Supplementary material of
AlignMixup: Improving Representations By Interpolating Aligned Features

Shashanka Venkataramanan\textsuperscript{1} Ewa Kijak\textsuperscript{1} Laurent Amsaleg\textsuperscript{1} Yannis Avrithis\textsuperscript{2}
\textsuperscript{1}Inria, Univ Rennes, CNRS, IRISA \textsuperscript{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).

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 \cite{cuturi2013sinkhorn} is differentiable, its iterations take place only in the forward pass. Importantly, AlignMixup is easy to implement and does not require sophisticated optimization like \cite{he2019momentum, liu2019towards}.

B. Hyperparameter settings

CIFAR-10/CIFAR-100 We train AlignMixup using SGD for 2000 epochs with an initial learning rate of 0.1, decayed by a factor 0.1 every 500 epochs. We set the momentum as 0.9 with a weight decay of 0.0001 and use a batch size of 128. The interpolation factor is drawn from $\text{Beta}(\alpha, \alpha)$ where $\alpha = 2.0$. Using these settings, we reproduce the results of SOTA mixup methods for image classification, robustness to FGSM and PGD attacks, calibration and out-of-distribution detection. For alignment, we apply the Sinkhorn-Knopp algorithm \cite{cuturi2013sinkhorn} for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

TinyImagenet We follow the training protocol of Kim \textit{et al.} \cite{kim2020tinyimagenet}, 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 \cite{cuturi2013sinkhorn} for 100 iterations with entropic regularization coefficient $\epsilon = 0.1$.

\begin{algorithm}[!h]
\caption{AlignMixup/AE (parts involved in the AE variant indicated in blue)}\label{alg:alignmixup}
\begin{algorithmic}[1]
\Require
\Statex encoders $F$; embedding $e$, decoder $D$; classifier $g$
\Ensure mini-batch $B := \{(x_i, y_i)\}_{i=1}^b$
\Ensure loss values $L := \{\ell_i\}_{i=1}^b$
\State $\pi \sim \text{unif}(\mathcal{S}_b)$ \Comment{random permutation of $\{1, \ldots, b\}$}
\State $\text{mode} \sim \text{unif}\{\text{clean, input, feat, feat'}\}$ \Comment{mixup?}
\For{$i \in \{1, \ldots, b\}$}
\State $(x, y) \leftarrow (x_i, y_i)$ \Comment{current example}
\If{$\text{mode} = \text{clean}$}
\State $x \leftarrow D(e(F(x)))$ \Comment{encode/decode}
\State $\ell_i \leftarrow L_c(x, \hat{x})$ \Comment{reconstruction loss}
\Else
\State $\lambda \sim \text{Beta}(\alpha, \alpha)$ \Comment{interpolation factor}
\State $(x', y') \leftarrow (x_{\pi(i)}, y_{\pi(i)})$ \Comment{paired example}
\If{$\text{mode} = \text{input}$}
\State $\text{out} \leftarrow F(\text{mix}_b(x, x'))$ \Comment{(2),(3)} \Comment{mixup}
\Else
\State $\text{SWAP}(x, x'), \text{SWAP}(y, y')$ \Comment{(2),(3)} \Comment{mixup}
\EndIf
\EndIf
\EndFor
\State $\mathbf{A} \leftarrow F(x), \mathbf{A'} \leftarrow F(x')$ \Comment{feature tensors}
\State $\mathbf{A} \leftarrow \text{RESHAPE}_{e \times r}(\mathbf{A})$ \Comment{to matrix}
\State $\mathbf{A'} \leftarrow \text{RESHAPE}_{e \times r}(\mathbf{A'})$ \Comment{to matrix}
\State $\mathbf{M} \leftarrow \text{DIST}(\mathbf{A}, \mathbf{A'})$ \Comment{pairwise distances (6)}
\State $P^* \leftarrow \text{SINKHORN}((\mathbf{M}/e)^\top)$ \Comment{tran. plan (8)}
\State $\mathbf{R} \leftarrow \text{DETACH}(rP^*)$ \Comment{assignments}
\State $\mathbf{A} \leftarrow \mathbf{A'R}^\top$ \Comment{alignment (9)}
\State $\mathbf{\bar{A}} \leftarrow \text{RESHAPE}_{c \times w \times h}(\mathbf{A})$ \Comment{to tensor}
\State $\text{out} \leftarrow f(\text{mix}_b(\mathbf{A}, \mathbf{\bar{A}}))$ \Comment{(2),(11)} \Comment{classification loss (5)}
\State $\ell_i \leftarrow L_c(g(\text{out}), \text{mix}_b(y, y'))$ \Comment{classification loss (5)}
\end{algorithmic}
\end{algorithm}
We compare AlignMixup with SOTA methods, training R-18 on CIFAR-100 as discussed in subsection 4.2. At inference, ID examples are test
images from CIFAR-100, while OOD examples are test images from LSUN [64] and Tiny-ImageNet, resizing OOD examples to 32 × 32 to match the resolution of ID images [65]. We also use test images from CIFAR-100 with Uniform and Gaussian noise as OOD samples. Uniform is drawn from \( U(0,1) \) 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 achieving 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.

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


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-
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.

<table>
<thead>
<tr>
<th>Iterations (i)</th>
<th>0</th>
<th>10</th>
<th>20</th>
<th>50</th>
<th>100</th>
<th>200</th>
<th>500</th>
<th>1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>AlignMixup</td>
<td>80.98</td>
<td>80.96</td>
<td>81.31</td>
<td>81.42</td>
<td>81.71</td>
<td>81.50</td>
<td>81.34</td>
<td>81.28</td>
</tr>
</tbody>
</table>

attention. In this case, we only normalize either the rows or columns in (7) once, such that \( P1 = 1/r \) (when \( A \) aligned to \( A' \)) or \( P^T 1 = 1/r \) (when \( A' \) aligned to \( A \)). We observe that while AlignMixup outperforms the best baseline–StyleCutMix (80.66)–in all cases, it performs best for \( i = 100 \) iterations.