Co-Mixup: Saliency Guided Joint Mixup with Supermodular Diversity

Anonymous authors
Paper under double-blind review

Abstract

While deep neural networks show great performance on fitting to the training distribution, improving the networks' generalization performance to the test distribution and robustness to the sensitivity to input perturbations still remain as a challenge. Although a number of mixup based augmentation strategies have been proposed to partially address them, it remains unclear as to how to best utilize the supervisory signal within each input data for mixup from the optimization perspective. We propose a new perspective on batch mixup and formulate the optimal construction of a batch of mixup data maximizing the data saliency measure of each individual mixup data and encouraging the supermodular diversity among the constructed mixup data. This leads to a novel discrete optimization problem minimizing the difference between submodular functions. We also propose an efficient modular approximation based iterative submodular minimization algorithm for efficient mixup computation per each minibatch suitable for minibatch based neural network training. Our experiments show the proposed method achieves the state of the art generalization, calibration, and weakly supervised localization results compared to other mixup methods.

1 Introduction

Deep neural networks have been applied to a wide range of artificial intelligence tasks such as computer vision, natural language processing, and signal processing with remarkable performance (Ren et al., 2015; Devlin et al., 2018; Oord et al., 2016). However, it has been shown that neural networks have excessive representation capability and can even fit random data (Zhang et al., 2016). Due to these characteristics, the neural networks can easily overfit to training data and show a large generalization gap when tested on previously unseen data.

To improve the generalization performance of the neural networks, a body of research has been proposed to develop regularizers based on priors or to augment the training data with task-dependent transforms (Bishop, 2006; Cubuk et al., 2019). Recently, a new task-independent data augmentation technique, called *mixup*, has been proposed (Zhang et al., 2018). The original mixup, called *Input Mixup*, linearly interpolates a given pair of input data and can be easily applied to various data and tasks, improving the generalization performance and robustness of neural networks. Other mixup methods, such as *manifold mixup* (Verma et al., 2019) or *CutMix* (Yun et al., 2019), have also been proposed addressing different ways to mix a given pair of input data. *Puzzle Mix* (Kim et al., 2020) utilizes saliency information and local statistics to ensure mixup data to have rich supervisory signals.

However, these approaches only consider mixing a given random pair of input data and do not fully utilize the rich informative supervisory signal in training data including collection of object saliency, relative arrangement, etc. In this work, we simultaneously consider mix-matching different salient regions among all input data so that each generated mixup example accumulates as many salient regions from multiple input data as possible while ensuring diversity among the generated mixup examples. To this end, we propose a novel optimization problem that maximizes the saliency measure of each individual mixup example while encouraging diversity among them collectively. This formulation results in a novel discrete submodular-supermodular objective. We also propose a practical modular approximation

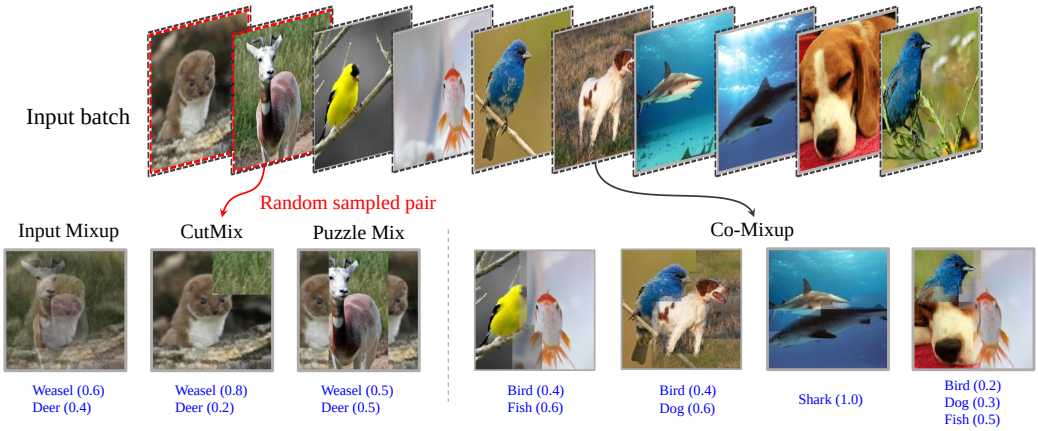

Figure 1: Example comparison of existing mixup methods and the proposed Co-Mixup. We provide more samples in supplementary material Section 8.

method for the supermodular term and present an efficient iterative submodular minimization algorithm suitable for minibatch-based mixup for neural network training. As illustrated in the Figure 1, while the proposed method, *Co-Mixup*, mix-matches the collection of salient regions utilizing inter-arrangements among input data, the existing methods do not consider the saliency information (Input Mixup & CutMix) or disassemble salient parts (Puzzle Mix).

We verify the performance of the proposed method by training classifiers on CIFAR-100, Tiny-ImageNet, ImageNet, and the Google commands dataset (Krizhevsky et al., 2009; Chrabaszcz et al., 2017; Deng et al., 2009; Warden, 2017). Our experiments show the models trained with Co-Mixup achieve the state of the performance compared to other mixup baselines. In addition to the generalization experiment, we conduct weakly-supervised object localization and robustness tasks and confirm Co-Mixup outperforms other mixup baselines.

2 Related works

Mixup Data augmentation has been widely used to prevent deep neural networks from over-fitting to the training data (Bishop, 1995). The majority of conventional augmentation methods generate new data by applying transformations depending on the data type or the target task (Cubuk et al., 2019). Zhang et al. (2018) proposed *mixup*, which can be independently applied to various data types and tasks, and improves generalization and robustness of deep neural networks. *Input mixup* (Zhang et al., 2018) linearly interpolates between two input data and utilizes the mixed data with the corresponding soft label for training. Following this work, *manifold mixup* (Verma et al., 2019) applies the mixup in the hidden feature space, and *CutMix* (Yun et al., 2019) suggests a spatial copy and paste based mixup strategy on images. Guo et al. (2019) trains an additional neural network to optimize a mixing ratio. *Puzzle Mix* (Kim et al., 2020) proposes a mixup method based on saliency and local statistics of the given data. In this paper, we propose a discrete optimization-based mixup method simultaneously finding the best combination of collections of salient regions among all input data while encouraging diversity among the generated mixup examples.

Saliency The seminal work from Simonyan et al. (2013) generates a saliency map using a pre-trained neural network classifier without any additional training of the network. Following the work, measuring the saliency of data using neural networks has been studied to obtain a more precise saliency map (Zhao et al., 2015; Wang et al., 2015) or to reduce the saliency computation cost (Zhou et al., 2016; Selvaraju et al., 2017). The saliency information is widely applied to the tasks in various domains, such as object segmentation or speech recognition (Jung and Kim, 2011; Kalinli and Narayanan, 2007).

Submodular-Supermodular optimization A submodular (supermodular) function is a set function with diminishing (increasing) returns property (Narasimhan and Bilmes, 2005). It is known that any set function can be expressed as the sum of a submodular and supermodular

function (Lovász, 1983), called BP function. Various problems in machine learning can be naturally formulated as BP functions (Fujishige, 2005), but it is known to be NP-hard (Lovász, 1983). Therefore, approximate algorithms based on modular approximations of submodular or supermodular terms have been developed (Iyer and Bilmes, 2012). Our formulation falls into a category of BP function consisting of smoothness function within a mixed output (submodular) and a diversity function among the mixup outputs (supermodular).

3 Preliminary

Existing mixup methods return $\{h(x_1, x_{i(1)}), \dots, h(x_m, x_{i(m)})\}$ for given input data $\{x_1, \dots, x_m\}$, where $h : \mathcal{X} \times \mathcal{X} \to \mathcal{X}$ is a mixup function and $(i(1), \dots, i(m))$ is a random permutation of the data indices. In the case of input mixup, $h(x, x')$ is $\lambda x + (1-\lambda)x'$, where $\lambda \in [0, 1]$ is a random mixing ratio. Manifold mixup applies input mixup in the hidden feature space, and CutMix uses $h(x, x') = \mathbb{1}_B \odot x + (1 - \mathbb{1}_B) \odot x'$, where $\mathbb{1}_B$ is a binary rectangular-shape mask for an image x and $\odot$ represents the element-wise product. Puzzle Mix defines $h(x, x')$ as $z \odot \Pi^\intercal x + (1-z) \odot \Pi'^\intercal x'$, where Π is a transport plan and z is a discrete mask. In detail, for $x \in \mathbb{R}^n$, $\Pi \in \{0, 1\}^n$ and $z \in \mathcal{L}^n$ for $\mathcal{L} = \{\frac{l}{L} \mid l = 0, 1, \dots, L\}$.

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

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

where $z_j \in \mathcal{L}^{m \times n}$ for $j = 1, \dots, m'$ with $\mathcal{L} = \{\frac{l}{L} \mid l = 0, 1, \dots, L\}$ and $g : \mathbb{R}^{m \times n} \to \mathbb{R}^n$ returns a column-wise sum of a given matrix. Note that, the kth 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 kth 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 $j = 1, \dots, m'$, 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 jth mixup data. We train models to estimate the soft target labels by minimizing the cross-entropy loss.

4 Method

4.1 Objective

Saliency Our main objective is to maximize the saliency measure of mixup data while maintaining the local smoothness of data, *i.e.*, spatially nearby patches in a natural image look similar, temporally adjacent signals have similar spectrum in speech, etc. (Kim et al., 2020). As we can see from CutMix in Figure 1, disregarding saliency can give a misleading supervisory signal by generating mixup data that does not match with the target soft label. While the existing mixup methods only consider the mixup between two inputs, we generalize the number of inputs m to any positive integer. Note, each kth location of outputs has m candidate sources from the inputs. We model the unary labeling cost as the negative value of the saliency, and denote the cost vector at the kth location as $c_k \in \mathbb{R}^m$. For the saliency measure, we calculate the gradient values of training loss with respect to the input and measure ℓ_2 norm of the gradient values across input channels (Simonyan et al., 2013; Kim et al., 2020). Note that this method does not require any additional architecture dependent modules for saliency calculation. In addition to the unary cost, we encourage adjacent locations to have similar labels for the smoothness of each mixup data. In summary, the objective can be formulated as follows:

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

where the prior p is given by $z_{j,k} \sim \frac{1}{L} Multi(L, \lambda)$ with $\lambda = (\lambda_1, \dots, \lambda_m) \sim Dirichlet(\alpha, \dots, \alpha)$, which is a generalization of the mixing ratio distribution of Zhang

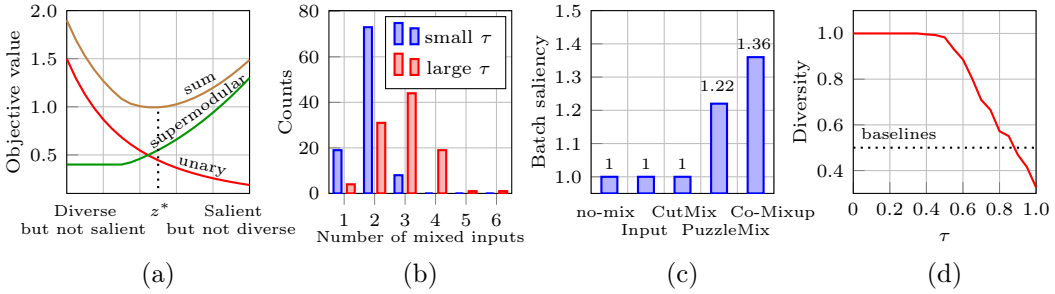

Figure 2: (a) Analysis of our BP optimization problem. The x-axis is a one-dimensional arrangement of solutions: The mixed output is more salient but not diverse towards the right and less salient but diverse on the left. The unary term (red) decreases towards the right side of the axis, while the supermodular 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 effectively control the diversity among Co-Mixup data (red) and find the optimum by controlling τ.

et al. (2018), and $\mathcal{N}$ denotes a set of adjacent locations (*i.e.*, neighboring image patches in vision, subsequent spectrums in speech, etc.).

Diversity Note that the naive generalization above leads to the identical outputs because the objective is separable and identical for each output. In order to obtain diverse mixup outputs, we model a similarity penalty between outputs. First, we represent the input source information of the jth output by aggregating assigned labels as $\sum_{k=1}^n z_{j,k}$. For simplicity, let us denote $\sum_{k=1}^n z_{j,k}$ as o_j. Then, we measure the similarity between o_j's by using the inner-product on $\mathbb{R}^m$. In addition to the input source similarity between outputs, we model the compatibility between input sources, represented as a symmetric matrix $A_c \in \mathbb{R}_+^{m \times m}$. Specifically, $A_c[i_1, i_2]$ quantifies the degree to which input i_1 and i_2 are suitable to be mixed together. In summary, we use inner-product on $A = (1-\omega)I + \omega A_c$ for $\omega \in [0, 1]$, resulting in a supermodular penalty term. Note that, by minimizing $\langle o_j, o_{j'} \rangle_A = o_j^\intercal A o_{j'}$, $\forall j \neq j'$, we penalize output mixup examples with similar input sources and encourage each individual mixup examples to have high compatibility within. In this work, we measure the distance between locations of salient objects in each input and use the distance matrix $A_c[i, j] = \|\text{argmax}_k s_i[k] - \text{argmax}_k s_j[k]\|_1$, where s_i is the saliency map of the ith input and k is a location index (*e.g.*, k is a 2-D index for image data). From now on, we denote this inner-product term as the *compatibility* term.

Over-penalization The conventional mixup methods perform mixup as many as the number of examples in a given mini-batch. In our setting, this is the case when $m = m'$. However, the compatibility penalty between outputs is influenced by the pigeonhole principle. For example, suppose the first output consists of two inputs. Then, the inputs must be used again for the remaining $m' - 1$ outputs, or only $m - 2$ inputs can be used. In the latter case, the number of available inputs $(m - 2)$ is less than the outputs $(m' - 1)$, and thus, the same input must be used more than twice. Empirically, we found that the remaining compatibility term above over-penalizes the optimization so that a substantial portion of outputs are returned as singletons without any mixup. To mitigate the over-penalization issue, we apply clipping to the compatibility penalty term. Specifically, we model the objective so that no extra penalty would occur when the compatibility among outputs is below a certain level.

Now we present our main objective as following:

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

where

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

In Figure 2, we describe the properties of the BP optimization problem of Equation (1) and statistics of the resulting mixup data. Next, we verify the supermodularity of the compatibility term. We first extend the definition of the submodularity of a multi-label function as follows (Windheuser et al., 2012).

Definition 1. *For a given label set $\mathcal{L}$, a function $s : \mathcal{L}^m \times \mathcal{L}^m \to \mathbb{R}$ is pairwise submodular, if $\forall x, x' \in \mathcal{L}^m$, $s(x,x) + s(x',x') \leq s(x,x') + s(x',x)$. A function s is pairwise supermodular, if $-s$ is pairwise submodular.*

Proposition 1. *The compatibility term f_c in Equation (1) is pairwise supermodular for every pair of $(z_{j_1,k}, z_{j_2,k})$ if A is positive semi-definite.*

Proof. See supplementary material, Section 2.1. □

Finally note that, $A = (1-\omega)I + \omega A_c$, where A_c is a symmetric matrix. By using spectral decomposition, A_c can be represented as $UDU^\intercal$, where D is a diagonal matrix and $U^\intercal U = UU^\intercal = I$. Then, $A = U((1-\omega)I + \omega D)U^\intercal$, and thus for small $\omega > 0$, we can guarantee A to be positive semi-definite.

4.2 Algorithm

Our main objective consists of modular (*unary, prior*), submodular (*smoothness*), and supermodular (*compatibility*) terms. To optimize the main objective, we employ the submodular-supermodular procedure by iteratively approximating the supermodular term as a modular function (Narasimhan and Bilmes, 2005). Note that z_j represents the labeling of the jth output and o_j represents the aggregated input source information of the jth output, $\sum_{k=1}^{n} z_{j,k}$. Before introducing our algorithm, we present the following property about the simpler case without clipping.

Proposition 2. *The compatibility term f_c without clipping is modular with respect to z_j.*

Proof. Note, A is a positive symmetric matrix by the definition. Then, for an index j_0, we can represent f_c without clipping in terms of o_{j_0} as $\sum_{j=1}^{m'} \sum_{j'=1, j' \neq j}^{m'} o_j^\intercal A o_{j'} = 2 \sum_{j=1, j \neq j_0}^{m'} o_j^\intercal A o_{j_0} + \sum_{j=1, j \neq j_0}^{m'} \sum_{j'=1, j' \notin \{j_0, j\}}^{m'} o_j^\intercal A o_{j'} = (2 \sum_{j=1, j \neq j_0}^{m'} A o_j)^\intercal o_{j_0} + c = v_{\text{-}j_0}^\intercal o_{j_0} + c$, where $v_{\text{-}j_0} \in \mathbb{R}^m$ and $c \in \mathbb{R}$ are values independent with o_{j_0}. Finally, $v_{\text{-}j_0}^\intercal o_{j_0} + c = \sum_{k=1}^{n} v_{\text{-}j_0}^\intercal z_{j_0,k} + c$ is a modular function of z_{j_0}. □

By Proposition 2, we can apply a submodular minimization algorithm to optimize the objective with respect to z_j when there is no clipping. Thus, we can optimize the main objective without clipping in coordinate descent fashion (Wright, 2015). For the case with clipping, we modularize the supermodular compatibility term under the following criteria:

1. The modularized function value should increase as the compatibility across outputs increases.
2. The modularized function should not apply an extra penalty for the compatibility below a certain level.

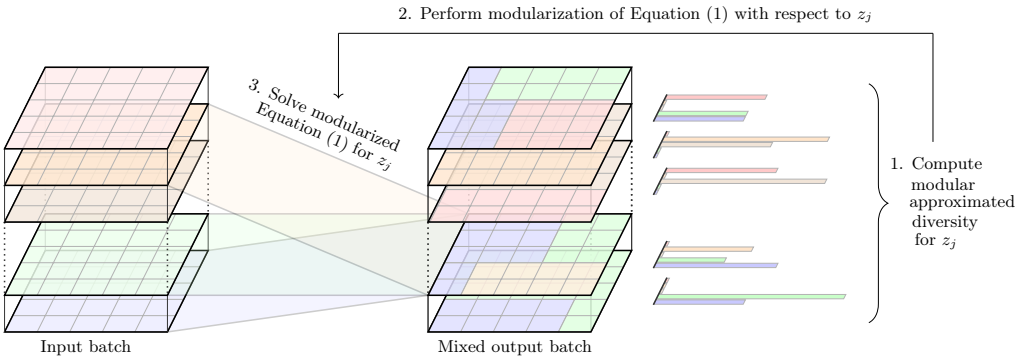

Figure 3: Visualization of the proposed mixup procedure. For a given batch of input data (left), a batch of mixup data (right) is generated, which mix-matches different salient regions among the input data while preserving the diversity among the mixup examples. The histograms on the right represent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

Borrowing the notation from the proof in Proposition 2, for an index j, $f_c(z) = \max\{\tau, v_{\text{-}j}^\intercal o_j + c\} = \max\{\tau - c, v_{\text{-}j}^\intercal o_j\} + c$. Note, $o_j = \sum_{k=1}^{n} z_{j,k}$ represents the input source information of the jth output and $v_{\text{-}j} = 2\sum_{j'=1, j' \neq j}^{m'} A o_{j'}$ encodes the status of the other outputs. Thus, we can interpret the supermodular term as a penalization of each label of o_j in proportion to the corresponding $v_{\text{-}j}$ value (criterion 1), but not for the compatibility below $\tau - c$ (criterion 2). As a modular function which satisfies the criteria above, we use the following function:

$$f_c(z) \approx \max\{\tau', v_{\text{-}j}\}^\intercal o_j \quad \text{for } \exists \tau' \in \mathbb{R}. \tag{2}$$

Note that, by satisfying the criteria above, the modular function reflects the diversity and over-penalization desiderata described in Section 4.1. We illustrate the proposed mixup procedure with the modularized diversity penalty in Figure 3.

Proposition 3. *The modularization given by Equation* (2) *satisfies the criteria above.*

Proof. See supplementary material, Section 2.2. □

By applying the modular approximation described in Equation (2) to f_c in Equation (1), we can iteratively apply a submodular minimization algorithm to obtain the final solution as described in Algorithm 1. In detail, each step can be performed as follows: 1) Conditioning the main objective f on the current values except z_j, denoted as $f_j(z_j) = f(z_j; z_{1:j-1}, z_{j+1:m'})$. 2) Modularization of the compatibility term of f_j as Equation (2), resulting in a submodular function $\tilde{f}_j$. We denote the modularization operator as Φ, *i.e.*, $\tilde{f}_j = \Phi(f_j)$. 3) Applying a submodular minimization algorithm to $\tilde{f}_j$. Please refer to supplementary material Section 3 for implementation details.

Algorithm 1 Iterative submodular minimization

Initialize z as $z^{(0)}$.
Let $z^{(t)}$ denote a solution of the tth step.
Φ: modularization operator based on Equation (2).
for $t = 1, \dots, T$ **do**
  for $j = 1, \dots, m'$ **do**
    $f_j^{(t)}(z_j) \coloneqq f(z_j; z_{1:j-1}^{(t)}, z_{j+1:m'}^{(t-1)})$.
    $\tilde{f}_j^{(t)} = \Phi(f_j^{(t)})$.
    Solve $z_j^{(t)} = \operatorname{argmin} \tilde{f}_j^{(t)}(z_j)$.
  end for
end for
return $z^{(T)}$

Analysis Narasimhan and Bilmes (2005) proposed a modularization strategy for general supermodular set functions, and apply a submodular minimization algorithm that can monotonically decrease the original BP objective. However, the proposed Algorithm 1 based on Equation (2) is much more suitable for minibatch based mixup for neural network training than the set modularization proposed by Narasimhan and Bilmes (2005) in terms of complexity and modularization variance due to randomness. For simplicity, let us assume

each $z_{j,k}$ is an m-dimensional one-hot vector. Then, our problem is to optimize $m'n$ one-hot m-dimensional vectors.

To apply the set modularization method, we need to assign each possible value of $z_{j,k}$ as an element of $\{1, 2, \ldots, m\}$. Then the supermodular term in Equation (1) can be interpreted as a set function with $m'nm$ elements, and to apply the set modularization, $O(m'nm)$ sequential evaluations of the supermodular term are required. In contrast, Algorithm 1 calculates $v_{\text{-}j}$ in Equation (2) in only $O(m')$ time per each iteration. In addition, each modularization step of the set modularization method requires a random permutation of the $m'nm$ elements. In this case, the optimization can be strongly affected by the randomness from the permutation step. As a result, the optimal labeling of each $z_{j,k}$ from the compatibility term is strongly influenced by the random ordering undermining the interpretability of the algorithm. Please refer to supplementary material Section 4.2 for empirical comparison between Algorithm 1 and the method by Narasimhan and Bilmes (2005).

5 Experiments

We evaluate our proposed mixup method on generalization, weakly supervised object localization, calibration, and robustness tasks. First, we compare the generalization performance of the proposed method against baselines by training classifiers on CIFAR-100 (Krizhevsky et al., 2009), Tiny-ImageNet (Chrabaszcz et al., 2017), ImageNet (Deng et al., 2009), and the Google commands speech dataset (Warden, 2017). Next, we test the localization performance of classifiers following the evaluation protocol of Qin and Kim (2019). We also measure calibration error (Guo et al., 2017) of classifiers to verify Co-Mixup successfully alleviates the over-confidence issue by Zhang et al. (2018). In Section 5.4, we evaluate the robustness of the classifiers on the test dataset with background corruption in response to the recent problem raised by Lee et al. (2020) that deep neural network agents often fail to generalize to unseen environments. Finally, we perform a sensitivity analysis of Co-Mixup and provide the results in supplementary material Section 6.3.

5.1 Classification

We first train PreActResNet18 (He et al., 2016), WRN16-8 (Zagoruyko and Komodakis, 2016), and ResNeXt29-4-24 (Xie et al., 2017) on CIFAR-100 for 300 epochs. We use stochastic gradient descent with an initial learning rate of 0.2 decayed by factor 0.1 at epochs 100 and 200. We set the momentum as 0.9 and add a weight decay of 0.0001. With this setup, we train a vanilla classifier and reproduce the mixup baselines (Zhang et al., 2018; Verma et al., 2019; Yun et al., 2019; Kim et al., 2020), which we denote as *Vanilla, Input, Manifold, CutMix, Puzzle Mix* in the experiment tables. Note that we use identical hyperparameters regarding Co-Mixup over all of the experiments with different models and datasets, which are provided in supplementary material Section 5.

Table 1 shows Co-Mixup significantly outperforms all other baselines in Top-1 error rate. Co-Mixup achieves 19.87% in Top-1 error rate with PreActResNet18, outperforming the best baseline by 0.75%. We further test Co-Mixup on different models (WRN16-8 & ResNeXt29-4-24) and verify Co-Mixup improves Top-1 error rate over the best performing baseline.

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)	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**

Table 1: Top-1 error rate on various datasets and models. For CIFAR-100, we train each model with three different random seeds and report the mean error.

We further test Co-Mixup on other datasets; Tiny-ImageNet, ImageNet, and the Google commands dataset (Table 1). For Tiny-ImageNet, we train PreActResNet18 for 1200 epochs

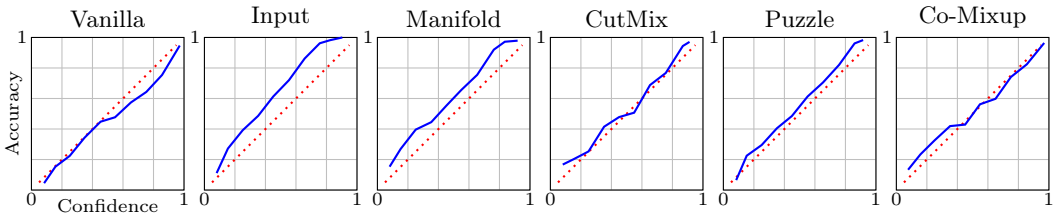

Figure 4: Confidence-Accuracy plots for classifiers on CIFAR-100. From the figure, the Vanilla network shows over-confident predictions, whereas other mixup baselines tend to have under-confident predictions. We can find that Co-Mixup has best-calibrated predictions.

following the training protocol of Kim et al. (2020). As a result, Co-Mixup consistently improves Top-1 error rate over baselines by 0.67%. In the ImageNet experiment, we follow the experimental protocol provided in Puzzle Mix (Kim et al., 2020), which trains ResNet-50 (He et al., 2015) for 100 epochs, with exactly the same learning rate scheduling and image pre-processing of Kim et al. (2020). As a result, Co-Mixup outperforms all of the baselines in Top-1 error rate. We further test Co-Mixup on the speech domain with the Google commands dataset and VGG-11 (Simonyan and Zisserman, 2014). We provide a detailed experimental setting and dataset description in supplementary material Section 6.1. From Table 1, we confirm that Co-Mixup is the most effective in the speech domain as well.

5.2 Localization

We compare weakly supervised object localization (WSOL) performance of classifiers trained on ImageNet (in Table 1) to demonstrate that our mixup method better guides a classifier to focus on salient regions. We test the localization performance using CAM (Zhou et al., 2016), a WSOL method using a pre-trained classifier. We evaluate localization performance following the evaluation protocol in Qin and Kim (2019), with binarization threshold 0.25 in CAM. Table 2 summarizes the WSOL performance of various mixup methods, which shows that our proposed mixup method outperforms other baselines.

5.3 Calibration

We evaluate the expected calibration error (ECE) (Guo et al., 2017) of classifiers trained on CIFAR-100. Note, ECE is calculated by the weighted average of the absolute difference between the confidence and accuracy of a classifier. As shown in Table 2, the Co-Mixup classifier has the lowest calibration error among baselines. From Figure 4, we find that other mixup baselines tend to have *under-confident* predictions resulting in higher ECE values even than *Vanilla* network (also pointed out by Wen et al. (2020)), whereas Co-Mixup has best-calibrated predictions resulting in relatively 48% less ECE value. We provide more figures and results with other datasets in the supplementary material Section 6.2.

Task	Vanilla	Input	Manifold	CutMix	Puzzle Mix	Co-Mixup
Localization (Acc. %) (↑)	54.36	55.07	54.86	54.91	55.22	**55.32**
Calibration (ECE %) (↓)	3.9	17.7	13.1	5.6	7.5	**1.9**

Table 2: WSOL results on ImageNet and ECE (%) measurements of CIFAR-100 classifiers.

5.4 Robustness

In response to the recent problem raised by Lee et al. (2020) that deep neural network agents often fail to generalize to unseen environments, we consider the situation where the statistics of the foreground object, such as color or shape, is unchanged, but with the corrupted (or replaced) background. In detail, we consider the following operations: 1) replacement with another image and 2) adding Gaussian noise. We use ground-truth bounding boxes to separate the foreground from the background, and then apply the previous operations

independently to obtain test datasets. We provide a detailed description of datasets in supplementary material Section 7.

With the test datasets described above, we evaluate the robustness of the pre-trained classifiers. As shown in Table 3, Co-Mixup shows significant performance gains at various background corruption tests compared to the other mixup baselines. For each corruption case, the classifier trained with Co-Mixup outperforms the others in Top-1 error rate with the performance margins of 2.86% and 3.33% over the Vanilla model.

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)

Table 3: Top-1 error rates of various mixup methods for background corrupted ImageNet validation set. The values in the parentheses indicate the error rate increment by corrupted inputs compared to clean inputs.

5.5 Baselines with multiple inputs

To further investigate the effect of the number of inputs for the mixup in isolation, we conduct an ablation study on baselines using multiple mixing inputs. For fair comparison, we use Dirichlet$(\alpha, \ldots, \alpha)$ prior for the mixing ratio distribution and select the best performing α in $\{0.2, 1.0, 2.0\}$. Note that we overlay multiple boxes in the case of CutMix. Table 4 reports the classification test errors on CIFAR-100 with PreActResNet18. From the table, we find that mixing multiple inputs decreases the performance gains of each mixup baseline. These results demonstrate that mixing multiple inputs could lead to possible degradation of the performance and support the necessity of considering saliency information and diversity as in Co-Mixup.

# 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.

6 Conclusion

We presented Co-Mixup for optimal construction of a batch of mixup examples by finding the best combination of salient regions among a collection of input data while encouraging diversity among the generated mixup examples. This leads to a discrete optimization problem minimizing a novel submodular-supermodular objective. In this respect, we present a practical modular approximation and iterative submodular optimization algorithm suitable for minibatch based neural network training. Our experiments on generalization, weakly supervised object localization, and robustness against background corruption show Co-Mixup achieves the state of the art performance compared to other mixup baseline methods.

References

C. M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural computation*, 7(1):108–116, 1995.

C. M. Bishop. *Pattern recognition and machine learning.* springer, 2006.

P. Chrabaszcz, I. Loshchilov, and F. Hutter. A downsampled variant of imagenet as an alternative to the cifar datasets. *arXiv preprint arXiv:1707.08819*, 2017.

E. D. Cubuk, B. Zoph, D. Mane, V. Vasudevan, and Q. V. Le. Autoaugment: Learning augmentation strategies from data. *In CVPR*, 2019.

J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and F. F. Li. Imagenet: a large-scale hierarchical image database. *In CVPR*, 2009.

J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

S. Fujishige. *Submodular functions and optimization.* Elsevier, 2005.

C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks. *In ICML*, 2017.

H. Guo, Y. Mao, and R. Zhang. Mixup as locally linear out-of-manifold regularization. *In AAAI*, 2019.

K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *In CVPR*, 2015.

K. He, X. Zhang, S. Ren, and J. Sun. Identity mappings in deep residual networks. *In ECCV*, 2016.

R. Iyer and J. Bilmes. Algorithms for approximate minimization of the difference between submodular functions, with applications. *arXiv preprint arXiv:1207.0560*, 2012.

C. Jung and C. Kim. A unified spectral-domain approach for saliency detection and its application to automatic object segmentation. *IEEE Transactions on Image Processing*, 21(3):1272–1283, 2011.

O. Kalinli and S. S. Narayanan. A saliency-based auditory attention model with applications to unsupervised prominent syllable detection in speech. In *Eighth Annual Conference of the International Speech Communication Association*, 2007.

J.-H. Kim, W. Choo, and H. O. Song. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. *In ICML*, 2020.

A. Krizhevsky, G. Hinton, et al. *Learning multiple layers of features from tiny images.* Citeseer, 2009.

K. Lee, K. Lee, J. Shin, and H. Lee. Network randomization: A simple technique for generalization in deep reinforcement learning. *In ICLR*, 2020.

L. Lovász. Submodular functions and convexity. In *Mathematical Programming The State of the Art*, pages 235–257. Springer, 1983.

M. Narasimhan and J. A. Bilmes. A submodular-supermodular procedure with applications to discriminative structure learning. *UAI*, 2005.

A. v. d. Oord, S. Dieleman, H. Zen, K. Simonyan, O. Vinyals, A. Graves, N. Kalchbrenner, A. Senior, and K. Kavukcuoglu. Wavenet: A generative model for raw audio. *arXiv preprint arXiv:1609.03499*, 2016.

Z. Qin and D. Kim. Rethinking softmax with cross-entropy: Neural network classifier as mutual information estimator. *arXiv preprint arXiv:1911.10688*, 2019.

S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *In NeurIPS*, 2015.

R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. *In ICCV*, 2017.

K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

K. Simonyan, A. Vedaldi, and A. Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

V. Verma, A. Lamb, C. Beckham, A. Najafi, I. Mitliagkas, A. Courville, D. Lopez-Paz, and Y. Bengio. Manifold mixup: Better representations by interpolating hidden states. *In ICML*, 2019.

L. Wang, H. Lu, X. Ruan, and M.-H. Yang. Deep networks for saliency detection via local estimation and global search. *In CVPR*, 2015.

P. Warden. *URL https://research.googleblog.com/2017/08/launching-speech-commands-dataset.html.*, 2017.

Y. Wen, G. Jerfel, R. Muller, M. W. Dusenberry, J. Snoek, B. Lakshminarayanan, and D. Tran. Improving calibration of batchensemble with data augmentation. *In ICML Workshop on Uncertainty and Robustness in Deep Learning*, 2020.

T. Windheuser, H. Ishikawa, and D. Cremers. Generalized roof duality for multi-label optimization: Optimal lower bounds and persistency. *In ECCV*, 2012.

S. J. Wright. Coordinate descent algorithms. *Mathematical Programming*, 151(1):3–34, 2015.

S. Xie, R. Girshick, P. Dollár, Z. Tu, and K. He. Aggregated residual transformations for deep neural networks. pages 1492–1500, 2017.

S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. *In ICCV*, 2019.

S. Zagoruyko and N. Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

C. Zhang, S. Bengio, M. Hardt, B. Recht, and O. Vinyals. Understanding deep learning requires rethinking generalization. *arXiv preprint arXiv:1611.03530*, 2016.

H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz. mixup: Beyond empirical risk minimization. *In ICLR*, 2018.

R. Zhao, W. Ouyang, H. Li, and X. Wang. Saliency detection by multi-context deep learning. *In CVPR*, 2015.

B. Zhou, A. Khosla, A. Lapedriza, A. Oliva, and A. Torralba. Learning deep features for discriminative localization. *In CVPR*, 2016.