A. Proofs

A.1. Proof of Lemma 1

Lemma 1. If ψ, φ satisfy pairwise submodularity and β, γ ∈ R+, then βψ + γφ satisfies pairwise submodularity.

Proof. By the assumption, β, γ ∈ R+, and by using pairwise submodularity of ψ and φ, ∀x, y ∈ L,

\[(βψ + γφ)(x, x) + (βψ + γφ)(y, y) = β(ψ(x, x) + φ(y, x)) + γ(φ(x, x) + φ(y, x)) ≤ β(ψ(x, y) + φ(y, x)) + γ(φ(x, y) + φ(y, x)) = (βψ + γφ)(x, y) + (βψ + γφ)(y, x)\].

B. Analysis of Algorithms

B.1. Comparison Experiments for Algorithm 1

Figure 1 and Figure 2 show the comparison results of Algorithm 1 with the exact Hungarian algorithm on 100 random samples per each vertex size n. Note that, the size of a transport plan is n × n. In the simulation, we generate a random cost matrix C' = C − s(x)zT, where s(x) is sampled from a uniform distribution and the mask z is sampled from a bernoulli distribution with probability p = 0.5. In the case of n = 1024, Algorithm 1 is about 8.6 times faster than the exact algorithm, with relative error of 0.0005. For comparison, we use lapjv solver from library\(^1\) as the exact optimizer, which to the best of our knowledge is the fastest solver.

\(^1\)https://github.com/berhane/LAP-solvers

Figure 1. Comparison of average execution time (log-log scale) to solve Equation (5) between the exact solver (black) and Algorithm 1 (blue). Execution times are mean of 100 trials.
In addition to the simulation test, we train classifiers with ten different random seeds to compare the Top-1 accuracy between using the Hungarian algorithm and Algorithm 1. In this experiment, we train WRN28-10 on CIFAR100 for 400 epochs. In summary, the mean difference of the Top-1 accuracy is $-0.025$, with a standard deviation of 0.239. Besides, we perform a two-sided paired t-test to check the statistical insignificance. As a result, T-statistics is -0.105 with a P-value of 0.919, which means there is no evidence for a statistical difference between the two methods.

B.2. Convergence of Alternating Minimization

We solve the optimization problem of a mask and transport plans via alternating minimization for one-cycle for computational efficiency. In this subsection, we analyze the convergence property of the alternating algorithm by optimizing 1,000 CIFAR100 image pairs with various numbers of regions. As a result, we observe that the most of optimal masks of the images are not changed after the first cycle, i.e., comparison between one-cycle and multi-cycle), i.e., \(\beta, \gamma, \eta, \xi = (1.2, 0.5, 0.2, 0.8)\). For adversarial training, we use 10/255 epsilon-ball with the step size \(\tau\) of 12/255 according to the step size protocol of Wong et al. (2020).

C.2. Tiny-ImageNet

We follow the training protocol of Verma et al. (2019) except for the learning schedule. Verma et al. (2019) train Tiny-ImageNet for 2000 epochs with an initial learning rate of 0.1, but we train models for 1200 epochs with an initial learning rate of 0.2. As in the CIFAR-100 experiment, we use SGD and decay learning rate by factor 0.1 at epochs 600 and 900. We set momentum as 0.9 and weight decay as 0.0001. In case of mixing weight \(\lambda\), for Input mixup and Manifold mixup, we follow the setting \(\alpha = 0.2\) described in Manifold mixup (Verma et al., 2019). For CutMix, we choose \(\alpha = 0.2\), which showed the best performance among \(0.2, 0.5, 1.0\), and for Puzzle Mix, we use \(\alpha = 1.0\). In the Tiny-ImageNet experiment, we use \((\beta, \gamma, \eta, \xi) = (1.2, 0.5, 0.2, 0.8)\), which is the same with the CIFAR-100 experiment. However, we apply regularization using clean input with \(\lambda_{\text{clean}} = 1\) for all experiments regarding Puzzle Mix, and use the same initial learning rate of 0.2 for Puzzle Mix (half).

C.3. ImageNet

For ImageNet, we modify the training protocol in Wong et al. (2020) and train models for 100 epochs. The learning rate starts from 0.5, linearly increases to 1.0 for 8 epochs, and linearly decreases to 0.125 until 15th epoch. Then, the learning rate jumps to 0.2 and linearly decreases to 0.02 until 40th epoch, 0.002 until 65th epoch, 0.0002 until 90th epoch, and 0.00002 until 100th epoch. In addition, we resize images to 160×160 for the first 15 epochs, and use images pre-resized to 352×352 for the next 85 epochs following the prescription in Wong et al. (2020). Mixing distribution parameter \(\alpha\) is 0.2, 0.2, 1.0, 1.0 each for Input mixup, Manifold mixup, CutMix, Puzzle Mix, which follows the settings.
of the original papers. In the case of Manifold mixup, there is no experiments on ImageNet, and thus, we tune $\alpha$ in $[0.2, 1.0]$ and report the best result. In the case of ImageNet, we use hyperparameter $(\beta, \gamma, \eta, \xi) = (1.5, 0.5, 0.2, 0.8)$ and apply clean input regularization of $\lambda_{\text{clean}} = 1$ for the first 40 epochs.

C.4. Hyperparameter Sensitivity

We analyze the sensitivity of the hyperparameters with WRN28-10 on CIFAR100 trained for 400 epochs. In detail, we sweep hyperparameters one by one while others being fixed, and calculate the mean and standard deviation of Top-1 accuracy. Note that, the hyperparameter setting $(\beta, \gamma, \eta, \xi)$ of the main experiment is $(1.2, 0.5, 0.2, 0.8)$ which achieves 15.95% Top-1 test error.

Table 1 shows the mean Top-1 error rates and standard deviations of various hyperparameter settings. From the table, we can find that there exists a well of hyperparameters of which performance is superior to that of baselines (manifold mixup: 17.40%).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Mean Top-1 Error(%) (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>[0.8, 1.6]</td>
<td>16.19% (0.22)</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>[0.0, 1.0]</td>
<td>16.43% (0.20)</td>
</tr>
<tr>
<td>$\eta$</td>
<td>[0.1, 0.35]</td>
<td>16.37% (0.18)</td>
</tr>
<tr>
<td>$\xi$</td>
<td>[0.4, 1.0]</td>
<td>16.25% (0.27)</td>
</tr>
</tbody>
</table>

Table 1. Mean Top-1 error rates and standard deviations (SD) for various hyperparameter settings on CIFAR 100 with WRN28-10. For $\beta$, $\gamma$, and $\xi$, we sweep the range with 0.1 step size, and for $\eta$, we sweep the range with 0.05 step size.

D. Effect of Adversarial Training

D.1. Trade-off between Generalization and Adversarial Robustness

Since adversarial training increases the adversarial robustness at the expense of clean accuracy (Madry et al., 2017), we introduced adversarial probability $p$, a probability of whether to add adversarial perturbation or not, to control the intensity of adversarial training. Table 2 shows the inverse relationship between clean error and FGSM error.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Range</th>
<th>Mean Top-1 Error(%) (SD)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\beta$</td>
<td>[0.8, 1.6]</td>
<td>16.19% (0.22)</td>
</tr>
<tr>
<td>$\gamma$</td>
<td>[0.0, 1.0]</td>
<td>16.43% (0.20)</td>
</tr>
<tr>
<td>$\eta$</td>
<td>[0.1, 0.35]</td>
<td>16.37% (0.18)</td>
</tr>
<tr>
<td>$\xi$</td>
<td>[0.4, 1.0]</td>
<td>16.25% (0.27)</td>
</tr>
</tbody>
</table>

Table 2. Top-1 / Top-5 / FGSM error rates on Tiny-ImageNet dataset for PreActResNet18 trained with various adversarial probability $p$.

is more robust against the PGD attack than the existing mixup methods.

Figure 3. Adversarial robustness of various mixup methods against the PGD attack.

E. Puzzle Mix Qualitative Results

E.1. Effect of Prior and Smoothness Term

In this section, we provide Puzzle Mix results while adjusting the hyperparameters associated with the optimal mask. In Figure 4, we visualize how the Puzzle Mix images change by increasing the mixing weight $\lambda$, and in Figure 5, we visualize how the results change as we increase the coefficients of the smoothness terms, $\beta$ and $\gamma$.

E.2. More Samples

In this section, we provide Puzzle Mix results with various resolutions of the optimal mask and transport. Figure 6 visualizes the Puzzle Mix results along with the given inputs.
References


Figure 4. Puzzle Mix images with increasing mixing weight $\lambda$. 
Figure 5. Puzzle Mix images with increasing smoothness coefficient $\beta$ and $\gamma$. 
Figure 6. Various Puzzle Mix image samples. Each row consists of input image 1 (left), Puzzle Mix image (middle), and input image 2 (right).