Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup

Jang-Hyun Kim [1 2]   **Wonho Choo** [1 2]   **Hyun Oh Song** [1 2]

Abstract

While deep neural networks achieve great performance on fitting the training distribution, the learned networks are prone to overfitting and are susceptible to adversarial attacks. In this regard, a number of mixup based augmentation methods have been recently proposed. However, these approaches mainly focus on creating previously unseen virtual examples and can sometimes provide misleading supervisory signal to the network. To this end, we propose Puzzle Mix, a mixup method for explicitly utilizing the saliency information and the underlying statistics of the natural examples. This leads to an interesting optimization problem alternating between the multi-label objective for optimal mixing mask and saliency discounted optimal transport objective. Our experiments show Puzzle Mix achieves the state of the art generalization and the adversarial robustness results compared to other mixup methods on CIFAR-100, Tiny-ImageNet, and ImageNet datasets, and the source code is available at https://github.com/snu-mllab/PuzzleMix.

1. Introduction

Deep neural network models are the bedrock of modern AI tasks such as object recognition, speech, natural language processing, and reinforcement learning. However, these models are known to memorize the training data and make overconfident predictions often resulting in degraded generalization performance on test examples (Srivastava et al., 2014; Zhang et al., 2016). Furthermore, the problem is exacerbated when the models are evaluated on examples under slight distribution shift (Ben-David et al., 2010).

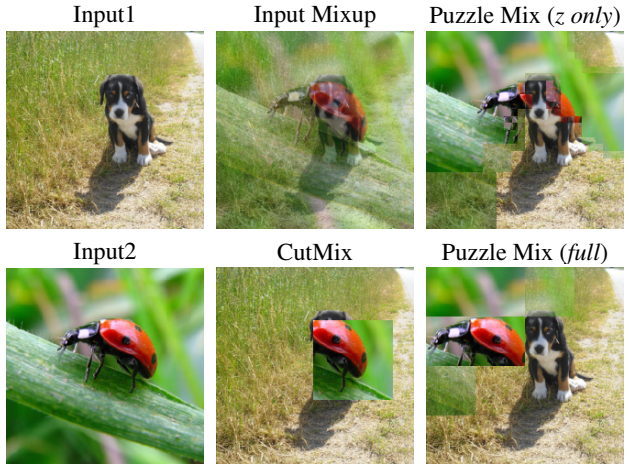To this end, data augmentation approaches aim to alleviate

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.

some of these issues by improving the model generalization performance (He et al., 2015; DeVries & Taylor, 2017). Recently, a line of research called *mixup* has been proposed. These methods mainly focus on creating previously unseen virtual mixup examples via convex combination or local replacement of data for training (Zhang et al., 2018; Verma et al., 2019; Yun et al., 2019; Guo et al., 2019).

However, the underlying data domains contain rich regional saliency information (*i.e.* foreground objects in vision, prominent syllables in speech, informative textual units in language) (Simonyan et al., 2013; Kalinli & Narayanan, 2007; Erkan & Radev, 2004) and exhibit local regularity structure far from random matrices of numbers (Huang & Mumford, 1999; Zhang et al., 2017; Smith, 2003). Thus, completely disregarding these aspects of data could lead to creating mixup examples which could misguide the training model and undermine the generalization performance.

Motivated by this intuition, we propose Puzzle Mix, a mixup method for explicitly leveraging the saliency information and the underlying local statistics of natural examples. Our proposed method jointly seek to find (1) the optimal mask for deciding how much of the two inputs to reveal versus conceal in the given region and for (2) the transport for finding the optimal moves in order to maximize the exposed saliency under the mask. The optimization process is remi-

---

[1]Department of Computer Science and Engineering, Seoul National University, Seoul, Korea [2]Neural Processing Research Center. Correspondence to: Hyun Oh Song <hyunoh@snu.ac.kr>.

niscent of the sliding block puzzle and thus the name Puzzle Mix. Additionally, we impose the objective to respect the various underlying local statistics encouraging the optimization to preserve the structural integrity of each data. The proposed method alternates between finding the optimal mask and optimizing the transport plans, and efficiently generates the mixup examples in a mini-batch stochastic gradient descent setting.

Furthermore, our method allows us to incorporate adversarial training without any computation overhead. Adversarial training is a method for training a robust model resistant to adversarial attacks via optimization (Madry et al., 2017). We adapt the fast adversarial training method from Wong et al. (2020) and stochastically include the adversarially perturbed examples with random restarts for robustness.

Our results on CIFAR-100, Tiny-ImageNet, and ImageNet datasets show significant improvement both in the generalization task and in the adversarial robustness over existing mixup methods by a large margin.

2. Related Works

Data augmentation Methods that implement data augmentation aim to regularize the models from overfitting to the training distribution and improve the generalization performance by generating virtual training examples in the vicinity of the given training dataset (Bishop, 2006). Some of the most commonly used data augmentation techniques are random cropping, horizontal flipping (Krizhevsky et al., 2012), and adding random noise (Bishop, 1995). Recently, a data augmentation method called AugMix is proposed to improve both the generalization performance and the corruption robustness (Hendrycks et al., 2020). Our method is complementary to these techniques and could be used in conjunction in order to further increase the generalization and robustness performance.

Mixup Input mixup creates virtual training examples by linearly interpolating two input data and corresponding one-hot labels (Zhang et al., 2018). The method induces models to have smoother decision boundaries and reduces overfitting to the training data. Manifold mixup extends this concept from input space to feature space (Verma et al., 2019). Also, Guo et al. (2019) proposed an adaptive mixup method, which improves Input mixup by preventing the generation of improper mixup data. Yun et al. (2019) proposed CutMix which implants a random rectangular region of the input into another. However, these methods can generate improper examples by randomly removing important regions of the data, which may mislead the neural network (see Figure 1). Our mixup method aims to prevent these issues by utilizing the saliency signal while preserving the local properties of the input data.

Saliency Simonyan et al. (2013) detects object saliency by computing gradients of a pre-trained deep neural network. Subsequently, other methods were introduced to obtain more precise saliency (Zhao et al., 2015; Wang et al., 2015). However, these methods require modifying the pre-trained network or training new models to compute the saliency. Zhou et al. (2016) and Selvaraju et al. (2017) proposed methods with the reduced computation cost but at the cost of saliency resolution. We follow the method from Simonyan et al. (2013), which does not require any modification to the model, to compute the saliency map. The saliency information has been used in various fields of machine learning (Ren et al., 2013; Wei et al., 2017).

Optimal transport A transport plan that moves a given distribution to another at the minimal cost is called the optimal transport (Villani, 2008). Also, the optimal transport with discrete domain can be represented as a linear program or an assignment problem (Munkres, 1957; Villani, 2008). The optimal transport problem is widely applied in various applications areas such as color transfer (Rabin et al., 2014) and domain adaptation (Courty et al., 2016). We formulate a binary transport problem for the optimal move, which maximizes the exposed saliency under the mask.

3. Preliminaries

Let us define $x \in \mathcal{X}$ to be an input data and $y \in \mathcal{Y}$ be its output label. Let $\mathcal{D}$ be the distribution over $\mathcal{X} \times \mathcal{Y}$. In mixup based data augmentation method, the goal is to optimize the model's loss $\ell : \mathcal{X} \times \mathcal{Y} \times \Theta \to \mathbb{R}$ given the data mixup function $h(\cdot)$ and the mixing distribution q as below.

$$\underset{\theta}{\text{minimize}} \; \underset{(x_0,y_0),(x_1,y_1)\in\mathcal{D}}{\mathbb{E}} \; \underset{\lambda \sim q}{\mathbb{E}} \; \ell(h(x_0,x_1), g(y_0,y_1);\theta), \quad (1)$$

where the label mixup function is $g(y_0, y_1) = (1-\lambda)y_0 + \lambda y_1$. Input mixup uses $h(x_0, x_1) = (1-\lambda)x_0 + \lambda x_1$. Manifold mixup employs $h(x_0, x_1) = (1-\lambda)f(x_0) + \lambda f(x_1)$ for some hidden representation f. CutMix defines $h(x_0, x_1) = (1 - \mathbb{1}_B) \odot x_0 + \mathbb{1}_B \odot x_1$ for a binary rectangular mask $\mathbb{1}_B$, where $B = [r_x, r_x + r_w] \times [r_y, r_y + r_h]$ with $\lambda = \frac{r_w r_h}{WH}$ and $\odot$ represents the element-wise product. In other words, B is a randomly chosen rectangle covering λ proportion of the input. We propose the following mixup function,

$$h(x_0, x_1) = (1 - z) \odot \Pi_0^\mathsf{T} x_0 + z \odot \Pi_1^\mathsf{T} x_1, \quad (2)$$

where z_i represents a mask in $[0, 1]$ with mixing ratio $\lambda = \frac{1}{n} \sum_i z_i$. Π_0 and Π_1 represent $n \times n$ transportation plans of the corresponding data with n dimensions. Π_{ij} encodes how much mass moves from location i to j after the transport. From now on, we omit the dependence of y and θ from the loss function ℓ for clarity. Table 1 summarizes various mixup functions described above. We begin Section 4 with the formal desiderata for our mixup function and the corresponding optimization objective.
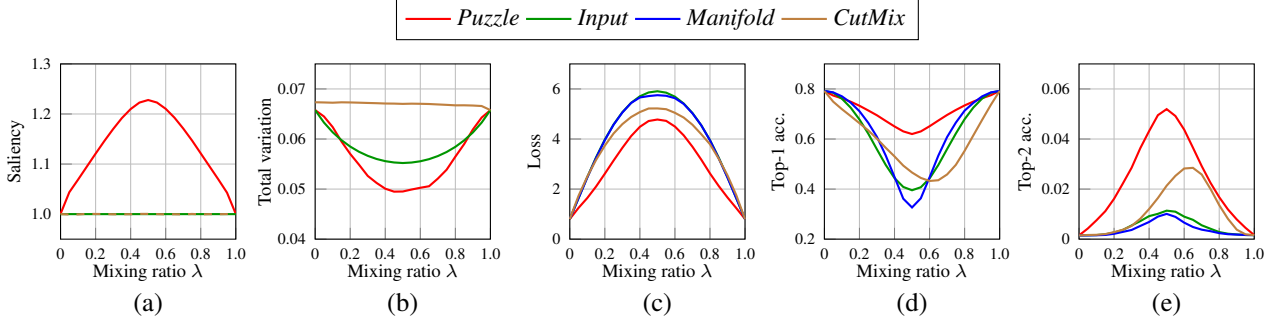
Figure 2. (a) Mixed saliency $||h(s(x_0), s(x_1))||_1$. Note the saliency map of each input $s(x_k)$ is normalized to sum up to 1. (b) Total variation of mixed data. (c) Cross entropy loss of mixup data and the corresponding soft-label evaluated by the vanilla classifier (ResNet18). (d) Top-1 prediction accuracy of mixed data. Prediction is counted as correct if the Top-1 prediction belongs to $\{y_0, y_1\}$. (e) Top-2 prediction accuracy of mixed data. Prediction is counted as correct if the Top-2 predictions are equal to $\{y_0, y_1\}$. Manifold mixup is omitted in (a) and (b) as manifold mixup generates mixup examples in the hidden space not in the input space.

Method	Mixup function $h(x_0, x_1)$
Input mixup	$(1 - \lambda)x_0 + \lambda x_1$
Manifold mixup	$(1 - \lambda)f(x_0) + \lambda f(x_1)$
CutMix	$(1 - \mathbb{1}_B) \odot x_0 + \mathbb{1}_B \odot x_1$
Puzzle Mix	$(1 - z) \odot \Pi_0^\mathsf{T} x_0 + z \odot \Pi_1^\mathsf{T} x_1$

Table 1. Summary of various mixup functions.

4. Methods

Our goal is to maximally utilize the saliency information of each input while respecting the underlying local statistics of the data. First, in order to maximally utilize the saliency information, we seek to find the optimal mixing mask z and the optimal transport plans Π under the following criteria.

1. • Given a pair of transported data and a specific region, the mask z should optimally reveal more salient data of the two while masking the less salient one in the given region.

2. • Given a data x and the mask z, the transport Π should find the optimal moves that would maximize the saliency of the revealed portion of the data.

The criteria above motivates us to maximize for $(1 - z) \odot \Pi_0^\mathsf{T} s(x_0) + z \odot \Pi_1^\mathsf{T} s(x_1)$. Note, we denote the saliency of the input x as $s(x)$ which is computed by taking ℓ_2 norm of the gradient values across input channels. Figure 2 (a) shows the proposed mixup function well preserves the saliency information after mixup. Second, in order to respect the underlying local statistics of the data (Huang & Mumford, 1999; Zhang et al., 2017; Smith, 2003), we consider the following criteria.

• The saliency information can be noisy, which could lead to a suboptimal solution. Therefore, we add spatial regularization terms ψ and $\phi_{i,j}$ to control the smoothness of the mask and regional smoothness of the result-

ing mixed example. Figure 2 (b) compares the local smoothness measured in total variation.

• We ensure the structural integrity within each data is generally preserved by considering the transport cost C_{ij} (defined as the distance between the locations i and j). Also, to further ensure the local salient structure of the data is preserved without being dispersed across after the transport, we optimize for the binary transport plans as opposed to continuous plans.

Evaluation results on the pretrained vanilla classifier in Figure 2 (c), (d), (e) show our mixup examples have the smallest loss and the highest accuracy compared to other methods, verifying our intuitions above. Moreover, we optimize the main objective after down-sampling the saliency information $s(x)$ with average pooling to support multi-scale transport and masking. From now on, we denote n as the down-sampled dimension. In practice, we select the down-sampling resolution randomly per each mini-batch.

To optimize the mask z, we first discretize the range of the mask value. Let $\mathcal{L}$ denote the discretized range $\{\frac{t}{m} \mid t = 0, 1, \ldots, m\}$. In addition, to control the mixing ratio of given inputs, we add a prior term $p(z_i)$, which follows a binomial distribution. We now formalize the complete objective in Equation (3).

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

subject to $\Pi_k 1_n = 1_n, \ \Pi_k^\mathsf{T} 1_n = 1_n \quad \text{for } k = 0, 1.$
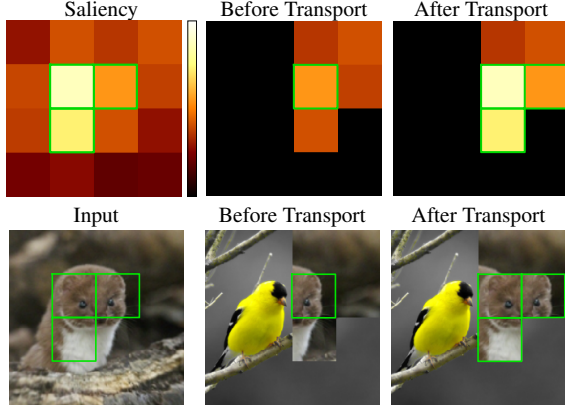
Figure 3. Illustration of Puzzle Mix process. After the transport, the salient regions (highlighted in green) replace the other regions, so that the salient information still remains after the mixup. The first row represents the saliency information after down-sampling, *i.e.*, $s(x)$, the masked saliency ($z \odot s(x)$), and the masked saliency after transport ($z \odot \Pi^\mathsf{T} s(x)$) respectively. The second row shows the corresponding data.

After solving the optimization problem in Equation (3), we obtain the mixed example $h(x_0, x_1) = (1 - z^*) \odot \Pi_0^{*\mathsf{T}} x_0 + z^* \odot \Pi_1^{*\mathsf{T}} x_1$ which is then used for the model training as in Equation (1). Figure 3 illustrates the mask z and the transport plan Π optimized with Equation (3).

We solve this optimization problem via alternating minimization through iterating first over z and then simultaneously over Π_0 and Π_1. In mixup augmentation, however, one needs to be able to efficiently generate the mixed examples as the generation process takes place per each mini-batch. Therefore, we optimize for one complete cycle of the alternating minimization, as repeated cycles require additional network evaluations, for efficiency. As for the initialization, we optimize the mask z with Π_k initialized as identity transport, and then optimize each Π_k with the previously solved z. We now formally discuss individual optimization problems in Section 4.1 and Section 4.2.

4.1. Optimizing Mask

Given Π_0 and Π_1, we seek to solve the following discrete optimization problem over z in Equation (4). The objective is to decide how to best mix the two transported inputs jointly based on the region saliency measure (unary), the label and data local smoothness (pairwise), and the mixing weight log prior (mix prior) criteria.

$$\underset{z \in \mathcal{L}^n}{\text{minimize}} \sum_i u_i(z_i) + \beta \sum_{(i,j) \in \mathcal{N}} \psi(z_i, z_j) \qquad (4)$$
$$+ \gamma \sum_{(i,j) \in \mathcal{N}} \phi_{i,j}(z_i, z_j) - \eta \sum_i \log p(z_i),$$
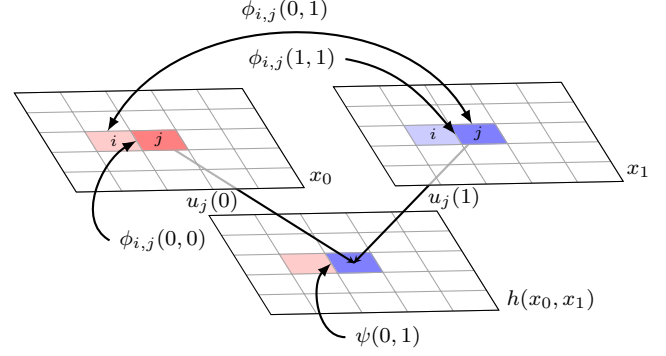

Figure 4. Visualization of different components in the mask optimization. Two rectangles in the top show the two inputs x_0 and x_1, and the rectangle in the bottom show the mixed output $h(x_0, x_1)$. Figure reproduced with permission from Julien Mairal.

where the unary term $u_i(z_i)$ is defined as $z_i(\Pi_0^\mathsf{T} s(x_0))_i + (1 - z_i)(\Pi_1^\mathsf{T} s(x_1))_i$. We define the neighborhood $\mathcal{N}$ as a set of adjacent regions, and use the following pairwise terms and the prior term. Figure 4 visualizes different components in Equation (4).

Definition 1. *(Label smoothness)* $\psi(z_i, z_j) := (z_i - z_j)^2$.

For data local smoothness, we measure the distance between input regions. Let d_p denote the distance function. First, we define pairwise terms under the binary case, $\mathcal{L} = \{0, 1\}$, and then extend them to the multi-label case.

Definition 2. *(Data local smoothness for binary labels)* *Let* $x_{k,i}$ *represent the* i^{th} *region of data* x_k. *Then,* $\phi_{i,j}^b(z_i, z_j) := d_p(x_{z_i,i}, x_{z_j,j})$.

The discrete optimization problem in Equation (4) is a type of multi-label energy minimization problem and can be efficiently solved via α-β swap algorithm (Boykov et al., 2001), which is based on the graph-cuts. In the binary label case, finding the minimum s-t cut in the graph returns an equivalent optimal solution if the pairwise term satisfies the submodularity condition (Kolmogorov & Zabih, 2004). In our problem, the pairwise term is $e_{i,j}(z_i, z_j) = \beta \psi(z_i, z_j) + \gamma \phi_{i,j}(z_i, z_j)$. We now assume that the function values of d_p are bounded in [0,1], which is generally satisfied when data values are bounded in [0,1].

Proposition 1. *Suppose* d_p *function is bounded in [0,1] and* $\phi = \phi^b$. *If* $\gamma \leq \beta$, *then* $e_{i,j}(z_i, z_j)$ *satisfies submodularity for* $z_i, z_j \in \{0, 1\}$.

Proof. $e(0,0) + e(1,1) = \gamma \phi_{i,j}(0,0) + \gamma \phi_{i,j}(1,1) \leq 2\gamma \leq 2\beta = \beta \psi(0,1) + \beta \psi(1,0) \leq e(0,1) + e(1,0)$. $\square$

For multi-label case, the α-β swap algorithm iteratively applies graph-cut as a sub-routine and converges to a
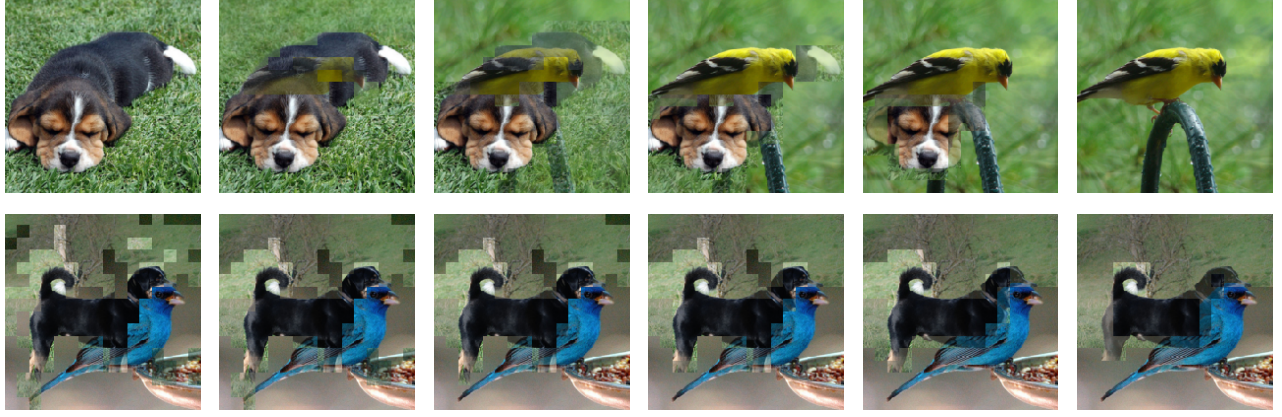
Figure 5. (Top row) Puzzle Mix images with increasing mixing weight λ. (Bottom row) Puzzle Mix images with increasing smoothness coefficients, β and γ. Note that the results are obtained without transport.

local-minimum if the pairwise term satisfies pairwise-submodularity (Schmidt & Alahari, 2011). We can guarantee pairwise-submodularity by slightly modifying $\phi_{i,j}^b$ as

$$\phi_{i,j}^{b'}(0,0) = \phi_{i,j}^b(0,0) + (\phi_{i,j}^b(0,1) + \phi_{i,j}^b(1,0))/2$$
$$\phi_{i,j}^{b'}(0,1) = \phi_{i,j}^b(0,1) + (\phi_{i,j}^b(0,0) + \phi_{i,j}^b(1,1))/2$$
$$\phi_{i,j}^{b'}(1,0) = \phi_{i,j}^b(1,0) + (\phi_{i,j}^b(0,0) + \phi_{i,j}^b(1,1))/2$$
$$\phi_{i,j}^{b'}(1,1) = \phi_{i,j}^b(1,1) + (\phi_{i,j}^b(0,1) + \phi_{i,j}^b(1,0))/2.$$

It is important to note that, $\phi_{i,j}^{b'}(1,0) + \phi_{i,j}^{b'}(0,1) - \phi_{i,j}^{b'}(0,0) - \phi_{i,j}^{b'}(1,1) = 0$.

Definition 3. *(Data local smoothness for the multi labels)*
$\phi_{i,j}(z_i, z_j) := z_i z_j \phi_{i,j}^{b'}(1,1) + z_i(1-z_j)\phi_{i,j}^{b'}(1,0) + (1 - z_i)z_j \phi_{i,j}^{b'}(0,1) + (1-z_i)(1-z_j)\phi_{i,j}^{b'}(0,0), \, \forall z_i, z_j \in \mathcal{L}.$

Proposition 2. *With $\phi_{i,j}$ defined as Definition 3, $e_{i,j}$ satisfies pairwise submodularity.*

Proof. We can represent $\phi_{i,j}$ as follows:

$$\phi_{i,j}(z_i, z_j)$$
$$= f(z_i, z_j)\frac{\phi_{i,j}^{b'}(1,0) + \phi_{i,j}^{b'}(0,1) - \phi_{i,j}^{b'}(0,0) - \phi_{i,j}^{b'}(1,1)}{2}$$
$$+ z_i \frac{\phi_{i,j}^{b'}(1,0) + \phi_{i,j}^{b'}(1,1) - \phi_{i,j}^{b'}(0,0) - \phi_{i,j}^{b'}(0,1)}{2}$$
$$+ z_j \frac{\phi_{i,j}^{b'}(0,1) + \phi_{i,j}^{b'}(1,1) - \phi_{i,j}^{b'}(0,0) - \phi_{i,j}^{b'}(1,0)}{2}$$
$$+ \phi_{i,j}^{b'}(0,0),$$

where $f(z_i, z_j) = (1 - z_i)z_j + z_i(1 - z_j)$. By definition $\phi_{i,j}^{b'}(1,0) + \phi_{i,j}^{b'}(0,1) - \phi_{i,j}^{b'}(0,0) - \phi_{i,j}^{b'}(1,1) = 0$, and thus, $\phi_{i,j}(z_i, z_j)$ can be represented as the form of $z_i \phi_{i,j}^{b'_1} + z_j \phi_{i,j}^{b'_2} + c$. Thus, $\forall x, y \in \mathcal{L}$, $\phi_{i,j}(x,y) + \phi_{i,j}(y,x) =$

$x\phi_{i,j}^{b'_1} + y\phi_{i,j}^{b'_2} + c + y\phi_{i,j}^{b'_1} + x\phi_{i,j}^{b'_2} + c = \phi_{i,j}(x,x) + \phi_{i,j}(y,y)$, which means $\phi_{i,j}$ satisfies pairwise submodularity.

By definition ψ satisfies pairwise submodularity, and by Lemma 1, $e_{i,j}$ satisfies pairwise submodularity. $\square$

Lemma 1. *If ψ, ϕ satisfies pairwise submodularity and β, $\gamma \in \mathbb{R}_+$, then $\beta\psi + \gamma\phi$ satisfies pairwise submodularity.*

Proof. See Supplementary A.1. $\square$

Finally, we use the prior term to control the ratio of inputs in the mixed output. For the given mixing weight λ, which represents the ratio of x_1 with respect to x_0, we define the prior term p to satisfy $\mathbb{E}_{z_i \sim p}[z_i] = \lambda, \forall i$. Specifically, for the label space $\mathcal{L} = \{\frac{t}{m} \mid t = 0, ..., m\}$, we define the prior term as $p(z_i = \frac{t}{m}) = \binom{m}{t}\lambda^t(1-\lambda)^{m-t}$ for $t = 0, 1, ..., m$. In other words, $z_i \sim \frac{1}{m}B(m, \lambda)$.

In Figure 5, we provide the resulted mixup images using the optimal mask from Equation (4). Specifically, we visualize how the Puzzle Mix images change by increasing the mixing weight λ and the coefficients of the smoothness terms, β and γ.

4.2. Optimizing Transport

After optimizing the mask z, we optimize the transportation plans for the input data under the optimal mask z^*. Our objective with respect to transportation plans is the following.

$$\underset{\Pi_0, \Pi_1 \in \{0,1\}^{n \times n}}{\text{minimize}} - \|(1 - z^*) \odot \Pi_0^\intercal s(x_0)\|_1$$
$$- \|(z^* \odot \Pi_1^\intercal s(x_1)\|_1$$
$$+ \xi \sum_{k=0,1} \langle \Pi_k, C \rangle$$
$$\text{subject to } \Pi_k 1_n = 1_n, \ \Pi_k^\intercal 1_n = 1_n \quad \text{for } k = 0, 1.$$

Note the problem is completely separable as two independent optimization problems of each Π_k. Let $s(x_1)_i$

Algorithm 1 Masked Transport

> **Input:** mask z^*, cost C', large value v
> Initialize $C^{(0)} = C'$, $t = 0$
> **repeat**
> $target = \mathrm{argmin}(C^{(t)}, \dim = 1)$
> $\Pi = 0_{n \times n}$
>   **for** $i = 0$ **to** $n - 1$ **do**
> $\Pi[i, target[i]] = 1$
>   **end for**
> $C_{conflict} = C^{(t)} \odot \Pi + v(1 - \Pi)$
> $source = \mathrm{argmin}(C_{conflict}, \dim = 0)$
> $\Pi_{win} = 0_{n \times n}$
>   **for** $j = 0$ **to** $n - 1$ **do**
> $\Pi_{win}[source[j], j] = 1$
>   **end for**
> $\Pi_{win} = \Pi_{win} \odot \Pi$
> $\Pi_{lose} = (1 - \Pi_{win}) \odot \Pi$
> $C^{(t+1)} = C^{(t)} + v\Pi_{lose}$
> $t = t + 1$
> **until** convergence
> **Return:** Π_{win}

---

denote the ith entry of the n-dimensional column vector $s(x_1)$. The term $\|z^* \odot \Pi_1^\mathsf{T} s(x_1)\|_1$ can be represented as $\sum_{i,j} z_j^* s(x_1)_i \Pi_{1 i,j} = \langle \Pi_1, s(x_1) z^{*\mathsf{T}} \rangle$. Finally, the transport optimization problem of Π_1 becomes

$$\underset{\Pi_1 \in \{0,1\}^{n \times n}}{\text{minimize}} \ \langle \Pi_1, C' \rangle \tag{5}$$
$$\text{subject to } \Pi_1 1_n = 1_n, \ \Pi_1^\mathsf{T} 1_n = 1_n,$$

where $C' = \xi C - s(x_1) z^{*\mathsf{T}}$. C'_{ij} is the cost of moving the ith region to the jth position, which consists of two components. The first component is the distance ξC_{ij}, which is defined as a distance from i to j. The second component is the saliency term, which discounts the transport cost with the saliency value of the ith region if the mask of jth position is non-zero. Briefly speaking, the larger the saliency value, the more the discount in the transport cost.

The optimization problem in Equation (5) can be solved exactly by using the Hungarian algorithm and its variants with time complexity of $O(n^3)$ (Munkres, 1957; Jonker & Volgenant, 1987). As we need to efficiently generate mixup examples per each mini-batch, this can be a computational bottleneck as n increases. Thus, we propose an approximate algorithm that can be parallelized on GPUs and efficiently computed in batches. The proposed algorithm can quickly decrease the objective $\langle \Pi, C' \rangle$ and converges to a local-minimum within $n(n-1)/2 + 1$ steps. Experimental results comparing the wall clock execution time and the relative error are in Supplementary B.

Algorithm 1 progressively alternates between row-wise and column-wise optimizations. The algorithm first minimizes

$\langle \Pi, C' \rangle$ only with the $\Pi 1_n = 1_n$ constraint. However, since the optimization is done without the column constraint, there can be multiple 1 values in a column of Π. In the following step, the column with multiple 1 values leaves only one 1 in the row with the smallest cost. We denote the result as Π_{win} in Algorithm 1. The corresponding cost entries for the rows that do not remain in Π_{win} are penalized with a large additive value, and the 1 values are moved to the other columns in the next iteration.

Our algorithm can also take advantage of intermediate Π_{win} as a solution, supported by the following two properties. We suppose that transport cost matrix C has zeros in diagonal entries and positive values in others. In addition, let $\Pi^{(t)}$ and $\Pi_{win}^{(t)}$ denote Π and Π_{win} at the end of tth step in Algorithm 1.

Proposition 3. *Suppose z^* has values in $\{0, 1\}$. Then for j s.t. $z_j^* = 1$, j^{th} column of $\Pi_{win}^{(t)}$ has exactly one 1.*

Proof. By the definition of $C^{(0)} = \xi C - s(x_0) z^{*\mathsf{T}}$, for j s.t. $z_j^* = 1$, j^{th} row of $C^{(0)}$ has a minimum at j^{th} entry. Thus, j^{th} column of $\Pi_{win}^{(0)}$ has exactly one 1 and others are 0. Suppose that, the claim is satisfied for $\Pi_{win}^{(t)}$ and $\Pi_{win}^{(t)}[i(j), j]$ is 1 for j s.t. $z_j^* = 1$. Then, by the definition of $\Pi_{win}^{(t)}$, $C_{win}^{(t)}[i(j), j]$ is the minimum of $i(j)^{th}$ row of $C^{(t)}$ and the row will not be updated in $C^{(t+1)}$. Thus, $i(j)^{th}$ row of $C^{(t+1)}$ has a minimum at j^{th} entry and j^{th} column of $\Pi_{win}^{(t+1)}$ has exactly one 1. By induction, the claim holds. $\square$

Proposition 4. *Under the assumption of Proposition 3, the partial objective $< \Pi_{win}^{(t)}, C' z^* >$ decreases as t increases.*

Proof. By Proposition 3, for j s.t. $z_j^* = 1$, j^{th} column of $\Pi_{win}^{(t)}$ has exactly one 1. Let $i(j; t)$ denote the corresponding row index with the entry 1. Then, it is enough to prove that $C'[i(j; t+1), j] \leq C'[i(j; t), j]$. However, in the last part of the proof of Proposition 3, we showed that $i(j; t)^{th}$ row of $C^{(t+1)}$ has a minimum at j^{th} entry, and thus $\Pi^{(t+1)}[i(j; t), j] = 1$. By Algorithm 1, index $i(j; t+1)$ satisfies $C^{(t+1)}[i(j; t+1), j] \leq C^{(t+1)}[i, j]$, $\forall i$ s.t. $\Pi^{(t+1)}[i, j] = 1$. Thus, $C^{(t+1)}[i(j; t+1), j] \leq C^{(t+1)}[i(j; t), j]$. Finally, $\Pi^{(t+1)}[i, j] = 1$ means that cost from i to j is not updated, i.e., $C^{(t+1)}[i, j] = C'[i, j]$. $\square$

Finally, we introduce the convergence property of Algorithm 1.

Proposition 5. *Algorithm 1 converges to a local-minimum with respect to the update rule at most $n(n-1)/2 + 1$ steps.*

Proof. See Supplementary A.2. $\square$

Algorithm 2 Stochastic Adversarial Puzzle Mix

Input: data x_0, x_1, attack ball ϵ, step τ, probability p
$x_{i,clean} = x_i$ for $i = 0, 1$
Sample $\nu_i \sim B(1, p)$ for $i = 0, 1$
for $i = 1, 2$ **do**
  if $\nu_i == 1$ **then**
    $\kappa_i \sim Uniform(-\epsilon, \epsilon)$
    $x_i \leftarrow x_i + \kappa_i$
  end if
end for
Calculate gradient $\nabla_x l(x_i)$ for $i = 0, 1$
Optimize z^* and Π_i^* in Equation (3)
Sample $\delta \sim Uniform(0, 1)$
for $i = 0, 1$ **do**
  if $\nu_i == 1$ **then**
    $\kappa_i \leftarrow \kappa_i + \tau \, sign(\nabla_x l(x_i))$
    $\kappa_i \leftarrow clip(\kappa_i, -\epsilon, \epsilon)$
    $x_i \leftarrow x_{i,clean} + \delta \, \kappa_i$
  end if
end for
Return: $(1 - z^*) \odot \Pi_0^{*\mathsf{T}} x_0 + z^* \odot \Pi_1^{*\mathsf{T}} x_1$

4.3. Adversarial Training

Since our mix-up strategy utilizes the gradients of the loss function with respect to the given inputs for saliency computation, we can incorporate adversarial training in our mix-up method without *any* additional computation cost.

For adversarial training on mixup data, we adapt the fast adversarial training method of Wong et al. (2020), which adds a uniform noise before creating an adversarial perturbation. As shown in Algorithm 2, we add the adversarial perturbation to the proper location of the mixed output, *i.e.*, adding an adversarial signal to the corresponding input and location specified by z. Note that the adversarial perturbation is added to each data probabilistically to prevent possible degradation in the generalization performance.

5. Implementation Details

First, to solve the discrete optimization problem with respect to the mask z, we use α-β swap algorithm from the pyGCO python wrapper[1]. Although the minimization is performed example-wise in CPUs, the α-β swap algorithm converges quickly, since we restrict the size of the graph with down-sampling. Note that, in our experiments, the computational bottleneck of the method is in the forward-backward passes of the neural network. In our experiments, we use label space $\mathcal{L} = \{0, \frac{1}{2}, 1\}$. In addition, we randomly sample the size of the graph, *i.e.*, size of mask z, from $\{2 \times 2, 4 \times 4, 8 \times 8, 16 \times 16\}$, and down-sample the given mini-batch

---

[1]https://github.com/Borda/pyGCO

---

for all experiments.

We normalize the down-sampled saliency map, which is used as the unary term, to sum up to 1. This allows us to use consistent hyperparameters *across all the* models and datasets. To measure the distance between the two adjacent data regions, we compute the mean of the absolute values of differences on the boundaries. For the mixing ratio λ, we randomly sample λ from $Beta(\alpha, \alpha)$ at each mini-batch. All of the computations in our algorithm except α-β swap are done in mini-batch and can be performed in parallel in GPUs. Note that for-loops in Algorithm 1 can be done in parallel by using the scatter function of PyTorch (Paszke et al., 2017).

Since we calculate the saliency information by back-propagating the gradient of loss function through the model, we can utilize this gradient information without any computational overhead. We regularize the gradient of the model with mixup data as $\nabla_\theta \ell(h(x_0, x_1), g(y_0, y_1); \theta) + \frac{1}{2}\lambda_{clean}(\nabla_\theta \ell(x_0, y_0; \theta) + \nabla_\theta \ell(x_1, y_1; \theta))$. This additional regularization helps us to improve generalization performance on Tiny-ImageNet and ImageNet.

6. Experiments

We train and evaluate classifiers on CIFAR-100 (Krizhevsky & Geoffrey, 2009), Tiny-ImageNet (Chrabaszcz et al., 2017), and ImageNet (Deng et al., 2009) datasets. We first study the generalization performance and adversarial robustness of our method (Section 6.1). Next, we show that our method can be used in conjunction with the existing augmentation method (AugMix) to simultaneously improve the corruption robustness and generalization performance (Section 6.2). Finally, we perform ablation studies for our method (Section 6.3).

6.1. Generalization Performance and Adversarial Robustness

6.1.1. CIFAR-100

We train two residual neural networks (He et al., 2015): WRN28-10 (Zagoruyko & Komodakis, 2016) and PreActResNet18 (He et al., 2016). We follow the training protocol of Verma et al. (2019), which trains WRN28-10 for 400 epochs and PreActResNet18 for 1200 epochs. Hyperparameter settings are available in Supplementary C.1. We reproduce the mixup baselines (Zhang et al., 2018; Verma et al., 2019; Yun et al., 2019; Hendrycks et al., 2020) and compare the baselines with our method under the same experimental settings described above. We denote the experiments as *Vanilla, Input, Manifold, CutMix, AugMix, Puzzle Mix* in the experiment tables.

Note however, our mixup method requires an additional

forward-backward evaluation of the network per mini-batch to calculate the saliency signal. For some practitioners, a fairer comparison would be to compare the performances at a fixed number of network evaluations (*i.e.* for power conservation). In order to compare our method in this condition, we also test our method trained for half the epochs and with twice the initial learning rate. We denote this experiment as *Puzzle Mix (half)* in the experiment tables.

In addition, we report experiments with the adversarial training described in Algorithm 2 with $p = 0.1$. We denote this experiment as *Puzzle Mix (adv)* in the tables. We assess the adversarial robustness against FGSM attack of 8/255 ℓ_∞ epsilon ball following the evaluation protocol of Zhang et al. (2018); Verma et al. (2019); Yun et al. (2019) for fair comparison. The results are summarized in Table 2 and Table 3, and adversarial robustness results against the PGD attack (Madry et al., 2017) are in Supplementary D.2.

We observe that Puzzle Mix outperforms other mixup baselines in generalization and adversarial robustness with WRN28-10 (Table 2) and PreActResNet18 (Table 3). With WRN28-10, *Puzzle Mix* improves Top-1 test error over the best performing baseline by 1.45%, and *Puzzle Mix (half)* outperforms by 1.17%. *Puzzle Mix (adv)* improves FGSM error rate over 8.41% than AugMix while achieving 1.39% lower Top-1 error rate than Manifold mixup, which had the best Top-1 score among baselines. We observe similar results with PreActResNet18. *Puzzle Mix (adv)* reduces the Top-1 error rate by 1.14% and the FGSM error rate by 12.98% over baselines.

Method	Top-1 Error(%)	Top-5 Error(%)	FGSM Error(%)
Vanilla	21.14	6.33	63.92
Input	18.27	4.98	56.60
Manifold	17.40	4.37	60.70
Manifold†	18.04	-	-
CutMix	17.50	4.69	79.34
AugMix	20.44	5.74	55.59
Puzzle Mix	**15.95**	3.92	63.71
Puzzle Mix (half)	16.23	**3.90**	66.74
Puzzle Mix (adv)	16.01	3.91	47.18
Puzzle Mix (half, adv)	16.39	3.94	**46.95**

Table 2. Top-1 / Top-5 / FGSM error rates on CIFAR-100 dataset of WRN28-10 trained with various mixup methods (400 epochs). † denotes the result reported in the original paper. Top-1 and Top-5 results are median test errors of models in the last 10 epochs.

6.1.2. TINY-IMAGENET

We train PreActResNet18 network on Tiny-ImageNet dataset, which contains 200 classes with 500 training images and 50 test images per class with 64×64 resolution (Chrabaszcz et al., 2017). Training settings are described in

Method	Top-1 Error(%)	Top-5 Error(%)	FGSM Error(%)
Vanilla	23.67	8.98	88.89
Input	23.16	7.58	70.09
Manifold	20.98	6.63	73.09
CutMix	23.20	8.09	86.38
AugMix	24.69	8.38	76.99
Puzzle Mix	**19.62**	5.85	79.47
Puzzle Mix (half)	20.09	**5.59**	75.72
Puzzle Mix (adv)	19.84	6.11	**57.11**
Puzzle Mix (half, adv)	19.96	6.20	59.33

Table 3. Top-1 / Top-5 / FGSM error rates on CIFAR-100 dataset of PreActResNet18 trained with various mixup methods.

Method	Top-1 Error(%)	Top-5 Error(%)	FGSM Error(%)
Vanilla	42.77	26.35	91.85
Input	43.41	26.98	88.68
Manifold	41.99	25.88	89.25
Manifold†	41.30	26.41	-
CutMix	43.33	24.48	87.19
AugMix	44.03	25.32	90.00
Puzzle Mix	**36.52**	**18.95**	92.52
Puzzle Mix (half)	37.64	19.37	92.57
Puzzle Mix (adv)	38.55	20.48	**82.07**
Puzzle Mix (half, adv)	38.14	19.70	83.91

Table 4. Top-1 / Top-5 / FGSM error rates on Tiny-ImageNet dataset for PreActResNet18 trained with various mixup methods.

Supplementary C.2.

As in the CIFAR-100 experiment, Puzzle Mix shows significant performance gains both on the generalization performance and the adversarial robustness compared to other mixup baselines (Table 4).

Puzzle Mix trained with the same number of epochs achieves 36.52% in Top-1 test error, 5.47% lower than the strongest baseline, and the model trained with same network evaluations (*half*) outperforms the best baseline by 4.35%. Puzzle Mix trained with stochastic adversarial method (*adv*) achieves best Top-1 and FGSM error rate ($\epsilon = 4/255$) compared to other mixup baselines providing 3.44% lower Top-1 error rate and 5.12% lower FGSM error rate.

6.1.3. IMAGENET

In ImageNet experiment, we use ResNet-50 to compare the performance. In order to train the model on ImageNet more efficiently, we utilize the cyclic learning rate, and use pre-resized images following the training protocol in Wong et al. (2020). Hyperparameter settings are in Supplementary C.3. Consistent with the previous experiments on CIFAR-100 and Tiny-ImageNet, Puzzle Mix shows the best performance

in both Top-1 / Top-5 error rate, achieving 0.43%, 0.24% improvement each, compared to the best baseline (Table 5).

Model	Top-1 Error(%)	Top-5 Error(%)
Vanilla	24.31	7.34
Input	22.99	6.48
Manifold	23.15	6.50
CutMix	22.92	6.55
AugMix	23.25	6.70
Puzzle Mix	**22.49**	**6.24**

Table 5. Top-1 / Top-5 error rates on ImageNet on ResNet-50.

6.2. Robustness Against Corruption

Hendrycks et al. (2020) proposed AugMix which performs Input mixup between clean and augmented images to improve robustness against corrupted datasets as well as the generalization performance. AugMix uses Jensen-Shannon divergence (JSD) between network outputs of a clean image and two AugMix images as a consistency loss. However, computing the JSD term requires triple the network evaluations compared to other mixup methods to train the network.

We found that simply using our mixup algorithm between two AugMix images, improves both the generalization and corruption robustness over the training strategy with the JSD objective. Note that our method requires only one additional (versus two) network evaluation per each mini-batch. We denote this experiment setting as *Puzzle Mix (aug)*.

We use CIFAR-100-C dataset (Hendrycks & Dietterich, 2019) to evaluate the corruption robustness. The dataset consists of 19 types of corruption, including noise, blur, weather, and digital corruption types. In Table 6, we report average test errors on CIFAR-100-C dataset as well as test errors on the clean CIFAR-100 test dataset. Table 6 demonstrates that our method using AugMix images improves both the generalization performance and the corruption accuracy by 3.95% and 2.31% each over AugMix baseline.

Method	Top-1 Error(%)	Corruption Error(%)
Vanilla	21.14	49.08
AugMix	20.45	32.22
Puzzle Mix	**15.95**	42.46
Puzzle Mix (aug)	16.50	**29.91**

Table 6. Top-1 / Corruption error rates on CIFAR-100 and CIFAR-100-C on WRN28-10.

6.3. Ablation Study

The generalization performance of Puzzle Mix stems from saliency-based multi-label masking and transport. We ver-

ified the effectiveness of these two factors in comparative experiments on CIFAR-100 with WRN28-10. Table 7 shows that Puzzle Mix with the binary label space (*binary*) has 1.44% higher Top-1 error rate than multi-label case, and Puzzle Mix without transport (*mask only*) has 0.43% higher Top-1 error rate than Puzzle Mix with transport.

Method	Top-1 Error(%)	Top-5 Error(%)
Vanilla	21.14	6.33
Puzzle Mix	**15.95**	3.92
Puzzle Mix (*binary*)	17.39	4.34
Puzzle Mix (*mask only*)	16.38	**3.78**

Table 7. Top-1 / Top-5 rates on CIFAR-100 dataset of WRN28-10 trained with our mixup methods.

We also verify the effects of different factors in stochastic adversarial training. In Algorithm 2, we add an adversarial perturbation to each data based on each Bernoulli sample ν_i and apply linear decay with δ sampled from the uniform distribution. From Table 8, we observe that using two independent random variables ν_0 and ν_1 (*adv*) has significant improvement in adversarial robustness over using one variable ($\nu_0 = \nu_1$). In the absence of linear decaying (*fgsm*), there is improvement in the FGSM error rate of 4.02%, but the Top-1 error increases by 0.41%. In all experiments, p is set to 0.1. We use FGSM attack of 8/255 ℓ_∞ epsilon-ball and 7-step PGD attack with a 2/255 step size. Additional experiments regarding the effect of p value in adversarial training are available in Supplementary D.1.

Method	Top-1 Error(%)	FGSM Error(%)	PGD Error(%)
Puzzle Mix (*adv*)	**16.01**	47.18	**90.18**
Puzzle Mix (*fgsm*)	16.42	**43.16**	91.19
Puzzle Mix ($\nu_0 = \nu_1$)	16.66	65.90	94.05

Table 8. Top-1 / FGSM / PGD error rates on CIFAR-100 dataset of WRN28-10 trained with our adversarial mixup methods.

7. Conclusion

We have presented Puzzle Mix, a mixup augmentation method for optimally leveraging the saliency information while respecting the underlying local statistics of the data. Puzzle Mix efficiently generates the mixup examples in a mini-batch stochastic gradient descent setting and outperforms other mixup baseline methods both in the generalization performance and the robustness against adversarial perturbations and data corruption by a large margin on CIFAR-100, Tiny-ImageNet, and ImageNet datasets.

Acknowledgements

References

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. A theory of learning from different domains. *Machine learning*, 79(1-2):151–175, 2010.

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

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

Boykov, Y., Veksler, O., and Zabih, R. Fast approximate energy minimization via graph cuts. *IEEE Transactions on pattern analysis and machine intelligence*, 23(11): 1222–1239, 2001.

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

Courty, N., Flamary, R., Tuia, D., and Rakotomamonjy, A. Optimal transport for domain adaptation. *IEEE transactions on pattern analysis and machine intelligence*, 2016.

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

DeVries, T. and Taylor, G. W. Improved regularization of convolutional neural networks with cutout. *arXiv preprint arXiv:1708.04552*, 2017.

Erkan, G. and Radev, D. R. Lexrank: Graph-based lexical centrality as salience in text summarization. *Journal of artificial intelligence research*, 22:457–479, 2004.

Guo, H., Mao, Y., and Zhang, R. Mixup as locally linear out-of-manifold regularization. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019.

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

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

Hendrycks, D. and Dietterich, T. Benchmarking neural network robustness to common corruptions and perturbations. *ICLR*, 2019.

Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. AugMix: A simple data processing method to improve robustness and uncertainty. *ICLR*, 2020.

Huang, J. and Mumford, D. Statistics of natural images and models. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pp. 541–547. IEEE, 1999.

Jonker, R. and Volgenant, A. A shortest augmenting path algorithm for dense and sparse linear assignment. *Computing*, 38(4):325–340, 1987.

Kalinli, O. and Narayanan, S. S. 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.

Kolmogorov, V. and Zabih, R. What energy functions can be minimizedvia graph cuts? *IEEE Transactions on Pattern Analysis & Machine Intelligence*, 26(2):147–159, 2004.

Krizhevsky, A. and Geoffrey, H. Learning multiple layers of features from tiny images. *University of Toronto*, 2009.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. *NeurlPS*, 2012.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*, 2017.

Munkres, J. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, 1957.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. *NeurIPS*, 2017.

Rabin, J., Ferradans, S., and Papadakis, N. Adaptive color transfer with relaxed optimal transport. In *2014 IEEE International Conference on Image Processing (ICIP)*, 2014.

Ren, Z., Gao, S., Chia, L.-T., and Tsang, I. W.-H. Region-based saliency detection and its application in object recognition. *IEEE Transactions on Circuits and Systems for Video Technology*, 24(5):769–779, 2013.

Schmidt, M. and Alahari, K. Generalized fast approximate energy minimization via graph cuts: Alpha-expansion beta-shrink moves. *arXiv preprint arXiv:1108.5710*, 2011.

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

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

Smith, C. S. *Modes of discourse: The local structure of texts*, volume 103. Cambridge University Press, 2003.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

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

Villani, C. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

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

Wei, Y., Feng, J., Liang, X., Cheng, M.-M., Zhao, Y., and Yan, S. Object region mining with adversarial erasing: A simple classification to semantic segmentation approach. *CVPR*, 2017.

Wong, E., Rice, L., and Kolter, J. Z. Fast is better than free: Revisiting adversarial training. *ICLR*, 2020.

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

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

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

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

Zhang, Y., Pezeshki, M., Brakel, P., Zhang, S., Bengio, C. L. Y., and Courville, A. Towards end-to-end speech recognition with deep convolutional neural networks. *arXiv preprint arXiv:1701.02720*, 2017.

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

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