Co-Mixup: Saliency Guided Joint Mixup with Supermodular Diversity

Pavel Shvets

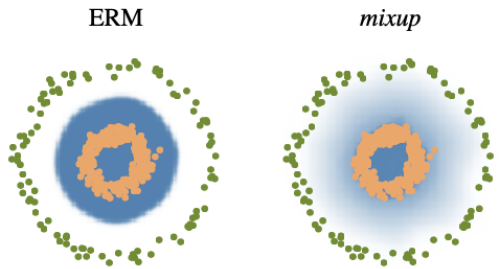
CMC MSU

April 13, 2021

Input Mixup[4]

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

Figure 1: Illustration of *mixup*, which converges to ERM as $\alpha \to 0$.

Expected Risk:

$$R_\delta = \int l(f(x), y) \mathrm{d}P$$

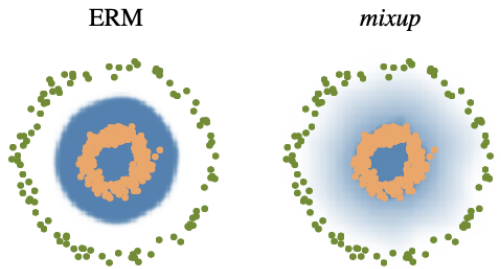
ERM:

$$P(x, y) \approx P_\delta(x, y) = \frac{1}{n} \sum_{i=1}^{n} \delta(x = x_i, y = y_i)$$

Input Mixup[4]

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

Figure 1: Illustration of *mixup*, which converges to ERM as $\alpha \to 0$.

Expected Risk:

$$R_\delta = \int l(f(x), y) \mathrm{d}P$$

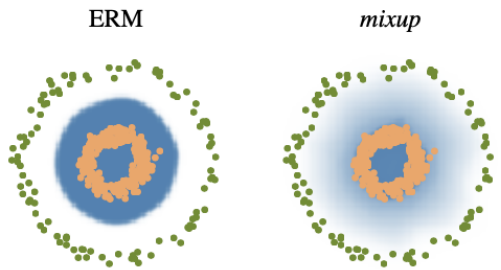
VRM:

$$P(x, y) \approx P_\nu(\tilde{x}, \tilde{y}) = \frac{1}{n} \sum_{i=1}^{n} \nu(\tilde{x}, \tilde{y} | x_i, y_i)$$

Input Mixup[4]

```
# y1, y2 should be one-hot vectors
for (x1, y1), (x2, y2) in zip(loader1, loader2):
    lam = numpy.random.beta(alpha, alpha)
    x = Variable(lam * x1 + (1. - lam) * x2)
    y = Variable(lam * y1 + (1. - lam) * y2)
    optimizer.zero_grad()
    loss(net(x), y).backward()
    optimizer.step()
```

(a) One epoch of *mixup* training in PyTorch.



(b) Effect of *mixup* ($\alpha = 1$) on a toy problem. Green: Class 0. Orange: Class 1. Blue shading indicates $p(y = 1|x)$.

Figure 1: Illustration of *mixup*, which converges to ERM as $\alpha \to 0$.

$$\mu(\tilde{x}, \tilde{y}|x_i, y_i) = \frac{1}{n} \sum_{j}^{n} \mathop{\mathbb{E}}_{\lambda} \left[\delta\left(\tilde{x} = \lambda x_i + (1-\lambda)x_j, \tilde{y} = \lambda y_i + (1-\lambda)y_j\right)\right]$$

$$\lambda \sim \mathrm{Beta}(\alpha, \alpha), \alpha \in [0, \infty)$$

$$h(x, x') = \lambda x + (1-\lambda)x'$$

CutMix[3]

```
lam = np.random.beta(args.beta, args.beta)
rand_index = torch.randperm(input.size()[0]).cuda()
target_a = target
target_b = target[rand_index]
bbx1, bby1, bbx2, bby2 = rand_bbox(input.size(), lam)
input[:, :, bbx1:bbx2, bby1:bby2] = input[rand_index, :, bbx1:bbx2, bby1:bby2]
# adjust lambda to exactly match pixel ratio
lam = 1 - ((bbx2 - bbx1) * (bby2 - bby1) / (input.size()[-1] * input.size()[-2]))
# compute output
output = model(input)
loss = criterion(output, target_a) * lam + criterion(output, target_b) * (1. - lam)
```

```
def rand_bbox(size, lam):
    W = size[2]
    H = size[3]
    cut_rat = np.sqrt(1. - lam)
    cut_w = np.int(W * cut_rat)
    cut_h = np.int(H * cut_rat)

    # uniform
    cx = np.random.randint(W)
    cy = np.random.randint(H)

    bbx1 = np.clip(cx - cut_w // 2, 0, W)
    bby1 = np.clip(cy - cut_h // 2, 0, H)
    bbx2 = np.clip(cx + cut_w // 2, 0, W)
    bby2 = np.clip(cy + cut_h // 2, 0, H)

    return bbx1, bby1, bbx2, bby2
```

$$h(x, x') = \mathbb{1}_B \odot x + (1 - \mathbb{1}_B) \odot x'$$

PuzzleMix[2]

$$
\begin{aligned}
\underset{\substack{z\in\mathcal{L}^n \\ \Pi_0,\Pi_1\in\{0,1\}^{n\times n}}}{\text{minimize}} \quad & -\|(1-z)\odot\Pi_0^\intercal s(x_0)\|_1 \\
& -\|z\odot\Pi_1^\intercal s(x_1)\|_1 \\
& +\beta\sum_{(i,j)\in\mathcal{N}}\psi(z_i,z_j)+\gamma\sum_{(i,j)\in\mathcal{N}}\phi_{i,j}(z_i,z_j) \\
& -\eta\sum_i\log p(z_i)+\xi\sum_{k=0,1}\langle\Pi_k,C\rangle \\
\text{subject to} \quad & \Pi_k 1_n=1_n,\ \Pi_k^\intercal 1_n=1_n \quad \text{for } k=0,1.
\end{aligned}
\tag{3}
$$

PuzzleMix[2]

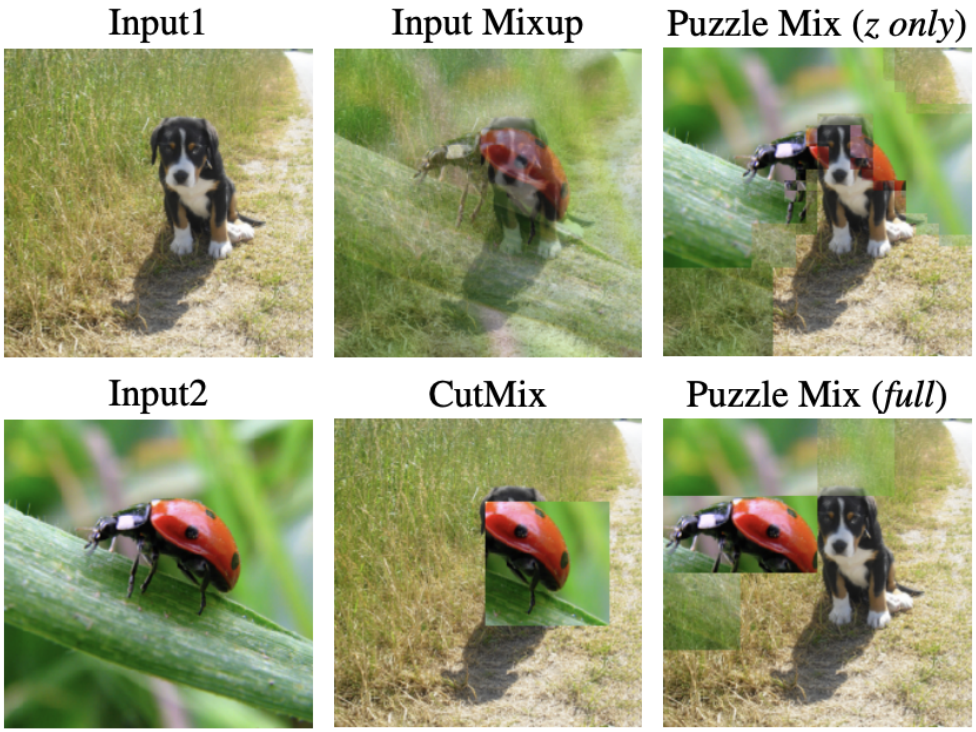


Figure 1. A visual comparison of the mixup methods. Puzzle Mix ensures to contain sufficient saliency information while preserving the local statistics of each input.

Co-Mixup[1]

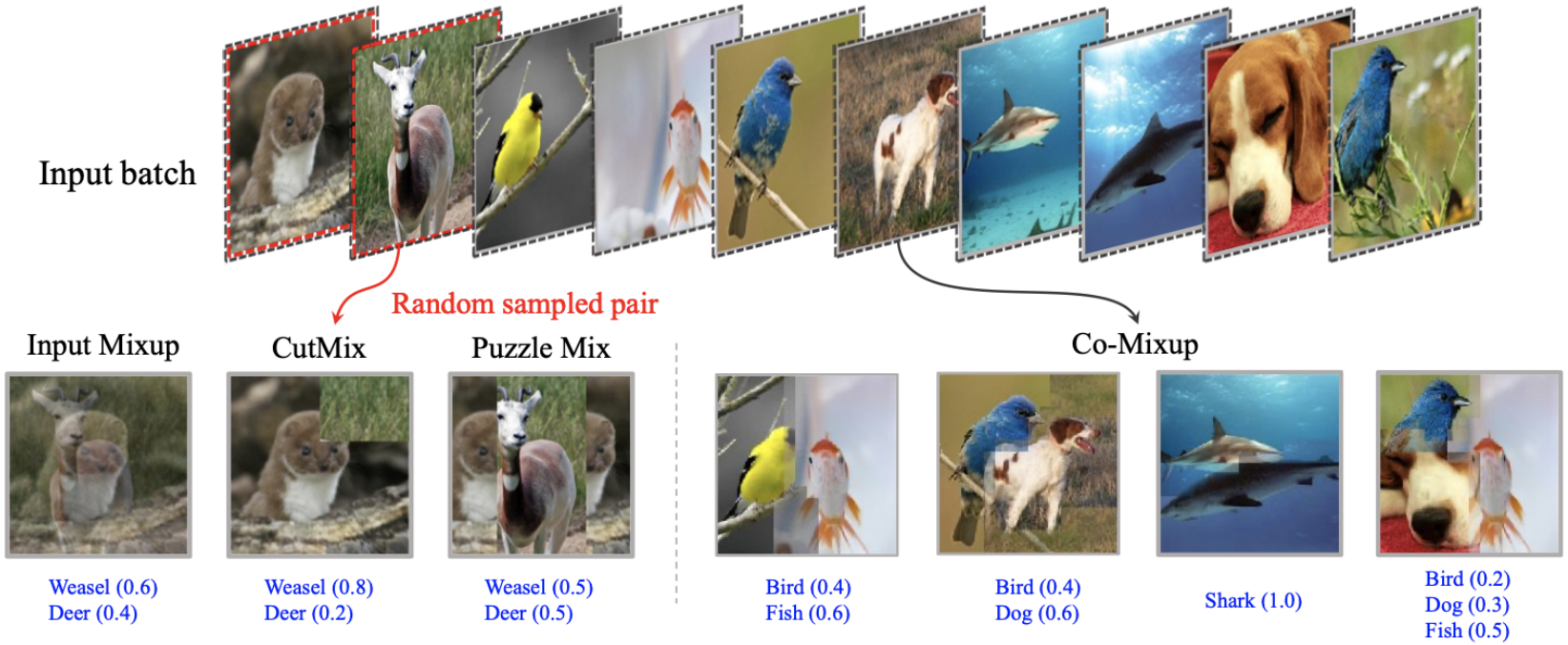


Figure 1: Example comparison of existing mixup methods and the proposed Co-Mixup. We provide more samples in Appendix H.

Co-Mixup[1]

In this work, we extend the existing mixup functions as $h : \mathcal{X}^m \to \mathcal{X}^{m'}$ which performs mixup on a collection of input data and returns another collection. Let $x_B \in \mathbb{R}^{m \times n}$ denote the batch of input data in matrix form. Then, our proposed mixup function is

$$h(x_B) = \big(g(z_1 \odot x_B), \ldots, g(z_{m'} \odot x_B)\big),$$

where $z_j \in \mathcal{L}^{m \times n}$ for $j = 1, \ldots, m'$ with $\mathcal{L} = \{\frac{l}{L} \mid l = 0, 1, \ldots, L\}$ and $g : \mathbb{R}^{m \times n} \to \mathbb{R}^n$ returns a column-wise sum of a given matrix. Note that, the k^{th} column of z_j, denoted as $z_{j,k} \in \mathcal{L}^m$, can be interpreted as the mixing ratio among m inputs at the k^{th} location. Also, we enforce $\|z_{j,k}\|_1 = 1$ to maintain the overall statistics of the given input batch. Given the one-hot target labels $y_B \in \{0, 1\}^{m \times C}$ of the input data with C classes, we generate soft target labels for mixup data as $y_B^\intercal \tilde{o}_j$ for $j = 1, \ldots, m'$, where $\tilde{o}_j = \frac{1}{n} \sum_{k=1}^{n} z_{j,k} \in [0, 1]^m$ represents the input source ratio of the j^{th} mixup data. We train models to estimate the soft target labels by minimizing the cross-entropy loss.

Co-Mixup[1]

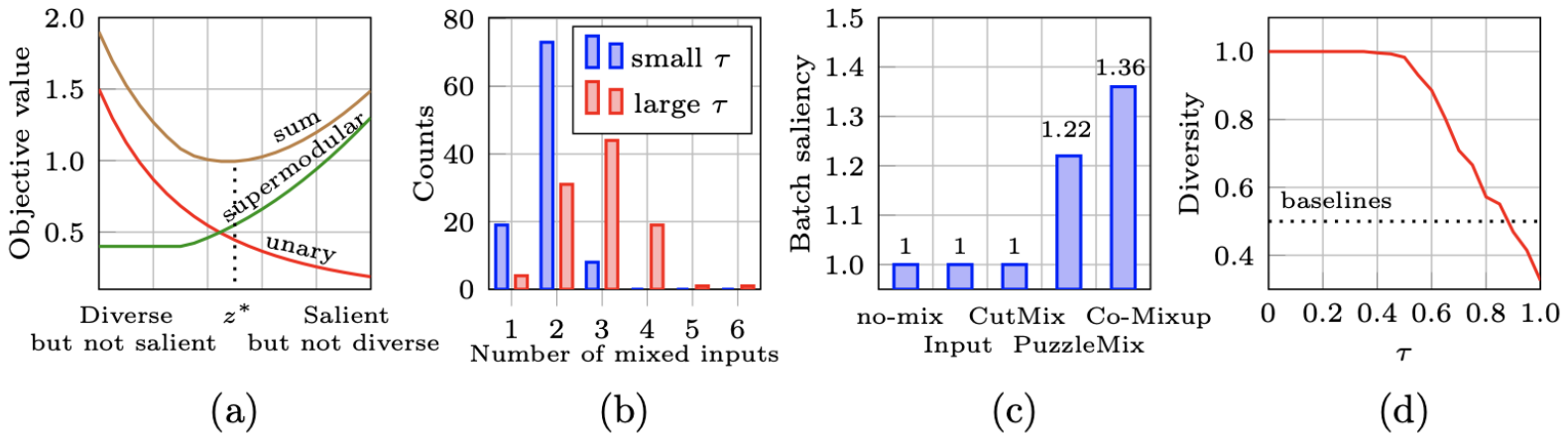


Figure 2: (a) Analysis of our BP optimization problem. The x-axis is a one-dimensional arrangement of solutions: The mixed output is more salient but not diverse towards the right and less salient but diverse on the left. The unary term (red) decreases towards the right side of the axis, while the supermodular term (green) increases. By optimizing the sum of the two terms (brown), we obtain the balanced output z^*. (b) A histogram of the number of inputs mixed for each output given a batch of 100 examples from the ImageNet dataset. As τ increases, more inputs are used to create each output on average. (c) Mean batch saliency measurement of a batch of mixup data using the ImageNet dataset. We normalize the saliency measure of each input to sum up to 1. (d) Diversity measurement of a batch of mixup data. We calculate the diversity as $1 - \sum_j \sum_{j' \neq j} \tilde{o}_j^\intercal \tilde{o}_{j'}/m$, where $\tilde{o}_j = o_j/\|o_j\|_1$. We can control the diversity among Co-Mixup data (red) and find the optimum by controlling τ.

Co-Mixup[1]

$$\sum_{j=1}^{m'}\sum_{k=1}^{n} c_k^\intercal z_{j,k} + \beta \sum_{j=1}^{m'} \sum_{(k,k')\in\mathcal{N}} (1 - z_{j,k}^\intercal z_{j,k'}) - \eta \sum_{j=1}^{m'}\sum_{k=1}^{n} \log p(z_{j,k}),$$

$$z^* = \underset{z_{j,k}\in\mathcal{L}^m,\ \|z_{j,k}\|_1=1}{\operatorname{argmin}} f(z),$$

where

$$f(z) := \sum_{j=1}^{m'}\sum_{k=1}^{n} c_k^\intercal z_{j,k} + \beta \sum_{j=1}^{m'} \sum_{(k,k')\in\mathcal{N}} (1 - z_{j,k}^\intercal z_{j,k'}) \tag{1}$$

$$+ \underbrace{\gamma \max\left\{\tau,\ \sum_{j=1}^{m'}\sum_{j'\neq j}^{m'} \left(\sum_{k=1}^{n} z_{j,k}\right)^\intercal A \left(\sum_{k=1}^{n} z_{j',k}\right)\right\}}_{=f_c(z)} - \eta \sum_{j=1}^{m'}\sum_{k=1}^{n} \log p(z_{j,k}).$$

Co-Mixup[1]

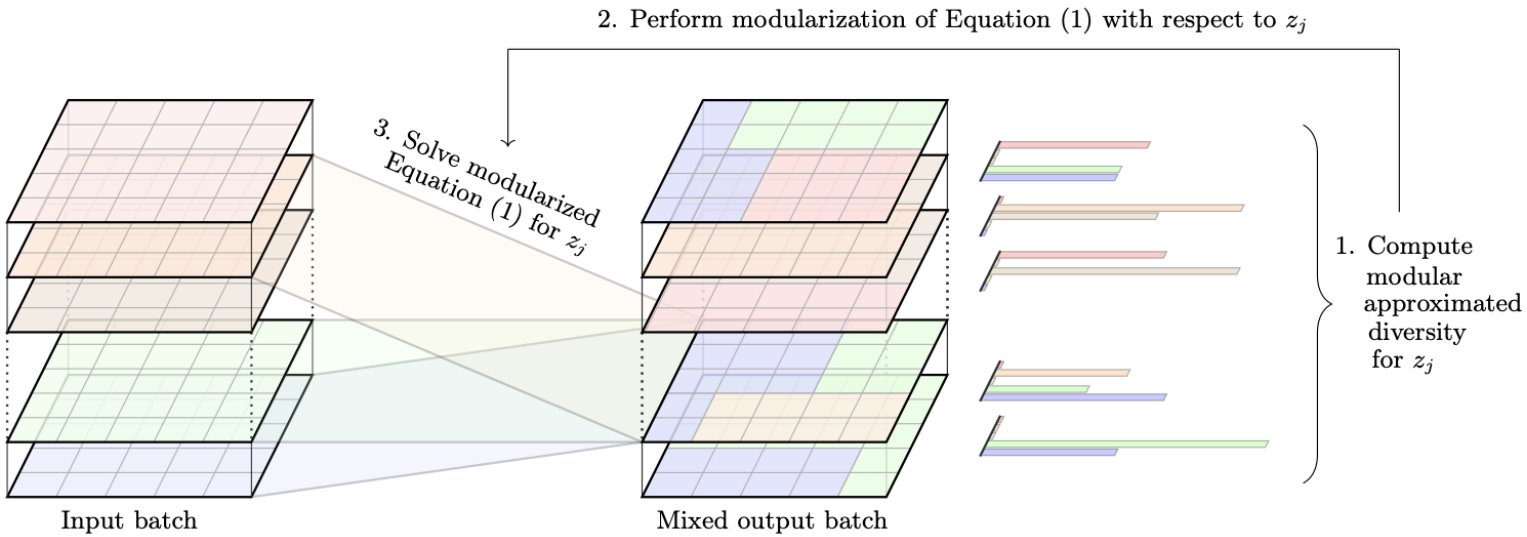


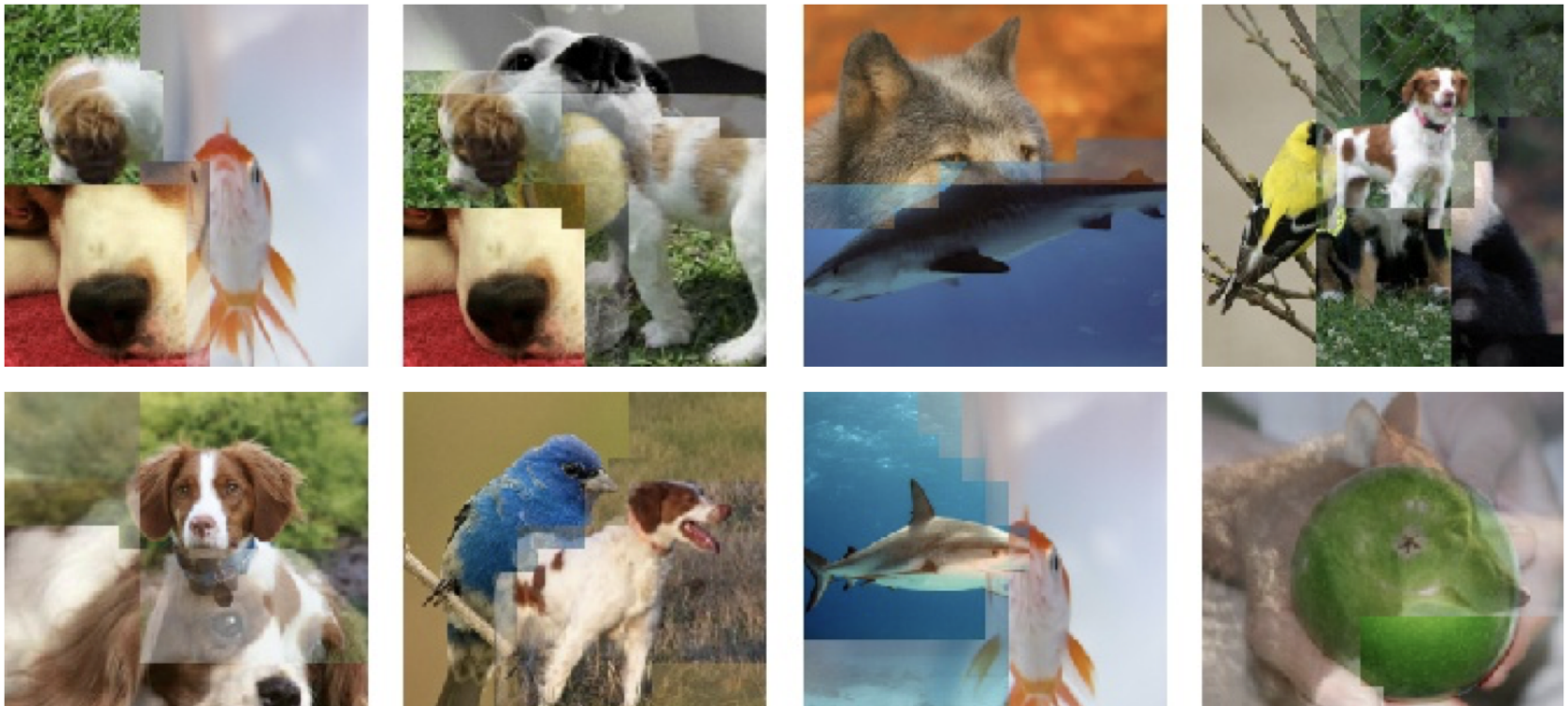
Figure 3: Visualization of the proposed mixup procedure. For a given batch of input data (left), a batch of mixup data (right) is generated, which mix-matches different salient regions among the input data while preserving the diversity among the mixup examples. The histograms on the right represent the input source information of each mixup data (o_j).

Dataset (Model)	Vanilla	Input	Manifold	CutMix	Puzzle Mix	Co-Mixup
CIFAR-100 (PreActResNet18)	23.59	22.43	21.64	21.29	20.62	**19.87**
CIFAR-100 (WRN16-8)	21.70	20.08	20.55	20.14	19.24	**19.15**
CIFAR-100 (ResNeXt29-4-24)	21.79	21.70	22.28	21.86	21.12	**19.78**
Tiny-ImageNet (PreActResNet18)	43.40	43.48	40.76	43.11	36.52	**35.85**
ImageNet (ResNet-50, 100 epochs)	24.03	22.97	23.30	22.92	22.49	**22.39**
Google commands (VGG-11)	4.84	3.91	3.67	3.76	3.70	**3.54**

Corruption type	Vanilla	Input	Manifold	CutMix	Puzzle Mix	Co-Mixup
Random replacement	41.63 (+17.62)	39.41 (+16.47)	39.72 (+16.47)	46.20 (+23.16)	39.23 (+16.69)	**38.77** (+16.38)
Gaussian noise	29.22 (+5.21)	26.29 (+3.35)	26.79 (+3.54)	27.13 (+4.09)	26.11 (+3.57)	**25.89** (+3.49)

# inputs for mixup	Input	Manifold	CutMix	Co-Mixup
# inputs = 2	22.43	21.64	21.29	19.87
# inputs = 3	23.03	22.13	22.01	
# inputs = 4	23.12	22.07	22.20	

Table 4: Top-1 error rates of mixup baselines with multiple mixing inputs on CIFAR-100 and PreActResNet18. We report the mean values of three different random seeds. Note that Co-Mixup optimally determines the number of inputs for each output by solving the optimization problem.



[1] Kim, Jang-Hyun, Choo, Wonho, Jeong, Hosan, and Song, Hyun Oh.
Co-mixup: Saliency guided joint mixup with supermodular diversity, 2021.

[2] Kim, Jang-Hyun, Choo, Wonho, and Song, Hyun Oh.
Puzzle mix: Exploiting saliency and local statistics for optimal mixup, 2020.

[3] Yun, Sangdoo, Han, Dongyoon, Oh, Seong Joon, Chun, Sanghyuk, Choe, Junsuk, and Yoo, Youngjoon.
Cutmix: Regularization strategy to train strong classifiers with localizable features, 2019.

[4] Zhang, Hongyi, Cisse, Moustapha, Dauphin, Yann N., and Lopez-Paz, David.
mixup: Beyond empirical risk minimization.
https://arxiv.org/pdf/1710.09412.pdf, 2018.