

# Mix up相关文章及启发

郭丹丹

2022/12/30

2021 ICLR

## SALIENCYMIX: A SALIENCY GUIDED DATA AUGMENTATION STRATEGY FOR BETTER REGULARIZATION

A. F. M. Shahab Uddin \*  
uddin@khu.ac.kr

Mst. Sirazam Monira \*  
monira@khu.ac.kr

Wheemyung Shin \*  
wheemi@khu.ac.kr

TaeChoong Chung \*†  
tcchung@khu.ac.kr

Sung-Ho Bae \*†  
shbae@khu.ac.kr

2022 NeurIPS

## C-Mixup: Improving Generalization in Regression

Huaxiu Yao<sup>1\*</sup>, Yiping Wang<sup>2\*</sup>, Linjun Zhang<sup>3</sup>, James Zou<sup>1</sup>, **Chelsea Finn<sup>1</sup>**

<sup>1</sup>Stanford University, <sup>2</sup>Zhejiang University, <sup>3</sup>Rutgers University

<sup>1</sup>{huaxiu,cbfinn}@cs.stanford.edu, jamesz@stanford.edu

<sup>2</sup>yipingwang6161@gmail.com, <sup>3</sup>linjun.zhang@rutgers.edu

2022 NeurIPS

## UMIX: Improving Importance Weighting for Subpopulation Shift via Uncertainty-Aware Mixup

Zongbo Han<sup>1\*†</sup>, Zhipeng Liang<sup>2\*§</sup>, Fan Yang<sup>3\*</sup>, Liu Liu<sup>3</sup>, Lanqing Li<sup>3</sup>, Yatao Bian<sup>3</sup>,  
Peilin Zhao<sup>3</sup>, Bingzhe Wu<sup>3†</sup>, Changqing Zhang<sup>1†</sup>, Jianhua Yao<sup>3†</sup>

<sup>1</sup>College of Intelligence and Computing, Tianjin University,

<sup>2</sup>Hong Kong University of Science and Technology, <sup>3</sup>Tencent AI Lab

2022 NeurIPS

## RecursiveMix: Mixed Learning with History





Lingfeng Yang<sup>1#</sup>, Xiang Li<sup>1#</sup>, Borui Zhao<sup>2</sup>, Renjie Song<sup>2</sup>, Jian Yang<sup>1\*</sup>

<sup>1</sup>Nanjing University of Science and Technology, <sup>2</sup>Megvii Technology

# Data augmentation

- **Manually designed data augmentations:**
  - For hand written character recognition: Lecun et al. performed several affine transformations such as translation, scaling, shearing, etc.
  - Bengio et al. (2011) applied more diverse transformation such as Gaussian noise, salt and pepper noise, Gaussian smoothing, motion blur, local elastic deformation, and various occlusions to the images.
  - Krizhevsky et al. (2012) applied random image patch cropping, horizontal flipping and random color intensity changing based on principal component analysis (PCA).
  - In Deep Image (Wu et al., 2015), color casting, vignetting, and lens distortion are applied besides flipping and cropping to improve the robustness of a very deep network.

# Data augmentation

		Zhang et al. (2017)	Devries & Taylor (2017)	Yun et al. (2019)
	ResNet-50	Mixup [48]	Cutout [3]	CutMix
Image				
Label	Dog 1.0	Dog 0.5 Cat 0.5	Dog 1.0	Dog 0.6 Cat 0.4

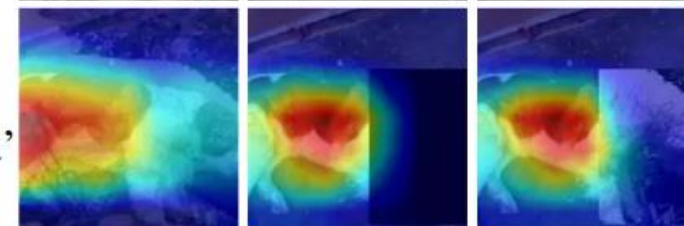
Original  
Samples



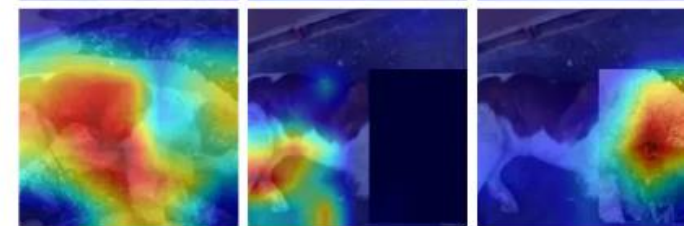
Input  
Image



CAM for  
'St. Bernard'



CAM for  
'Poodle'



Mixup

Cutout

CutMix

Kim et al. (2020) proposed **PuzzleMix** that jointly optimize two objectives i.e., selecting an optimal mask and an optimal mixing plan. The mask tries to reveal most salient data of two images and the **optimal transport plan** aims to maximize the saliency of the revealed portion of the data.

2020 ICML 《Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup》

# SaliencyMix: a saliency guided data augmentation strategy for better regularization

$$\tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j,$$



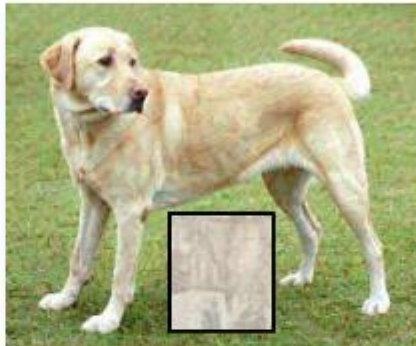

Target Image	Source Image	Augmented Image	
			
Mixed label for randomly mixed images		Dog - 80% & Cat 20% ?	Dog - 80% & Cat 20% ?

Figure 1: Problem of randomly selecting image patch and mixing labels according to it. When the selected source patch does not represent the source object, the interpolated label misleads the model to learn unexpected feature representation.

Because CNNs are highly sensitive to textures (Geirhos et al., 2019) and since the interpolated label indicates the selected background patch as the source object, it may encourage the classifier to learn the background as the representative feature for the source object class.

# SALIENCY DETECTION

- Saliency detection aims to simulate the natural attention mechanism of human visual system (HVS) and can be classified into two main categories.
- The first one is a bottom-up approach that focuses on exploring low-level vision features.
- The second one is a top-down approach which is task-driven and utilizes supervised learning with labels.
- In this study, we require a saliency model to focus on the important object/region in a given scene without knowing their labels. As a result, we rely on bottom-up approach which are unsupervised, scale-invariant and more robust for unseen data.



# SELECTION OF THE SOURCE PATCH

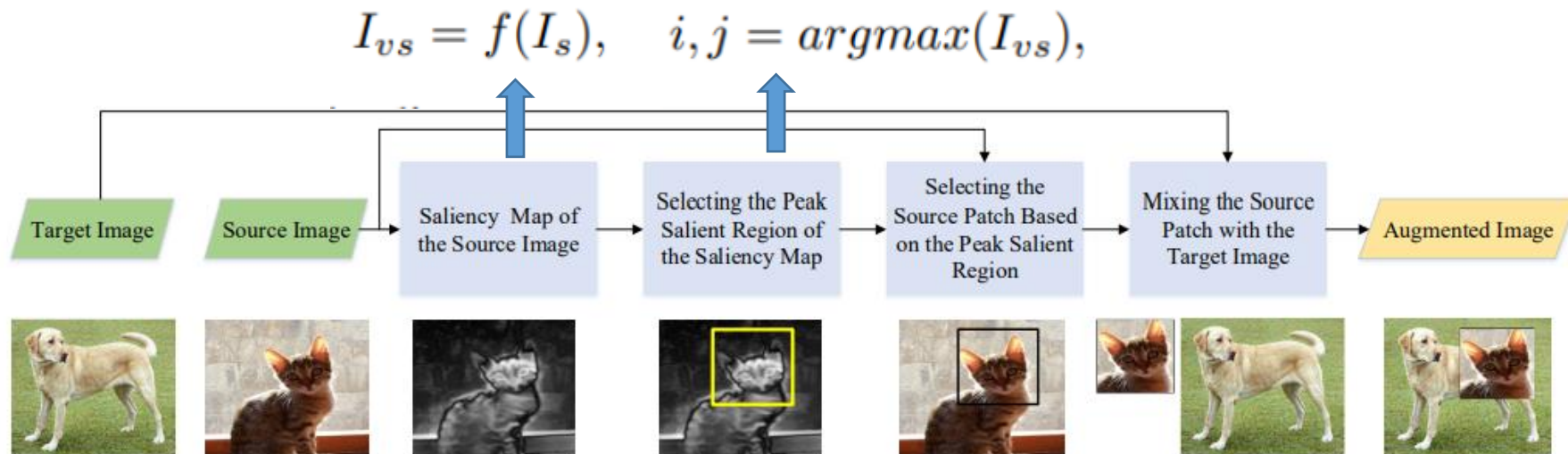


Figure 2: The proposed SaliencyMix data augmentation. We first extract the saliency map of the source image that highlights the regions of interest. Then we select a patch around the peak salient pixel location and mix it with the target image.

Then we select a patch, either by centering on the  $I_{vs}^{i,j} - th$  pixel if possible, or keeping the  $I_{vs}^{i,j} - th$  pixel on the selected patch. It ensures that the patch is selected from the object region, not from the background. The size of the patch is determined based on a combination ratio  $\lambda$  which is sampled from the uniform distribution  $(0, 1)$  to decide the percentage of an image to be cropped.

# MIXING THE PATCHES AND LABELS

Let  $I_t \in \mathbb{R}^{W \times H \times C}$  is another randomly selected training (target) image with label  $y_t$ , to where the source patch will be mixed. SaliencyMix partially mixes  $I_t$  and  $I_s$  to produce a new training sample  $I_a$ , the augmented image, with label  $y_a$ . The mixing of two images can be defined as

$$I_a = M \odot I_s + M' \odot I_t, \quad (3)$$

where  $I_a$  denotes the augmented image,  $M \in \{0, 1\}^{W \times H}$  represents a binary mask,  $M'$  is the complement of  $M$  and  $\odot$  represents element-wise multiplication. First, the source patch location is defined by using the peak salient information and the value of  $\lambda$  and then the corresponding location of the mask  $M$  is set to 1 and others to 0. The element-wise multiplication of  $M$  with the source image results with an image that removes everything except the region decided to keep. In contrast,  $M'$  performs in an opposite way of  $M$  i.e., the element-wise multiplication of  $M'$  with the target image keeps all the regions except the selected patch. Finally, the addition of those two creates a new training sample that contains the target image with the selected source patch in it (See Figure 2). Besides mixing the images we also mix their labels based on the size of the mixed patches as

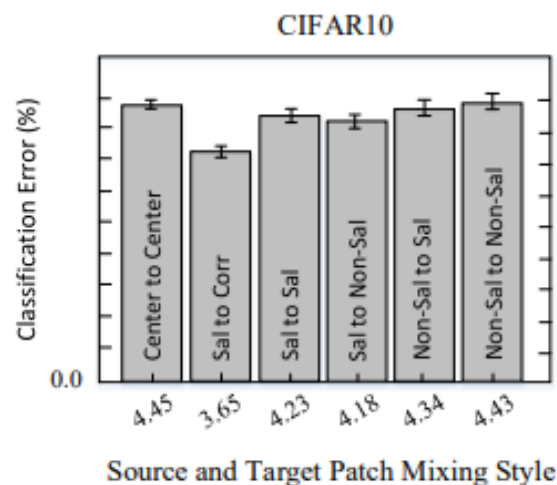
$$y_a = \lambda y_t + (1 - \lambda) y_s, \quad (4)$$

where  $y_a$  denotes the label for the augmented sample and  $\lambda$  is the combination ratio. Other ways of mixing are investigated in Section 3.4.

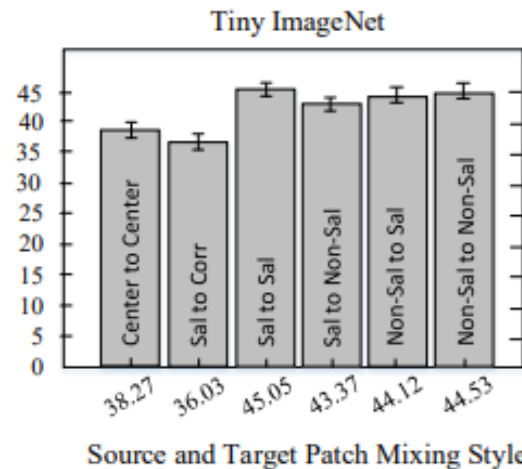


# DIFFERENT WAYS OF SELECTING AND MIXING THE SOURCE PATCH

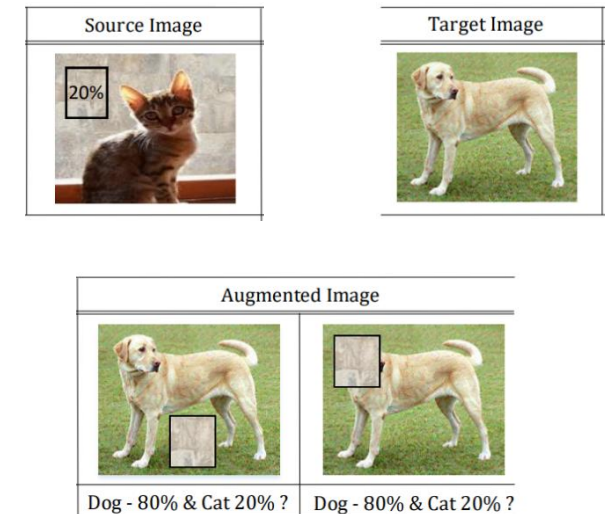
CIFAR-10 and Tiny-ImageNet datasets, respectively. We consider five possible schemes: (i) *Salient to Corresponding*, that selects the source patch from the most salient region and mix it to the corresponding location of the target image; (ii) *Salient to Salient*, that selects the source patch from the most salient region and mix it to the salient region of the target image; (iii) *Salient to Non-Salient*, that selects the source patch from the most salient region but mix it to the non-salient region of the target image; (iv) *Non-Salient to Salient*, that selects the source patch from the non-salient region of the source image but mix it to the salient region of the target image; and (v) *Non-Salient to Non-Salient*, that selects the source patch from the non-salient region of the source image and also mix it to the non-salient region of the target image. To find out the non-salient region, we use the least important pixel of an image.



(c)



(d)



# Brain storming

- For fewshot classification or imbalanced classification task
- 以不平衡为例，对劣势样本 $x_t$ 及源样本 $x_s$ ，可选择源样本的背景（non-salient）与 $x_t$ 混合，即non-salient to corresponding。仍旧对混合样本赋予 $x_s$ 的类别
- 即non-salient to non-salient。仍旧对混合样本赋予 $x_s$ 的类别

# Background

Deep learning practitioners commonly face the challenge of overfitting. To improve generalization, prior works have proposed a number of techniques, including data augmentation [3, 10, 12, 81, 82]

and explicit regularization [15, 38, 60]. Representatively, mixup [82, 83] densifies the data distribution and implicitly regularizes the model.

**Mixup.** The mixup algorithm samples a pair of instances  $(x_i, y_i)$  and  $(x_j, y_j)$ , sampled uniformly at random from the training dataset, and generates new examples by performing linear interpolation on the input features and corresponding labels as:

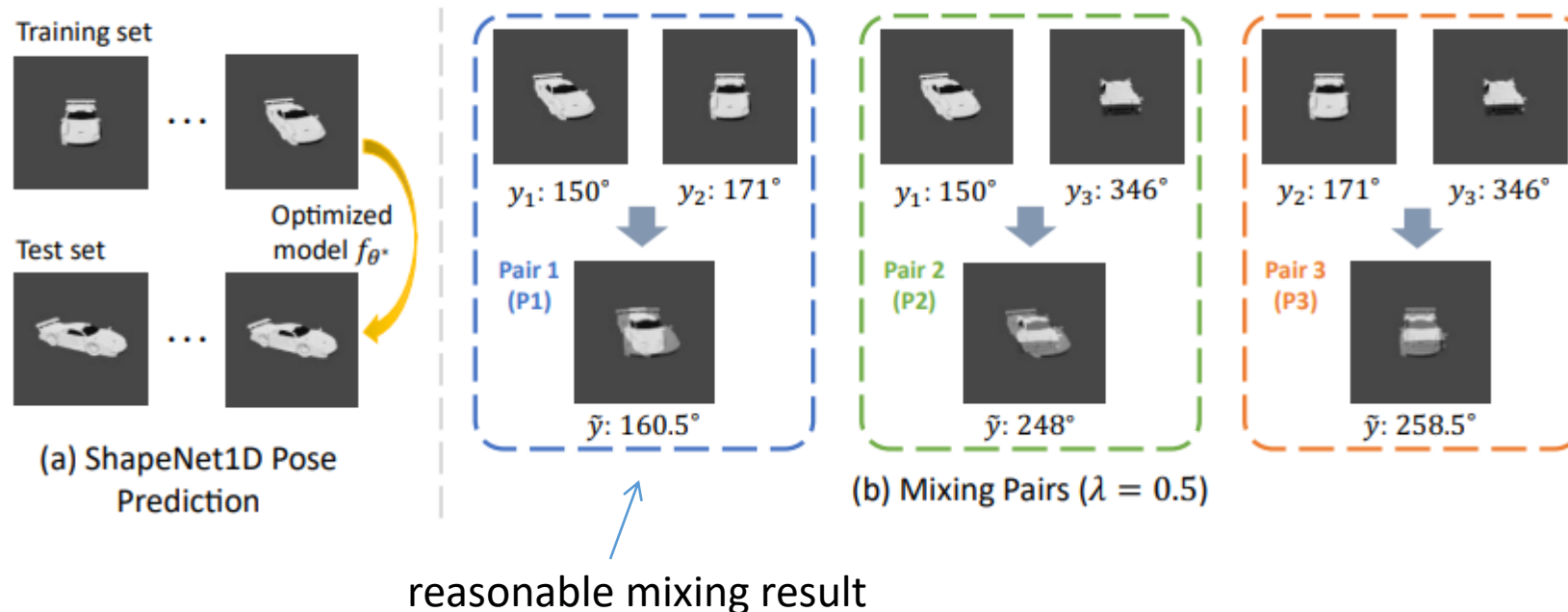
$$\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j, \quad (2)$$

where the interpolation ratio  $\lambda \in [0, 1]$  is drawn from a Beta distribution, i.e.,  $\lambda \sim \text{Beta}(\alpha, \alpha)$ . The interpolated examples are then used to optimize the model as follows:

$$\theta^* \leftarrow \arg \min_{\theta \in \Theta} \mathbb{E}_{(x_i, y_i), (x_j, y_j) \sim P^{tr}} [\ell(f_{\theta}(\tilde{x}), \tilde{y})]. \quad (3)$$

# C-Mixup: Improving Generalization in Regression

- In contrast to classification, which formalizes the label as a one-hot vector, the goal of regression is to predict a continuous label from each input.
- Directly applying mixup to input features and labels in regression tasks may yield arbitrarily incorrect labels.



# Mixup for Regression (C-Mixup)

For continuous labels, the example in Figure 1(b) illustrates that applying vanilla mixup to the entire distribution is likely to produce arbitrary labels. To resolve this issue, C-Mixup proposes to sample closer pairs of examples with higher probability. Specifically, given an example  $(x_i, y_i)$ , C-Mixup introduces a symmetric Gaussian kernel to calculate the sampling probability  $P((x_j, y_j)|(x_i, y_i))$  for another  $(x_j, y_j)$  example to be mixed as follows:

$$P((x_j, y_j)|(x_i, y_i)) \propto \exp\left(-\frac{d(i, j)}{2\sigma^2}\right) \quad (6)$$

where  $d(i, j)$  represents the distance between the examples  $(x_i, y_i)$  and  $(x_j, y_j)$ , and  $\sigma$  describes the bandwidth. For the example  $(x_i, y_i)$ , the set  $\{P((x_j, y_j)|(x_i, y_i))|\forall j\}$  is then normalized to a probability mass function that sums to one.



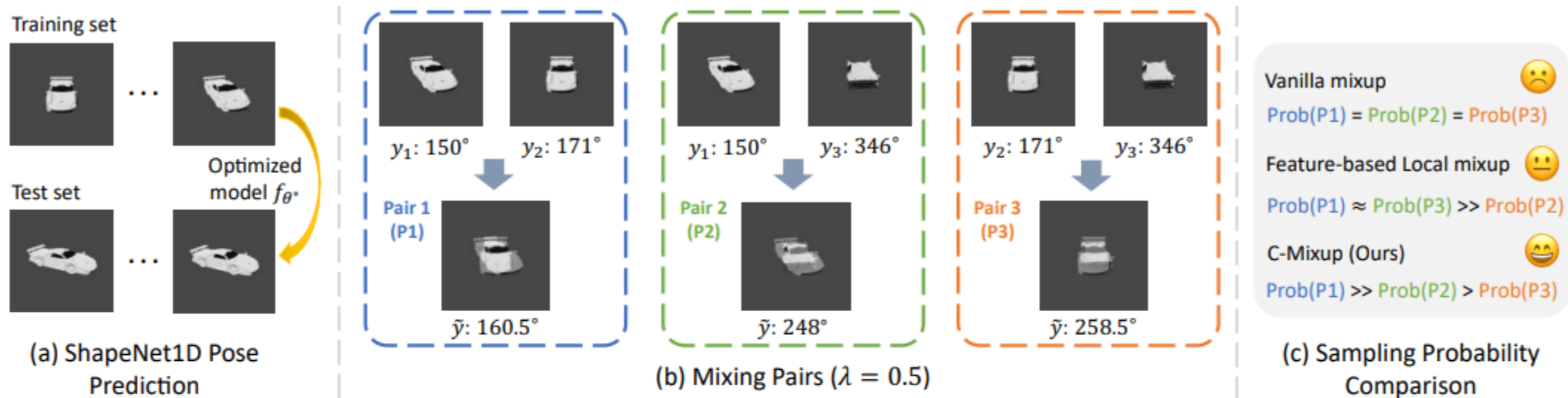


Figure 1: Illustration of C-Mixup on ShapeNet1D pose prediction.  $\lambda$  represents the interpolation ratio. (a) ShapeNet1D pose prediction task, aiming to predict the current orientation of the object relative to its canonical orientation. (b) Three mixing pairs are randomly picked, where the interpolated images are visualized and  $\tilde{y}$  represents interpolated labels. (c) Illustration of a rough comparison of sampling probabilities among three mixing pairs in (b). The Euclidean distance measures input feature distance and the corresponding results between examples in pairs 1, 2, 3 are  $1.51 \times 10^5$ ,  $1.82 \times 10^5$ ,  $1.50 \times 10^5$ , respectively. Hence, pairs 1 and 3 have similar results, leading to similar sampling probabilities. C-Mixup is able to assign higher sampling probability for more reasonable mixing pairs.

One natural way to compute the distance is using the input feature  $x$ , i.e.,  $d(i, j) = d(x_i, x_j)$ . However, when dealing with the high-dimensional data such as images or videos, we lack good distance metrics to capture structured feature information and the distances can be easily influenced by feature noise. Additionally, computing feature distances for high-dimensional data is time-consuming. Instead, C-Mixup leverages the labels with  $d(i, j) = d(y_i, y_j) = \|y_i - y_j\|_2^2$ , where  $y_i$  and  $y_j$  are vectors with continuous values. The dimension of label is typically much smaller

---

### Algorithm 1 Training with C-Mixup

---

**Require:** Learning rates  $\eta$ ; Shape parameter  $\alpha$

**Require:** Training data  $\mathcal{D} := \{(x_i, y_i)\}_{i=1}^N$

1: Randomly initialize model parameters  $\theta$

2: Calculate pairwise distance matrix  $P$  via Eqn. (6)  $P((x_j, y_j)|(x_i, y_i)) \propto \exp\left(-\frac{d(i, j)}{2\sigma^2}\right)$

3: **while** not converge **do**

4:   Sample a batch of examples  $\mathcal{B} \sim \mathcal{D}$

5:   **for** each example  $(x_i, y_i) \in \mathcal{B}$  **do**

6:     Sample  $(x_j, y_j)$  from  $P(\cdot | (x_i, y_i))$  and  $\lambda$  from Beta( $\alpha, \alpha$ )

7:     Interpolate  $(x_i, y_i), (x_j, y_j)$  to get  $(\tilde{x}, \tilde{y})$  according to Eqn. (2)

8:     Use interpolated examples to update the model via Eqn. (3)

$$\tilde{x} = \lambda \cdot x_i + (1 - \lambda) \cdot x_j, \quad \tilde{y} = \lambda \cdot y_i + (1 - \lambda) \cdot y_j,$$

$$\theta^* \leftarrow \arg \min_{\theta \in \Theta} \mathbb{E}_{(x_i, y_i), (x_j, y_j) \sim P^{tr}} [\ell(f_\theta(\tilde{x}), \tilde{y})].$$


---

# Theoretically explain how C-Mixup benefits

- In-distribution generalization: the features are observed with noise, and the response depends on a small fraction of the features in a monotonic way.
- Task generalization

each task follows the model discussed in the last section. Concretely, we apply C-Mixup to MetaMix [74]. For each query example, the support example with a more similar label will have a higher probability of being mixed. The algorithm of C-Mixup on MetaMix is summarized in Appendix A.2.
- Out-of-distribution robustness: spurious correlation between angle (label) and other attributes/features.

---

**Algorithm 1** Meta-Training Process of MetaMix with mixReg

---

**Require:** Outer-loop learning rate  $\eta$ ; Inner-loop learning rate  $\xi$  (we change  $\alpha$  in Eqn. (4) of the main paper to  $\xi$  to avoid notation conflict); Shape parameter  $\alpha$ ; Task distribution  $p(\mathcal{T})$

- 1: Randomly initialize model parameters  $\theta$
  - 2: **while** not converge **do**
  - 3:   Sample a batch of tasks  $\{\mathcal{T}_i\}_{i=1}^{|M|}$  with the corresponding dataset  $\mathcal{D}_m$
  - 4:   **for** all  $\mathcal{T}_m$  **do**
  - 5:     Sample a support set  $\mathcal{D}_m^s$  and a query set  $\mathcal{D}_m^q$  from  $\mathcal{D}_m$
  - 6:     Calculate pairwise distance matrix  $P$  between query set and support set via Eqn. (6).  $P((x_j, y_j)|(x_i, y_i)) \propto \exp\left(-\frac{d(i, j)}{2\sigma^2}\right)$
  - 7:     Calculate the task-specific parameter  $\phi_m$  via the inner-loop gradient descent, i.e.,  $\phi_m = \theta - \xi \nabla_{\theta} \mathcal{L}(f_{\theta}; \mathcal{D}_m^s)$
  - 8:     **for** each query example  $(x_{m,i}^q, y_{m,i}^q)$  **do**
  - 9:       Sample MetaMix parameter  $\lambda \sim \text{Beta}(\alpha, \alpha)$
  - 10:       Sample support set example  $(x_{m,j}^s, y_{m,j}^s)$  according to the probability  $P(\cdot | (x_{m,i}^q, y_{m,i}^q))$
  - 11:       Linearly interpolate  $(x_{m,i}^q, y_{m,i}^q)$  and  $(x_{m,j}^s, y_{m,j}^s)$  to get  $(\tilde{x}_{m,i}^q, \tilde{y}_{m,i}^q)$
  - 12:       Replace  $(x_{m,i}^q, y_{m,i}^q)$  with  $(\tilde{x}_{m,i}^q, \tilde{y}_{m,i}^q)$
  - 13:   Use interpolated examples to update the model via  $\theta \leftarrow \theta - \eta \frac{1}{|M|} \sum_{i=1}^{|M|} \mathcal{L}(f_{\phi_m}; \tilde{\mathcal{D}}_m^q)$
-



#### 40 A.4 Discussion between mixReg and mixup

41 In this paper, we regard mixReg is an complementary approach to mixup and its most representative  
42 variants (e.g., Manifold Mixup [16], CutMix [20]). Here, we use vanilla mixup as an exemplar to  
43 show the difference. According to our description of mixup in Section 2 of the main paper, the entire  
44 mixup process includes three stages:

- 45 • Stage I: sample two instances  $(x_i, y_i), (x_j, y_j)$  from the training set.
- 46 • Stage II: sample the interpolation factor  $\lambda$  from the Beta distribution  $\text{Beta}(\alpha, \alpha)$ .
- 47 • Stage III: mixing the sampled instances with interpolation factor  $\lambda$  according to the following  
48 mixing formulation:

$$x_{mix} = \lambda x_i + (1 - \lambda)x_j, y_{mix} = \lambda y_i + (1 - \lambda)y_j, \lambda \sim \text{Beta}(\alpha, \alpha).$$

49 In the original mixup, the interpolation factor  $\lambda$  sampled in the stage II controls how to mix these  
50 two instances. mixReg instead manipulates stage I and pairs with closer labels are more likely to be  
51 sampled.

52 In addition to the discussion of the complementarity of mixReg, the original mixup paper further  
53 shows that randomly interpolating examples from the same label performs worse than completely  
54 random mixing examples in classification. Compared to classification, randomly mixing examples in  
55 regression may be easier to generate semantically wrong labels. Intuitively, linearly mixing one-hot  
56 labels in classification is easy to generate semantically meaningful artificial labels, where the mixed  
57 label represents the probabilities of mixed examples to some extent. While in regression, the mixed  
58 labels may be semantically meaningless (e.g., pairs 2 and 3 in Figure 1) and more significantly  
59 affect the performance. By mixing examples with closer labels, mixReg mitigates the influence of  
60 semantically wrong labels and improves the in-distribution and task generalization in regression.  
61 Additionally, mixReg further shows its superiority in improving out-of-distribution robustness in  
62 regression, which is not discussed in the original mixup paper.



# Brain storming

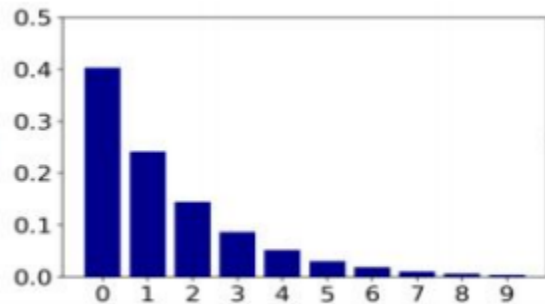
- For fewshot classification or imbalanced classification task
- cutmix时不再随意选择样本对，而是对给定样本 $x_i$ ，依据相似性，选择 $x_j$ ，参考改文章和2022 TPAMI 《PatchMix Augmentation to Identify Causal Features in Few-shot Learning 》
- 以不平衡为例，对劣势样本 $x_i$ 及剩余 $N$ 个样本，构造 $N$ 维simplex的采样概率，之后mixup时根据采样概率选择样本 $x_j$ ，构造增广样本
- 采样概率可用类间相似度及样本间相似度共同得到，简单版本：只依赖类间相似度，对 $x_i$ 所在类与其他类的相关性排序， $c_1 c_2 c_3 c_4 \dots$ ，之后采样 $x_j$ 时优先选择类别 $c_1$ ，之后再采类内样本。

# UMIX: Improving Importance Weighting for **Subpopulation Shift** via Uncertainty-Aware Mixup

subpopulation shift : the training and test distributions consist of the same subpopulation groups but differ in subpopulation frequencies [6, 8]. Many practical research problems (e.g., fairness of machine learning and class imbalance) can all be considered as a special case of subpopulation shift [21, 28, 32].

## imbalanced classification

train



test: balanced

## fairness of machine learning

train:



test: label  $y$  is independent of attribute



# Existing solutions

- Importance weighting (IW) is a classical yet effective technique by imposing static or adaptive weights on each sample when building weighted empirical loss. Therefore each subpopulation group contributes comparably to the final training objective.
- Early works propose to reweight the sample inverse proportionally to the subpopulation frequencies (i.e., static weights) , such as class-imbalanced.
- Alternatively, a more flexible way is to reweight individual samples adaptively according to training dynamics [35, 41, 48, 49, 63, 66, 72, 74].
- Distributional robust optimization (DRO) is one of the most representative methods in this line, which minimizes the loss over the worst-case distribution in a neighborhood of the empirical training distribution.

# Method

In this section, we introduce technical details of UMIX. The key idea of UMIX is to exploit uncertainty information to upweight mixed samples, and thus can encourage the model to perform uniformly well on all subpopulations. We first introduce the basic procedure of UMIX and then present how to provide high-quality uncertainty estimations which is the fundamental block of UMIX.

The necessary background and notations are provided here. Let the input and label space be  $\mathcal{X}$  and  $\mathcal{Y}$  respectively. Given training dataset  $\mathcal{D}$  with  $N$  training samples  $\{(x_i, y_i)\}_{i=1}^N$  i.i.d. sampled from a probability distribution  $P$ . We consider the setting that the training distribution  $P$  is a mixture of  $G$  predefined subpopulations, i.e.,  $P = \sum_{g=1}^G k_g P_g$ , where  $k_g$  and  $P_g$  denote the  $g$ -th subpopulation's proportion and distribution respectively. Our goal is to obtain a model  $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$  parameterized by  $\theta \in \Theta$  that performs well on all subpopulations.

Previous works on improving subpopulation shift robustness investigate several different settings, i.e., group-aware and group-oblivious [42, 59, 72]. Most of the previous works have assumed that the group label is available during training [59, 70]. This is called the group-aware setting. However, due to some reasons, we may not have training group labels. For example, in many real applications, it's hard to extract group label information. Meanwhile, the group label information may not be available due to privacy concerns. This paper studies the group-oblivious setting, which cannot obtain group information for each example at training time. This requires the model to identify underperforming samples and then pay more attention to them during training.

y: blond hair  
a: male



# Vanilla mixup

overfitting. Specifically, vanilla mixup [75, 76] constructs virtual training examples (i.e., mixed samples) by performing linear interpolations between data/features and corresponding labels as:

$$\tilde{x}_{i,j} = \lambda x_i + (1 - \lambda)x_j, \quad \tilde{y}_{i,j} = \lambda y_i + (1 - \lambda)y_j, \quad (1)$$

where  $(x_i, y_i), (x_j, y_j)$  are two samples drawn at random from empirical training distribution and  $\lambda \in [0, 1]$  is usually sampled from a beta distribution. Then vanilla mixup optimizes the following loss function:

$$\mathbb{E}_{\{(x_i, y_i), (x_j, y_j)\}} [\ell(\theta, \tilde{x}_{i,j}, \tilde{y}_{i,j})]. \quad (2)$$

When the cross entropy loss is employed, Eq. 2 can be rewritten as:

$$\mathbb{E}_{\{(x_i, y_i), (x_j, y_j)\}} [\lambda \ell(\theta, \tilde{x}_{i,j}, y_i) + (1 - \lambda) \ell(\theta, \tilde{x}_{i,j}, y_j)]. \quad (3)$$

Eq. 3 can be seen as a linear combination (mixup) of  $\ell(\theta, \tilde{x}_{i,j}, y_i)$  and  $\ell(\theta, \tilde{x}_{i,j}, y_j)$ . Unfortunately,

Since vanilla mixup doesn't consider the subpopulations with poor performance, it has been shown experimentally to be non-robust against subpopulation shift.



# Importance-weighted mixup

In contrast to previous IW methods, the importance weights of UMIX are used on the mixed samples. To do this, we first estimate the uncertainty of each sample and then use this quantity to construct the importance weight (i.e., the higher the uncertainty, the higher the weight, and vice versa). For the  $i$ -th sample  $x_i$ , we denote its importance weight as  $w_i$ . Once we obtain the importance weight, we can perform weighted linear combination of  $\ell(\theta, \tilde{x}_{i,j}, y_i)$  and  $\ell(\theta, \tilde{x}_{i,j}, y_j)$  by:

$$\mathbb{E}_{\{(x_i, y_i), (x_j, y_j)\}} [w_i \lambda \ell(\theta, \tilde{x}_{i,j}, y_i) + w_j (1 - \lambda) \ell(\theta, \tilde{x}_{i,j}, y_j)], \quad (4)$$

where  $w_i$  and  $w_j$  denote the importance weight of the  $i$ -th and  $j$ -th samples respectively. In practice, to balance the UMIX and normal training, we set a hyperparameter  $\sigma$  that denotes the probability to apply UMIX. The whole training pseudocode for UMIX is shown in Algorithm 1.

---

**Algorithm 1:** The training pseudocode of UMIX.

---

**Input:** Training dataset  $\mathcal{D}$  and the corresponding importance weights  $\mathbf{w} = [w_1, \dots, w_N]$ , hyperparameter  $\sigma$  to control the probability of doing UMIX, and parameter  $\alpha$  of the beta distribution;

```
1 for each iteration do
2   Obtain training samples  $(x_i, y_i), (x_j, y_j)$  and the corresponding weight  $w_i, w_j$ ;
3   Sample  $p \sim \text{Uniform}(0,1)$ ;
4   if  $p < \sigma$  then Sample  $\lambda \sim \text{Beta}(\alpha, \alpha)$ ; else  $\lambda = 0$ ;
5   Obtain the mixed input  $\tilde{x}_{i,j}$  where  $\tilde{x}_{i,j} = \lambda x_i + (1 - \lambda) x_j$ ;
6   Obtain the loss of the model with  $w_i \lambda \ell(\theta, \tilde{x}_{i,j}, y_i) + w_j (1 - \lambda) \ell(\theta, \tilde{x}_{i,j}, y_j)$ ;
7   Update model parameters  $\theta$  to minimize loss with an optimization algorithm.
```

---

# Uncertainty-aware importance weights

Given a well trained neural classifier  $f_\theta : \mathcal{X} \rightarrow \mathcal{Y}$  that could produce the predicted class  $\hat{f}_\theta(x)$ , a simple way to obtain the uncertainty of a sample is whether the sample is correctly classified.

However, as pointed out in previous work [36], a single model cannot accurately characterize the sampling uncertainty. Therefore, we propose to obtain the uncertainty through Bayesian sampling from the model posterior distribution  $p(\theta; \mathcal{D})$ . Specifically, given a sample  $(x_i, y_i)$ , we define the training uncertainty as:

$$u_i = \int \kappa(y_i, \hat{f}_\theta(x_i)) p(\theta; \mathcal{D}) d\theta, \text{ where } \kappa(y_i, \hat{f}_\theta(x_i)) = \begin{cases} 0, & \text{if } y_i = \hat{f}_\theta(x_i) \\ 1, & \text{if } y_i \neq \hat{f}_\theta(x_i) \end{cases}. \quad (5)$$

Then, we can obtain an approximation of Eq. 5 with  $T$  Monte Carlo samples as  $u_i \approx \frac{1}{T} \sum_{t=1}^T \kappa(y_i, \hat{f}_{\theta_t}(x_i))$ , where  $\theta_t \in \Theta$  can be obtained by minimizing the expected risk.

In practice, sampling  $\{\theta_t\}_{t=1}^T$  from the posterior (i.e.,  $\theta_t \sim p(\theta; \mathcal{D})$ ) is computationally expensive process. More specifically, we train a model with ERM and save the prediction results  $\hat{f}_{\theta_t}(x_i)$  of each sample on each iteration epoch  $t$ . Then, to avoid the influence of inaccurate predictions at the beginning of training, we estimate uncertainty with predictions after training  $T_s - 1$  epochs with:

$$u_i \approx \frac{1}{T} \sum_{t=T_s}^{T_s+T} \kappa(y_i, \hat{f}_{\theta_t}(x_i)). \quad (6)$$

During training, samples from the minority populations are classified correctly less frequently, which corresponds to higher training uncertainty.

To obtain reasonable importance weights, we assume that the samples with high uncertainty should be given a higher weight and vice versa. Therefore a reasonable importance weight could be linearly positively related to the corresponding uncertainty,

$$w_i = \eta u_i + c, \quad (7)$$

where  $\eta \in \mathbb{R}_+$  is a hyperparameter and  $c \in \mathbb{R}_+$  is a constant that keeps the weight to be positive. In practice, we set  $c$  to 1. The whole process for obtaining training importance weights is shown in Algorithm 2.

---

**Algorithm 2:** The process for obtaining training importance weights.

---

**Input:** Training dataset  $\mathcal{D}$ , sampling start epoch  $T_s$ , the number of sampling  $T$ , and upweight hyperparameter  $\eta$  ;

**Output:** The training importance weights  $\mathbf{w} = [w_1, \dots, w_n]$ ;

- 1 **for** *each* iteration **do**
  - 2     Train  $f_\theta$  by minimizing the expected risk  $\mathbb{E}\{\ell(\theta, x_i, y_i)\}$ ;
  - 3     Save the prediction results  $\{\hat{f}_{\theta_t}(x_i)\}_{i=1}^N$  of the current epoch  $t$ ;
  - 4 Obtain the uncertainty of each sample with  $u_i \approx \frac{1}{T} \sum_{t=T_s}^{T_s+T} \kappa(y_i, \hat{f}_{\theta_t}(x_i))$ ;
  - 5 Obtain the importance weight of each sample with  $w_i = \eta u_i + c$ .
-



## B.2 Datasets details

We describe the datasets used in the experiments in detail and summarize the datasets in Table 4.

- **WaterBirds** [59]. The task of this dataset is to distinguish whether the bird is a waterbird or a landbird. According to the background and label of an image, this dataset has four predefined subpopulations, i.e., “landbirds on land”, “landbirds on water”, “waterbirds on land”, and “waterbirds on water”. In the training set, the largest subpopulation is “landbirds on land” with 3,498 samples, while the smallest subpopulation is “landbirds on water” with only 56 samples.
- **CelebA** [44]. CelebA is a well-known large-scale face dataset. Same as previous works [42, 59], we employ this dataset to predict the color of the human hair as “blond” or “not blond”. There are four predefined subpopulations based on gender and hair color, i.e., “dark hair, female”, “dark hair, male”, “blond hair, female” and “blond hair, male” with 71,629, 66,874, 22,880, and 1,387 training samples respectively.
- **CivilComments** [9]. For this dataset, the task is to classify whether an online comment is toxic or not, where according to the demographic identities (e.g., Female, Male, and White) and labels, 16 overlapping subpopulations can be defined. We use 269,038, 45,180, and 133,782 samples as training, validation, and test datasets respectively.
- **Camelyon17** [5, 33]. Camelyon17 is a pathological image dataset with over 450, 000 lymph-node scans used to distinguish whether there is cancer tissue in a patch. The training data is drawn from three hospitals, while the validation and test data are sampled from other hospitals. However, due to the different coloring methods, even the same hospital samples have different distributions. Therefore, we cannot get reliable subpopulation labels of Camelyon17.

Table 4: Summary of the datasets used in the experiments.

Datasets	Labels	Groups	Population type	Data type	Backbone model
Waterbirds	2	2	Label×Group	Image	ResNet-50
CelebA	2	2	Label×Group	Image	ResNet-50
CivilComments	2	8	Label×Group	Text	DistilBERT-uncased
Camelyon17	2	5	Group	Image	DenseNet-121

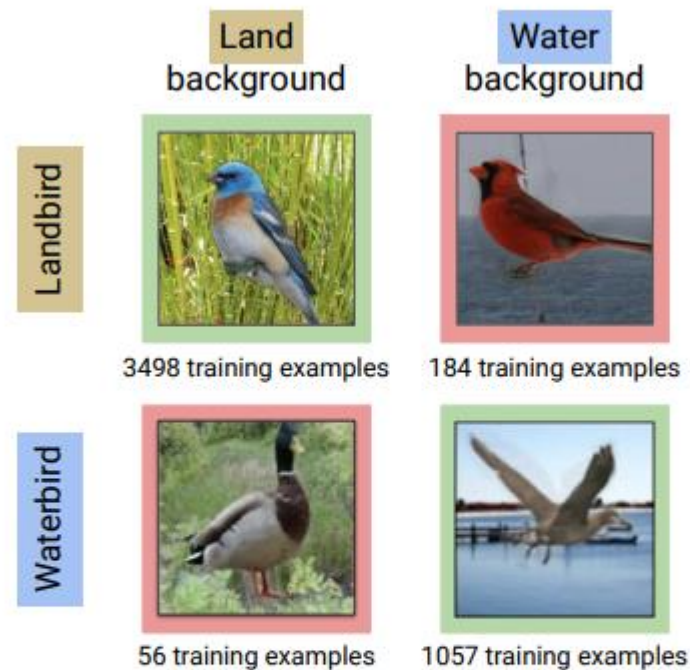


Figure 1. The four groups on the Waterbirds dataset, formed by the background spurious attribute and bird type label. Most training examples belong to the groups where the background matches the bird type (highlighted in green), while only a small fraction belong to the groups where the background does not match the bird type (highlighted in red).