 2018) and more complex, often also requiring an adversarial discriminator (Beckham et al., 2019; Liu et al., 2018). More importantly, they perform poorly compared to standard input mixup on large datasets (Liu et al., 2018), due to the low quality of generated images.

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

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

We make the following contributions:

1. We introduce a novel mixup operation, called *AlignMix*, advocating interpolation of local structure in the feature space (subsection 3.2). Feature tensors are ideal for alignment, giving rise to semantic correspondences and being of low resolution. Alignment is efficient by using *Sinkhorn distance* (Cuturi, 2013).
2. We also show that a *vanilla autoencoder* can further improve representation learning under mixup training, without the classifier seeing decoded clean or mixed images (section 4).
3. We set a new state-of-the-art on *image classification*, *robustness to adversarial attacks*, *calibration*, *weakly-supervised localization* and *out-of-distribution detection* against more sophisticated mixup operations on several networks and datasets (section 4).

2 Related Work

Mixup Zhang et al. (2018a), concurrently with similar methods (Inoue, 2018; Tokozume et al., 2018), introduce *mixup*, augmenting data by linear interpolation between two examples. While (Zhang et al., 2018a) attempt to apply mixup on intermediate representations of the network, it is Verma et al. (2019) who make this work, introducing *manifold mixup*. Without alignment, the result is an overlay of either images (Zhang et al., 2018a) or features (Verma et al., 2019). Guo et al. (2019) eliminate "manifold intrusion"—mixed data conflicting with true data. Unlike manifold mixup, AlignMix interpolates feature tensors from deeper layers after aligning them.

Nonlinear mixing over random image regions is an alternative, *e.g.* from masking square regions (DeVries & Taylor, 2017) to cutting a rectangular region from one image and pasting it onto another (Yun et al., 2019), as well as several variants using arbitrary regions (Takahashi et al., 2018; Summers & Dinneen, 2019; Harris et al., 2020). Instead of choosing regions at random, *saliency* can be used to locate objects from different images and fit them in one (Uddin et al., 2021; Qin et al., 2020; Kim et al., 2020; 2021). Exploiting the knowledge of a teacher network to mix images based on saliency has been proposed in (Dabouei et al., 2021). Instead of combining more than one objects in an image, AlignMix attempts to deform one object into another.

Another alternative is Automix (Zhu et al., 2020), 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 (Verma et al., 2019). StyleMix and StyleCutMix (Hong et al., 2021) interpolate content and style between two images, using AdaIN (Huang & Belongie, 2017), a style transfer autoencoder network. By contrast, AlignMix 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* (Choy et al., 2016; Long et al., 2014), *optical flow* (Weinzaepfel et al., 2013), *semantic alignment* (Rocco et al., 2018; Han et al., 2017) and *image retrieval* (Siméoni et al., 2019). 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 (Doersch et al., 2020) and DeepEMD (Zhang et al., 2020). The *earth mover's distance* (EMD) (Rubner et al., 2000), or *Wasserstein metric*, is an instance of *optimal transport* (Villani, 2008), addressed by linear programming. To accelerate, Cuturi (2013) computes optimal matching by *Sinkhorn distance* with *entropic regularization*. This distance is widely applied between distributions in generative models (Genevay et al., 2018; Patrini et al., 2020).

EMD has been used for mixup in the input space, for instance *point mixup* for 3D point clouds (Chen et al., 2020) and OptTransMix for images (Zhu et al., 2020), 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 (Cuturi, 2013).

3 AlignMix

3.1 Preliminaries

Problem formulation Let (x, y) be an image $x \in \mathcal{X}$ with its one-hot encoded class label $y \in Y$, where $\mathcal{X}$ is the input image space, $Y = [0, 1]^k$ and k is the number of classes. The *encoder network* consists of two stages. The first is $F : \mathcal{X} \to \mathbb{R}^{c\times w\times h}$, which maps x to feature tensor $\mathbf{A} := F(x)$, where c is the number of channels and $w \times h$ is the spatial resolution. The second is $f : \mathbb{R}^{c\times w\times h} \to \mathbb{R}^d$, which maps tensor $\mathbf{A}$ to embedding $e := f(\mathbf{A})$. Finally, the *classifier* $g : \mathbb{R}^d \to \mathbb{R}^k$ maps the embedding e to the vector $p := g(e)$ of probabilities over classes.

Mixup We follow (Verma et al., 2019) in mixing the representations from different layers of the network, focusing on the deepest layers at or near the embedding. 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 $\mathrm{Beta}(\alpha, \alpha)$ (Zhang et al., 2018a) and then we interpolate labels y, y' linearly by the *standard* mixup operator

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

and inputs x, x' by the generic formula

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

where Mix_λ is a mixup operator to be defined. This generic formula allows interpolation of the input, feature or embedding by decomposing the network mapping $f \circ F$ as $f_2 \circ f_1$ according to

$$\text{input } (x): \quad f_1 := \mathrm{id}, f_2 := f \circ F \tag{3}$$

$$\text{feature } (\mathbf{A}): \quad f_1 := F, f_2 := f \tag{4}$$

$$\text{embedding } (e): \quad f_1 := f \circ F, f_2 := \mathrm{id}, \tag{5}$$

where id is the identity mapping. For (3) and (5), we define Mix_λ in (2) as standard mixup mix_λ (1), like *input* (Zhang et al., 2018a) and *manifold mixup* (Verma et al., 2019), respectively; while for (4), we define Mix_λ as discussed in subsection 3.2.

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

$$L_c(g(\mathrm{Mix}_\lambda^{f_1,f_2}(x, x')), \mathrm{mix}_\lambda(y, y')), \tag{6}$$

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

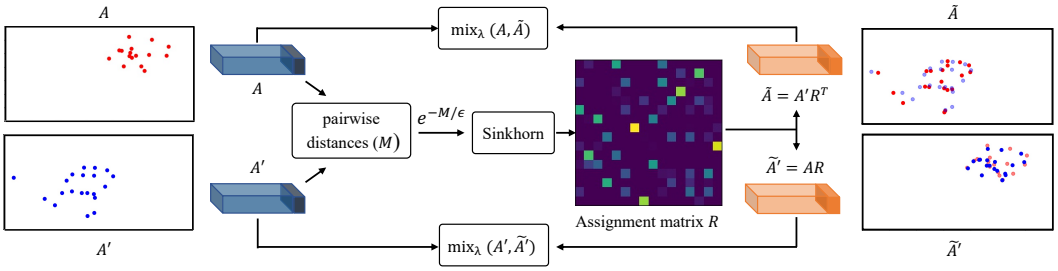

Figure 2: *Feature tensor alignment and interpolation*. Cost matrix M contains pairwise distances of feature vectors in tensors $\mathbf{A}, \mathbf{A}'$. Assignment matrix R is obtained by Sinkhorn-Knopp (Knight, 2008) on similarity matrix $e^{-M/\epsilon}$. $\mathbf{A}$ is aligned to $\mathbf{A}'$ 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.

3.2 Interpolation of Aligned Feature Tensors

Alignment Alignment refers to finding a geometric correspondence between image elements before interpolation. The feature tensor is ideal for this purpose, because its spatial resolution is low, reducing the optimization cost, and allows for semantic correspondence, because features close to the classifier—the second encoder f are small. Importantly, we are not attempting to combine two or more objects into one image (Kim et al., 2020), 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 (Villani, 2008) and *Sinkhorn distance* (SD) (Cuturi, 2013) in particular. Let $\mathbf{A} := F(x), \mathbf{A}' := F(x')$ be the $c \times w \times h$ feature tensors of images $x, x' \in \mathcal{X}$. We reshape them to $c \times r$ matrices A, A' by flattening the spatial dimensions, where $r := hw$. Then, every column $a_j, a'_j \in \mathbb{R}^c$ of A, A' for $j = 1, \ldots, r$ is a feature vector representing corresponding to a spatial position in the original image x, x'. Let M be the $r \times r$ *cost matrix* with its elements being the pairwise distances of these vectors:

$$m_{ij} := \left\| a_i - a'_j \right\|^2 \tag{7}$$

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

$$U_r := \{P \in \mathbb{R}_+^{r \times r} : P\mathbf{1} = P^\top \mathbf{1} = \mathbf{1}/r\} \tag{8}$$

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:

$$P^* = \arg\min_{P \in U_r} \langle P, M \rangle - \epsilon H(P), \tag{9}$$

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 (Knight, 2008), *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 (Alvarez-Melis & Jaakkola, 2018), 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:

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

$$\widetilde{A}' := AR. \tag{11}$$

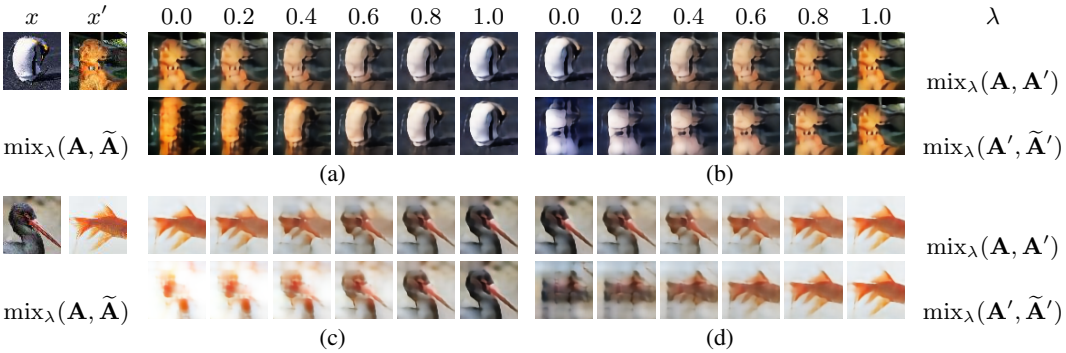

Figure 3: *Visualizing alignment*. For different $\lambda \in [0, 1]$, we interpolate feature tensors $\mathbf{A}, \mathbf{A}'$ without alignment (top) or aligned feature tensors (bottom) of two images x, x' and then we generate a new image by decoding the resulting embedding through the decoder D. (a), (c) We align $\mathbf{A}$ to $\mathbf{A}'$ and mix with (12). (b), (d) We align $\mathbf{A}'$ to $\mathbf{A}$ and mix with (13). Only meant for illustration: No decoded images are seen by the classifier at training.

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 $\mathbf{A}$:

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

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

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

When mixing feature tensors with alignment (4), we define Mix_λ in (2) as the mapping of $(\mathbf{A}, \mathbf{A}')$ to either (12) or (13), chosen at random.

3.3 Visualization and Discussion

Decoder We use a decoder to study images generated without or with feature alignment. Specifically, we use $f \circ F$ as an encoder and a *decoder* $D : \mathbb{R}^d \to \mathcal{X}$ maps the embedding 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 AlignMix training in section 4.

Discussion For different $\lambda \in [0, 1]$, we interpolate the feature tensors $\mathbf{A}, \mathbf{A}'$ of two images x, x' without or with alignment, using (12) or (13), and we generate a new image by decoding the resulting embedding through the decoder D.

In Figure 3, we visualize such generated images. Interestingly, by aligning $\mathbf{A}$ to $\mathbf{A}'$ and mixing using (12) with $\lambda = 0$, the generated image retains the pose of x and the texture of x'. In Figure 3(a) in particular, when x is 'penguin' and x' is 'dog', the generated image retains the pose of the penguin, while the texture of the dog aligns to the body of the penguin. Similarly, in Figure 3(c), the texture from the goldfish is aligned to that of the stork, while the pose of the stork is retained. Vice versa, as shown in Figure 3(b,d), by aligning $\mathbf{A}'$ to $\mathbf{A}$ and mixing using (13) 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, which captures 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 4.

DATASET NETWORK	CIFAR-10 R-18	 W16-8	CIFAR-100 R-18	 W16-8	TI R-18
Baseline	5.19	5.11	23.24	20.63	43.40
Input (Zhang et al., 2018a)	4.03	3.98	20.21	19.88	43.48
CutMix (Yun et al., 2019)	3.27	3.54	19.37	19.71	43.11
Manifold (Verma et al., 2019)	2.95	3.56	19.80	19.23	40.76
PuzzleMix (Kim et al., 2020)	2.93	**2.99**	20.01	19.25	36.52
Co-Mixup (Kim et al., 2021)	2.89	3.04	19.81	19.57	35.85
SaliencyMix (Uddin et al., 2021)	2.99	3.53	19.69	19.59	33.81
StyleMix (Hong et al., 2021)	3.76	3.89	20.04	20.45	36.13
StyleCutMix (Hong et al., 2021)	3.06	3.12	19.34	19.28	33.49
AlignMix (ours)	2.95	3.09	18.08	18.67	**31.81**
AlignMix/AE (ours)	**2.83**	3.15	**17.82**	**18.09**	32.73
Gain	**+0.06**	**-0.10**	**+1.52**	**+1.14**	**+1.68**

(a) *Image classification* top-1 error (%) on CIFAR-10/100 and TI (TinyImagenet). R: PreActResnet, W: WRN.

METHOD	PARAM.	MSEC/ BATCH	TOP-1 ERROR
Baseline	25M	418	23.68
Input† (Zhang et al., 2018a)	25M	436	22.58
CutMix† (Yun et al., 2019)	25M	427	21.40
Manifold† (Verma et al., 2019)	25M	441	22.50
PuzzleMix† (Kim et al., 2020)	25M	846	21.24
Co-Mixup (Kim et al., 2021)	25M	1022	–
SaliencyMix* (Uddin et al., 2021)	25M	462	21.26
StyleMix* (Hong et al., 2021)	25M	828	24.06
StyleCutMix* (Hong et al., 2021)	25M	912	22.71
AlignMix (ours)	28M	450	20.32
AlignMix/AE (ours)	35M	688	**18.83**
Gain			**+2.41**

(b) *Image classification* top-1 error (%) and *computational analysis* on ImageNet using Resnet-50. $*$: reported by authors; †: reported by PuzzleMix.

Table 1: *Image classification* top-1 error (%, lower is better) and *computational analysis*. Blue: second best. Gain: reduction of error over best performing baseline.

4 EXPERIMENTS

4.1 IMPLEMENTATION DETAILS

Architecture We use a residual network as the stage 1 encoder F. The output $\mathbf{A}$ is a $c \times 4 \times 4$ tensor. This is followed by a fully-connected layer as stage 2 encoder f with output embedding $e \in \mathbb{R}^d$ and another fully-connected layer as classifier g.

Autoencoder In Figure 3, we have used a decoder to visualize the effect of feature tensor alignment. In our experiments, we also use a decoder optionally during training of AlignMix, to investigate its effect on representation learning under mixup. This results in a vanilla autoencoder architecture, which we denote as AlignMix/AE, while other options are investigated in subsection C.1 of the Appendix. We use a residual generator (Gulrajani et al., 2018) as the decoder D. The encoder and decoder have the same architecture.

Training By default, we train AlignMix using only the classfication loss L_c (6) on mixed examples. For a given mini-batch during training, we mix either x, $\mathbf{A}$ (using either (12) or (13) for alignment) or e. We choose between the four cases uniformly at random. For AlignMix/AE, we either use the reconstruction loss L_r on clean examples, training the encoder and decoder, or the classfication loss L_c (6) on mixed examples, training the encoder and classifier. This gives rise to a fifth case and we choose again uniformly at random. The complete algorithm is summarized in Appendix A.

Hyperparameters The hyperparameters used for different datasets are reported in Appendix B.

4.2 IMAGE CLASSIFICATION AND ROBUSTNESS

We use PreActResnet18 (He et al., 2016) (R-18) and WRN16-8 (Zagoruyko & Komodakis, 2016) as the backbone architecture on CIFAR-10 and CIFAR-100 datasets (Krizhevsky et al., 2009). Using our experimental settings (in supplementary material), we reproduce the state-of-the-art (SOTA) mixup methods: Baseline network (without mixup), Input mixup (Zhang et al., 2018a), Manifold mixup (Verma et al., 2019), CutMix (Yun et al., 2019), PuzzleMix (Kim et al., 2020), Co-Mixup (Kim et al., 2021), SaliencyMix (Uddin et al., 2021), StyleMix (Hong et al., 2021) and StyleCutMix (Hong et al., 2021) using official code provided by the authors. We do not compare AlignMix with AutoMix (Zhu et al., 2020) and Re-Mix (Cao et al., 2021), since its experimental settings are different from ours and there is no available code.

In addition, we use R-18 as the backbone network on TinyImagenet (Yao & Miller, 2015) (TI) and reproduce SaliencyMix (Uddin et al., 2021), StyleMix (Hong et al., 2021) and StyleCutMix (Hong

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 (Zhang et al., 2018a)	78.42	79.21	81.30	67.33	88.68	99.77	99.43	99.96	99.37
CutMix (Yun et al., 2019)	77.72	78.33	86.96	60.16	88.68	99.82	98.10	98.67	97.98
Manifold (Verma et al., 2019)	77.63	76.11	80.29	56.45	89.25	97.22	98.49	99.66	98.43
PuzzleMix (Kim et al., 2020)	41.11	50.73	78.70	57.77	83.91	97.73	97.00	96.42	95.28
Co-Mixup (Kim et al., 2021)	40.19	48.93	77.61	56.59	–	97.59	96.19	95.35	94.23
SaliencyMix (Uddin et al., 2021)	57.43	68.10	77.79	58.10	81.16	97.51	97.04	95.68	93.76
StyleMix (Hong et al., 2021)	79.54	71.05	80.54	67.94	84.93	98.23	97.46	98.39	98.24
StyleCutMix (Hong et al., 2021)	38.79	46.12	77.49	56.83	80.59	97.87	96.70	93.88	93.78
AlignMix (ours)	38.33	53.41	77.29	**55.05**	**77.34**	**96.36**	96.73	**93.18**	**92.16**
AlignMix/AE (ours)	**32.13**	**44.86**	**76.40**	55.44	78.98	97.16	**95.32**	93.69	**92.23**
Gain	**+6.66**	**+1.26**	**+1.09**	**+1.40**	**+3.25**	**+0.86**	**+0.87**	**+0.70**	**+1.40**

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.

et al., 2021) following the experimental settings of (Kim et al., 2021), and Resnet-50 (R-50) on ImageNet (Russakovsky et al., 2015), following the training protocol of (Kim et al., 2020). Using top-1 error (%) as evaluation metric, we show the effectiveness of AlignMix on image classification and robustness to FGSM (Goodfellow et al., 2015) and PGD (Madry et al., 2018) attacks.

Image classification As shown in Table 1(a), AlignMix and AlignMix/AE is on par or outperforms the SOTA methods by achieving the lowest top-1 error, especially on large datasets. On CIFAR-10, AlignMix and AlignMix/AE is on par with Co-Mixup and Puzzlemix with R-18 and WRN16-8. On CIFAR-100, AlignMix outperforms StyleCutMix and Manifold mixup by 1.52% and 1.14% with R-18 and WRN16-8, respectively. On TI, AlignMix outperforms Co-Mixup by 3.12% using R-18. According to Table 1(b), AlignMix/AE outperforms PuzzleMix by 2.41% on ImageNet. Importantly, while the overall improvement by SOTA methods on ImageNet over Baseline is around 2%, AlignMix improves SOTA by another 2.5%.

Computational complexity Table 1(b) shows the computational analysis of AlignMix training as compared with baseline and SOTA mixup methods on ImageNet, in terms of number of parameters and msec/batch on a NVIDIA RTX 2080 TI GPU. AlignMix has nearly the same computational overhead as Manifold mixup while achieving 3.16% increase of accuracy. While SOTA methods like Co-Mixup, PuzzleMix, StyleMix and StyleCutMix are computationally more expensive than AlignMix by 1.8×, 2.3×,1.8× and 2× respectively, they are outperformed by AlignMix by 2% on average. AlignMix/AE brings a further 1.49% gain in accuracy over AlignMix. It is important to note that 40% increase in number of parameters of AlignMix/AE is due to the residual decoder, which is only used in one out of five cases on clean images without mixup. Computational complexity during inference is the same for all methods.

Robustness to FGSM and PGD attacks Following the evaluation protocol of (Kim et al., 2020), we use $8/255$ l_∞ ϵ-ball for FGSM and $4/255$ l_∞ ϵ-ball with step size $2/255$ for PGD. We reproduce the results of competitors for FGSM and PGD on CIFAR-10 and CIFAR-100; results of baseline, Input, Manifold, Cutmix and Puzzlemix on TI for FGSM are as reported in (Kim et al., 2020) and reproduced for SaliencyMix, StyleMix and StyleCutMix.

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.

4.3 OVERCONFIDENCE

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

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 (Zhang et al., 2018a)	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 (Yun et al., 2019)	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 (Verma et al., 2019)	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 (Kim et al., 2020)	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 (Kim et al., 2021)	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 (Uddin et al., 2021)	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 (Hong et al., 2021)	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 (Hong et al., 2021)	70.8	78.6	83.7	74.9	70.6	82.4	83.7	76.5	75.3	82.6	82.9	78.4
AlignMix (ours)	76.1	80.7	85.9	75.8	73.4	**85.1**	84.3	80.2	79.4	85.0	88.4	85.0
AlignMix/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**	**+2.7**	**+2.2**	**+1.9**	**+4.4**	**+3.2**	**+2.6**	**+3.8**

Table 3: *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 Appendix C.

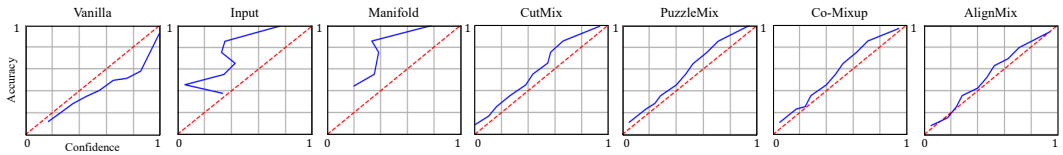

Figure 4: *Calibration* plots on CIFAR-100 using PreActResnet18: near diagonal is better. Baseline is clearly overconfident while Input and Manifold mixup are clearly under-confident. AlignMix has the best calibrated predictions.

Out-of-distribution detection According to (Hendrycks & Gimpel, 2017), *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 subsection 4.2. At inference, ID examples are test images from CIFAR-100, while OOD examples are test images from LSUN (crop) (Yu et al., 2015), iSUN (Xiao et al., 2010) and Tiny-ImageNet (crop); where crop denotes that the OOD examples are center-cropped to 32×32 to match the resolution of ID images (Yun et al., 2019). Following (Hendrycks & Gimpel, 2017), 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, 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 (Yun et al., 2019). More results are given in Table 6 of Appendix C.

Calibration According to (DeGroot & Fienberg, 1983), 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.

As shown in Figure 4, while SOTA methods are under-confident compared to Baseline, AlignMix results in the best calibration among all competitors. We quantitatively evaluate the calibration of AlignMix against SOTA methods in terms of *expected calibration error* (ECE) (Guo et al., 2017) and *overconfidence error* (OE) (Thulasidasan et al., 2019) using R-18 on CIFAR-100. Results indicating that AlignMix has the lowest ECE and OE are given in Table 7 of Appendix C.

METRIC	TOP-1 LOC.		MAXBOXACC-V2	
NETWORK	VGG-GAP	RESNET-50	VGG-GAP	RESNET-50
ACoL (Zhang et al., 2018b)	45.9	–	57.4	–
ADL (Choe & Shim, 2019)	52.4	–	61.3	58.4
Baseline CAM (Zhou et al., 2016)	37.1	49.4	59.0	59.7
Input (Zhang et al., 2018a)	41.7	49.3	57.1	60.6
Cutout (DeVries & Taylor, 2017)	44.8	52.8	–	–
CutMix (Yun et al., 2019)	52.5	54.8	62.6	64.8
AlignMix (ours)	**53.7**	**57.8**	**64.5**	**65.9**
Gain	**+1.2**	**+3.0**	**+1.9**	**+1.1**

Table 4: *Weakly-supervised object localization* on CUB200-2011. Top-1 loc.: Top-1 localization accuracy (%), MaxBoxAcc-v2: Maximal box accuracy (Choe et al., 2020). Higher is better. Blue: second best. Gain: increase of accuracy.

4.4 WEAKLY-SUPERVISED OBJECT LOCALIZATION (WSOL)

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

We train AlignMix using the same procedure as for image classification. At inference, following (Yun et al., 2019), we compute a saliency map using CAM (Zhou et al., 2016), binarize it using a threshold of 0.15 and take the bounding box of the mask. We use VGG-GAP (Simonyan & Zisserman, 2015) and Resnet-50 (He et al., 2016) as pretrained on Imagenet (Russakovsky et al., 2015) and we fine-tune them on CUB200-2011 (Wah et al., 2011). We follow the evaluation protocol by Choe et al. (2020) and use top-1 localization accuracy with IoU threshold of 0.5 and Maximal Box Accuracy (MaxBoxAcc-v2) to compare AlignMix with baseline CAM (without mixup), Input mixup (Zhang et al., 2018a), CutOut (DeVries & Taylor, 2017) and CutMix (Yun et al., 2019).

According to Table 4, AlignMix outperforms Input mixup, CutOut 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 in terms of top-1 localization accuracy. Furthermore, AlignMix outperforms CutMix by 1.9% and 1.1% using VGG-GAP and Resnet-50 respectively in terms of MaxBoxAcc-v2. It also outperforms dedicated WSOL methods ACoL (Zhang et al., 2018b) and ADL (Choe & Shim, 2019), which focus on learning spatially dispersed representations. Qualitative localization results are given in Appendix C.

4.5 ABLATION

We study the effect of mixing different layers (x, $\mathbf{A}$ or e), aligning $\mathbf{A}$ or not before mixing, the resolution used when aligning, as well as the autoencoder architecture in subsection C.1. The analysis also includes studying the effect of using a *vanilla autoencoder* (AlignMix/AE), a *variational autoencoder* (Kingma & Welling, 2013) (AlignMix/VAE) and *no decoder* (AlignMix).

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" 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

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

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.

Yoshua Bengio, Grégoire Mesnil, Yann Dauphin, and Salah Rifai. Better mixing via deep representations. In *International Conference on Machine Learning (ICML)*, 2013.

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.

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

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

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

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

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

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)*, pp. 113–123, 2019.

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

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

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.

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

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

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

Aude Genevay, Gabriel Peyré, and Marco Cuturi. Learning generative models with sinkhorn divergences. In *International Conference on Artificial Intelligence and Statistics*, 2018.

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

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

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning (ICML)*, 2017.

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

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)*, pp. 1831–1840, 2017.

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.

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)*, pp. 770–778, 2016.

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.

Minui Hong, Jinwoo Choi, and Gunhee Kim. Stylemix: Separating content and style for enhanced data augmentation. In *CVPR*, 2021.

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

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

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)*, 2020.

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.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Philip A Knight. The Sinkhorn-Knopp algorithm: convergence and applications. *SIAM Journal on Matrix Analysis and Applications*, 2008.

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

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.

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.

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

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.

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.

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)*, pp. 3646–3653, 2014.

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.

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

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.

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.

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*, pp. 239–274. Springer, 1998.

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)*, pp. 11651–11660, 2019.

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

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

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.

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.

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

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

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.

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)*, pp. 6438–6447, 2019.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

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.

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)*, pp. 1385–1392, 2013.

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)*, 2010.

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

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.

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)*, pp. 6023–6032, 2019.

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

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)*, pp. 12203–12213, 2020.

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.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations (ICLR)*, 2018a.

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)*, pp. 1325–1334, 2018b.

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)*, pp. 2921–2929, 2016.

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)*, 2020.

A ALGORITHM

AlignMix and AlignMix/AE are summarized in algorithm 1. By default (AlignMix), for each mini-batch, we uniformly draw at random one among four choices (line 2) over mixup on input (x), embeddings (e), or feature tensors ($\mathbf{A}$, using either (12) or (13) for mixing). For AlignMix/AE, there is a fifth choice where we only use reconstruction loss on clean examples (line 7).

For mixup, use only classification loss (6) (line 27). Following (Verma et al., 2019), 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) and embeddings e, e' by (2),(5) (line 14). Feature tensors $\mathbf{A}$ and $\mathbf{A}'$ are first aligned and then mixed by (2),(12) ($\mathbf{A}$ aligns to $\mathbf{A}'$) or (2),(13) ($\mathbf{A}'$ aligns to $\mathbf{A}$) (lines 17,26).

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

Input: encoders F, f; 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* $\sim \text{unif}\{\textit{clean}, \textit{input}, \textit{embed}, \textit{feat}, \textit{feat}'\}$ ▷ mixup?
3 **for** $i \in \{1, \ldots, b\}$ **do**
4 $(x, y) \leftarrow (x_i, y_i)$ ▷ current example
5 **if** *mode* $=$ *clean* **then** ▷ no mixup
6 $\hat{x} \leftarrow D(f(F(x)))$ ▷ encode/decode
7 $\ell_i \leftarrow L_r(x, \hat{x})$ ▷ reconstruction loss
8 **else** ▷ mixup
9 $\lambda \sim \text{Beta}(\alpha, \alpha)$ ▷ interpolation factor
10 $(x', y') \leftarrow (x_{\pi(i)}, y_{\pi(i)})$ ▷ paired example
11 **if** *mode* $=$ *input* **then** ▷ as in (Zhang et al., 2018a)
12 $e \leftarrow f(F(\text{mix}_\lambda(x, x')))$ ▷ (2),(3)
13 **else if** *mode* $=$ *embed* **then** ▷ as in (Verma et al., 2019)
14 $e \leftarrow \text{mix}_\lambda(f(F((x)), f(F(x')))$ ▷ (2),(5)
15 **else** ▷ *mode* $\in \{\textit{feat}, \textit{feat}'\}$
16 **if** *mode* $=$ *feat*$'$ **then** ▷ choose (13) over (12)
17 SWAP (x, x'), SWAP (y, y')
18 $\mathbf{A} \leftarrow F(x)$, $\mathbf{A}' \leftarrow F(x')$ ▷ 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 (7)
22 $P^* \leftarrow \text{SINKHORN}(\exp(-M/\epsilon))$ ▷ tran. plan (9)
23 $R \leftarrow \text{DETACH}(rP^*)$ ▷ assignments
24 $\widetilde{A} \leftarrow A' R^\top$ ▷ alignment (10)
25 $\widetilde{\mathbf{A}} \leftarrow \text{RESHAPE}_{c \times w \times h}(\widetilde{A})$ ▷ to tensor
26 $e \leftarrow f(\text{mix}_\lambda(\mathbf{A}, \widetilde{\mathbf{A}}))$ ▷ (2),(12)
27 $\ell_i \leftarrow L_c(g(e), \text{mix}_\lambda(y, y'))$ ▷ classification loss (6)

In computing loss derivatives, we backpropagate through embeddings e, e' or feature tensors $\mathbf{A}, \mathbf{A}'$ but not through the transport plan P^* (line 23). Hence, although the Sinkhorn-Knopp algorithm (Knight, 2008) is differentiable, its iterations take place only in the forward pass. Importantly, AlignMix is easy to implement and does not require sophisticated optimization like (Kim et al., 2020; 2021).

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 $\text{Beta}(\alpha, \alpha)$

NETWORK	RESNET-50
Baseline	24.03
Input (Zhang et al., 2018a)	22.97
Manifold (Verma et al., 2019)	23.30
CutMix (Yun et al., 2019)	22.92
PuzzleMiz (Kim et al., 2020)	22.49
Co-Mixup (Kim et al., 2021)	22.39
AlignMix (ours)	**20.76**
Gain	**+1.63**

Table 5: *Image classification* on ImageNet for 100 epochs using ResNet-50. Top-1 error (%): lower is better. Blue: second best. Gain: reduction of error.

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 (Knight, 2008) for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

TinyImagenet We follow the training protocol of Kim *et al*. (Kim et al., 2020), training R-18 as stage-1 encoder F using SGD for 1200 epochs. We set the initial learning rate to 0.1 and decay it by 0.1 at 600 and 900 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 128 on 2 GPUs. The interpolation factor is drawn from $\text{Beta}(\alpha, \alpha)$ where $\alpha = 2.0$. For alignment, we apply the Sinkhorn-Knopp algorithm (Knight, 2008) for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

ImageNet We follow the training protocol of Kim *et al*. (Kim et al., 2020), where training R-50 as F using SGD for 300 epochs. The initial learning rate of the classifier and the remaining layers is set to 0.1 and 0.01, respectively. We decay the learning rate by 0.1 at 100 and 200 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 100 on 4 GPUs. The interpolation factor is drawn from $\text{Beta}(\alpha, \alpha)$ where $\alpha = 2.0$. For alignment, we apply the Sinkhorn-Knopp algorithm (Knight, 2008) 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*. (Kim et al., 2021).

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 (Yun et al., 2019), 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 (Zhou et al., 2016). The classifier is modified to have 200 classes instead of 1000.

For fair comparisons with (Yun et al., 2019), 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 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 (Kim et al., 2021), Table 5 reports classification performance when training for 100 epochs on ImageNet. Using the top-1 error (%) reported for competitors by (Kim et al., 2021), AlignMix outperforms all methods, including Co-Mixup (Kim et al., 2021). Importantly, while the overall improvement by SOTA methods over Baseline is around 1.64%, AlignMix improves SOTA by another 1.63%.

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 (Zhang et al., 2018a)	61.5	66.5	66.4	65.8	59.6	63.8	63.0	63.4
Cutmix (Yun et al., 2019)	71.3	77.4	79.1	75.5	69.1	79.4	79.8	73.3
Manifold (Verma et al., 2019)	67.8	78.9	76.3	71.3	62.5	77.8	76.8	72.2
PuzzleMix (Kim et al., 2020)	74.9	79.9	84.0	77.5	73.9	77.3	80.6	71.9
Co-Mixup (Kim et al., 2021)	73.8	82.6	86.8	76.9	68.1	78.9	82.5	74.2
SaliencyMix (Uddin et al., 2021)	75.8	79.7	82.2	84.4	75.3	81.2	83.8	79.5
StyleMix (Hong et al., 2021)	73.0	74.6	72.4	73.4	72.9	79.5	78.2	74.6
StyleCutMix (Hong et al., 2021)	74.3	83.1	86.9	78.9	73.8	80.9	83.1	76.3
AlignMix (ours)	76.1	84.3	87.1	**85.8**	74.7	82.6	86.1	80.9
AlignMix/AE (ours)	**77.0**	**85.8**	**87.9**	83.7	**76.2**	**84.8**	**87.2**	**82.3**
Gain	**+2.1**	**+2.7**	**+1.0**	**+1.4**	**+0.9**	**+3.6**	**+3.4**	**+2.8**
NOISE	UNIFORM				GAUSSIAN			
Baseline	58.3	75.3	75.0	69.0	60.8	64.3	62.9	63.9
Input (Zhang et al., 2018a)	50.0	67.9	71.8	71.7	60.2	65.0	63.1	64.1
Cutmix (Yun et al., 2019)	74.8	80.0	84.9	72.4	75.7	79.0	84.0	70.9
Manifold (Verma et al., 2019)	69.8	75.9	83.2	71.9	70.8	78.8	81.3	71.6
PuzzleMix (Kim et al., 2020)	78.6	85.2	86.0	74.4	78.5	85.1	85.9	74.3
Co-Mixup (Kim et al., 2021)	80.4	87.6	87.4	75.2	81.6	78.6	89.5	74.2
SaliencyMix (Uddin et al., 2021)	83.1	87.4	89.1	76.6	82.4	85.4	81.1	**81.3**
StyleMix (Hong et al., 2021)	75.3	71.8	77.8	65.5	78.0	75.2	84.3	71.0
StyleCutMix (Hong et al., 2021)	84.5	83.2	88.6	78.3	84.8	81.9	83.3	73.9
AlignMix (ours)	86.9	89.1	93.6	77.7	**86.7**	**87.9**	91.8	77.4
AlignMix/AE (ours)	**88.0**	**90.6**	**94.0**	**80.8**	86.0	87.2	**91.9**	75.6
Gain	**+3.5**	**+3.0**	**+4.9**	**+2.5**	**+1.9**	**+2.8**	**+2.4**	**-3.9**

Table 6: *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.

METRIC	ECE	OE
Baseline	10.25	1.11
Input (Zhang et al., 2018a)	18.50	1.42
CutMix (Yun et al., 2019)	7.60	1.05
Manifold (Verma et al., 2019)	18.41	0.79
PuzzleMix (Kim et al., 2020)	8.22	0.61
Co-Mixup (Kim et al., 2021)	5.83	0.55
SaliencyMix (Uddin et al., 2021)	5.89	0.59
StyleMix (Hong et al., 2021)	11.43	1.31
StyleCutMix (Hong et al., 2021)	9.30	0.87
AlignMix (ours)	5.78	**0.41**
AlignMix/AE (ours)	**5.06**	0.48
Gain	**+0.77**	**+0.14**

Table 7: *Calibration* using PreActResnet18 on CIFAR-100. ECE : expected calibration error; OE: overconfidence error. Lower is better. Blue: second best. Gain: reduction of error.

Out-of-distribution detection We compare AlignMix with SOTA methods, training R-18 on CIFAR-100 as discussed in subsection 4.2. At inference, ID examples are test images from CIFAR-100, while OOD examples are test images from LSUN (Yu et al., 2015) and Tiny-ImageNet, resizing OOD examples to 32×32 to match the resolution of ID images (Yun et al., 2019). 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 (Hendrycks & Gimpel, 2017), 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 6, AlignMix outperforms SOTA methods under all metrics by a large margin, indicating that it is better in reducing over-confident predictions.

Calibration We compare AlignMix with SOTA methods, training R-18 on CIFAR-100 as discussed in subsection 4.2. All SOTA mixup methods are reproduced using the same experimental settings. Following (Thulasidasan et al., 2019), we measure measure the *expected calibration error* (ECE) and *overconfidence error* (OE). As shown in Table 7, AlignMix 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 5 indicate that AlignMix encodes semantically discriminative representations, resulting in better localization performance.

C.1 ABLATION STUDY

All ablations are performed on CIFAR-100 using R-18 as stage 1 encoder F with feature tensor $\mathbf{A}$ being $512 \times 4 \times 4$ and embedding $e \in \mathbb{R}^{512}$. We study the effect of mixing different layers (x, $\mathbf{A}$ or e), 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* (Kingma & Welling, 2013) (AlignMix/VAE) and *no decoder* (AlignMix). We report top-1 accuracy (%). All results are shown in Table 8.

Layers In general, we may mix any layer in $\{x, \mathbf{A}, e\}$ in a given iteration. We ablate the effect of allowing only a particular subset of layers. In general, $e \in \mathbb{R}^{512}$ is a vector. Here, we also consider the case where e is a $128 \times 2 \times 2$ tensor, denoted as $\mathbf{E}$ and obtained from $\mathbf{A}$ by a convolutional layer of kernel size 2×2 and stride 2. In AlignMix/AE architecture, among different choices of unaligned layer sets, mixing from $\{x, e\}$ results in the highest classification accuracy. Furthermore, AlignMix/AE outperforms Baseline and the best performing competitor StyleCutMix for all choices of layer sets, even when features are unaligned.

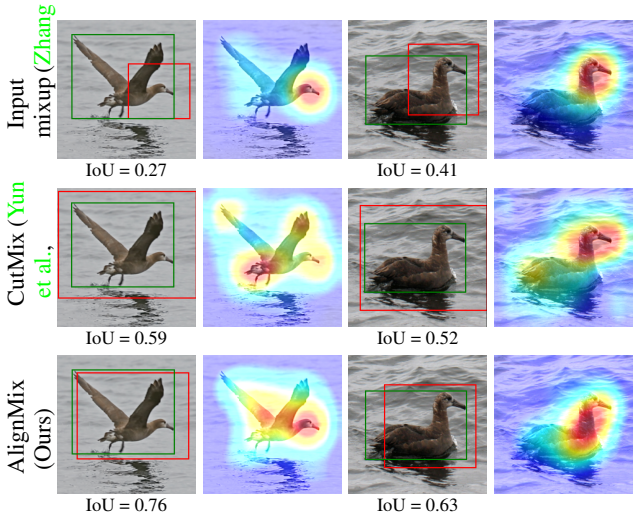

Figure 5: *Localization examples* using ResNet-50 on CUB200-2011. Red boxes: predicted; green: ground truth.

METHOD/ARCH	LAYERS	UNALIGNED	ALIGNED
Baseline		76.76	–
Manifold (Verma et al., 2019)		80.20	-
StyleCutMix (Hong et al., 2021)		80.66	-
AlignMix	$\{x, e\}$	80.81	–
	$\{x, \mathbf{A}\}$	80.34	81.61
	$\{x, \mathbf{A}, \mathbf{E}\}$	80.46	81.36
	$\{x, \mathbf{A}, e\}$	80.33	**81.92**
AlignMix/AE	$\{x, e\}$	81.92	-
	$\{x, \mathbf{A}\}$	81.78	81.85
	$\{x, \mathbf{E}\}$	80.80	81.54
	$\{x, \mathbf{A}, e\}$	81.61	**82.18**
AlignMix/AE	$\{x, \mathbf{A}_{2\times2}, e\}$	81.47	81.20
	$\{x, \mathbf{A}_{4\times4}, e\}$	81.61	82.18
	$\{x, \mathbf{A}_{8\times8}, e\}$	80.49	**82.20**
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**

Table 8: *Ablation study* using R-18 on CIFAR-100. Top-1 classification accuracy (%): higher is better. Arch: autoencoder architecture. AE: vanilla; VAE: variational (Kingma & Welling, 2013).. Layer $x, \mathbf{A}, e$: (3), (4), (5).

Tensor alignment We ablate the effect of aligning feature tensor $\mathbf{A}$ or not before mixing it, by using standard mixup (2) or (12), (13), respectively. In AlignMix/AE architecture, we observe that aligning $\mathbf{A}$ before mixing improves classification accuracy significantly. It is important to note that when e is a vector, we do not align it. However, when it is a tensor $\mathbf{E}$, aligning it improves significantly. Overall, AlignMix/AE works the best when $x, \mathbf{A}, e$ are mixed, with $\mathbf{A}$ being aligned. This setting outperforms StyleCutMix by 1.52%. Mixing $\mathbf{A}$ only helps when it is aligned; otherwise, it is preferable to just mix e.

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. The former has two vectors $\mu, \sigma \in \mathbb{R}^{512}$ instead of e, representing mean and standard deviation, respectively. We also investigate $128 \times 2 \times 2$ tensors, denoted as $\mathbf{M}, \mathbf{\Sigma}$ where the two variables are mixed simultaneously. As for AlignMix/AE, we investigate different combinations of layers with or without alignment. Both AlignMix and AlignMix/VAE are inferior to AlignMix/AE. However, their best setting still outperforms Baseline and StyleCutmix. All three architecture work best when $x, \mathbf{A}, e$ are mixed. Alignment improves consistently on all three architectures.