Supplementary material of
AlignMixup: Improving Representations By Interpolating Aligned Features

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

A. Algorithm

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

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

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

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

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

B. Hyperparameter settings

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

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

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

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

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

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

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

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

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

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

C. Additional experiments

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

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

Out-of-distribution detection We compare AlignMixup with SOTA methods, training R-18 on CIFAR-100 as discussed in subsection 4.2. At inference, ID examples are test

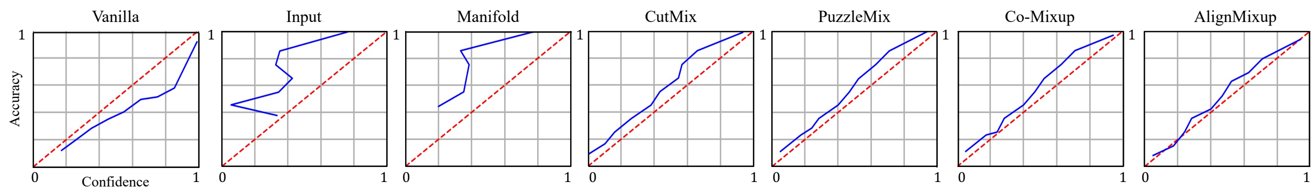

Figure 4. *Calibration plots* on CIFAR-100 using PreActResnet18: near diagonal is better. We plot accuracy *vs*. confidence, that is, probability for the predicted class.

images from CIFAR-100, while OOD examples are test images from LSUN [64] and Tiny-ImageNet, resizing OOD examples to 32×32 to match the resolution of ID images [65]. We also use test images from CIFAR-100 with Uniform and Gaussian noise as OOD samples. Uniform is drawn from $\mathcal{U}(0, 1)$ and Gaussian from $\mathcal{N}(\mu, \sigma)$ with $\mu = \sigma = 0.5$. All SOTA mixup methods are reproduced using the same experimental settings. Following [27], we measure *detection accuracy* (Det Acc) using a threshold of 0.5, *area under ROC curve* (AuROC) and *area under precision-recall curve* (AuPR).

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

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

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

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

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

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

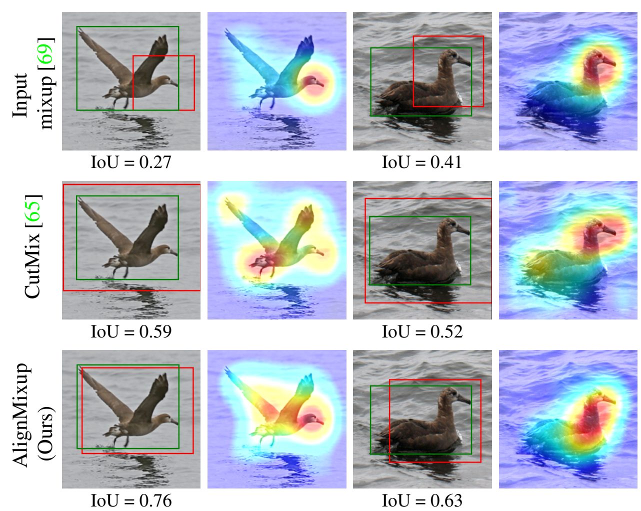

Figure 5. *Localization examples* using ResNet-50 on CUB200-2011. Red boxes: predicted; green: ground truth.

D. Additional ablations

Iterations in Sinkhorn-Knopp The default number of iterations for the Sinkhorn-Knopp algorithm in solving (8) is $i = 100$. Here, we investigate more choices, as shown in Table 10. The case of $i = 0$ is similar to cross-

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

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

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.