M2177.0043 Introduction to Deep Learning

Lecture 23: Data Augmentation and Mixup

Hyun Oh Song[1]

[1]Dept. of Computer Science and Engineering, Seoul National University

June 18, 2020

Outline

Expected Risk

Definition

For a given data distribution $\mathcal{D}$ and a hypothesis set $\mathcal{F}$, **expected risk** R of $f \in \mathcal{F}$ is defined as

$$R(f) = \mathbb{E}_{(x,y)\sim\mathcal{D}}[\ell(f(x), y)],$$

where ℓ is a loss function.

Empirical Risk Minimization

- We estimate the expected risk by using observations $\{(x_i, y_i)\}_{i=1}^N$ as

$$\hat{R}(f) = \frac{1}{N} \sum_{i=1}^{N} \ell(f(x_i), y_i),$$

  which is called **empirical risk**.

- Minimizing the empirical risk is called Empirical Risk Minimization (ERM).

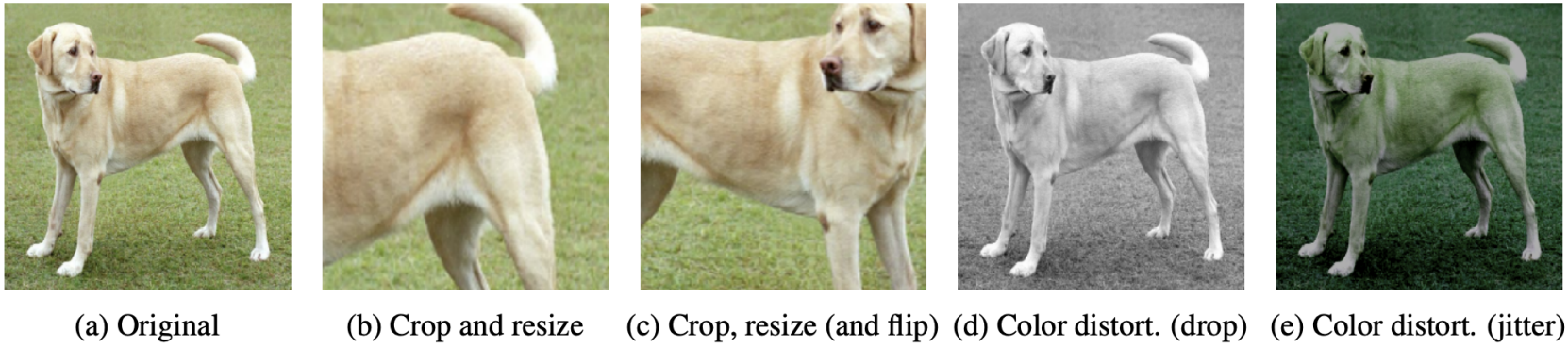
Problems of ERM

- Deep neural networks have powerful representation ability to memorize **random data**[1], *e.g.*, random Gaussian noise or random labels.
- Deep neural networks with naive ERM often **over-fit** to training data, which leads to poor generalization performance and undesirable behavior such as adversarial examples.

[1] Zhang et al., Understanding deep learning requires rethinking generalization, 2017.

Data Augmentation

- Data augmentation techniques, which generate data $\{(\tilde{x}_j, \tilde{y}_j)\}_{j=1}^{N'}$, are widely used to prevent the memorization of the neural network.



(a) Original (b) Crop and resize (c) Crop, resize (and flip) (d) Color distort. (drop) (e) Color distort. (jitter)

(f) Rotate $\{90°, 180°, 270°\}$ (g) Cutout (h) Gaussian noise (i) Gaussian blur (j) Sobel filtering

Figure: Various data augmentation techniques. (Original image cc-by: Von.grzanka)

Data Augmentation

- Transformations used in data augmentations **rely on priors**.
- For example, classification is invariant to rotation, but color inversion can affect the ground-truth class.
- Note that the priors depend on target tasks and data types, such as image, audio, text, etc.

Data Augmentation Optimization

- Recently, there is a line of works to optimize data augmentation policy. **AutoAugment**[2], proposes an RL based search algorithm to find the optimal combination of augmentation strategies.

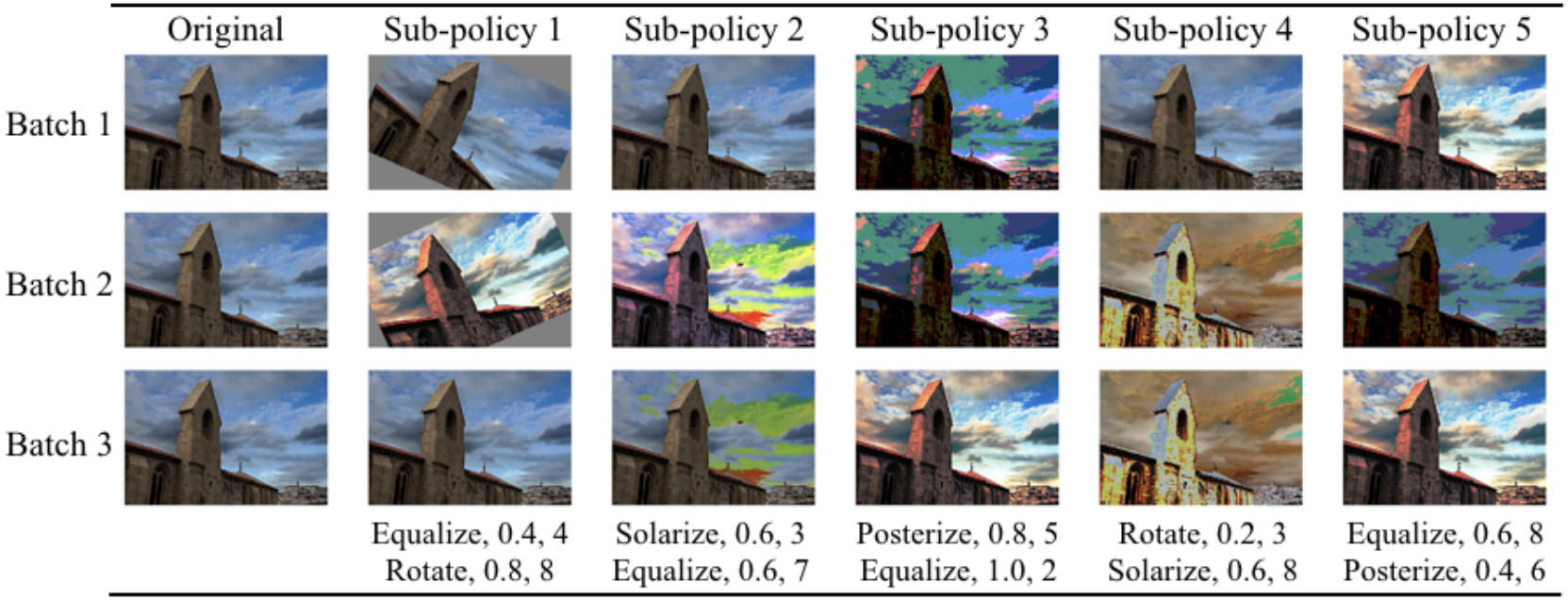


Figure: AutoAugment policy on ImageNet. For each batch, AutoAugment randomly selects one of the sub-polices, which each consists of multiple operations.

[2] Cubuk et al., AutoAugment: Learning Augmentation Strategies from Data, 2019

Outline

What is data mixup?

- **Mixup** mitigates over-confidence issue of deep neural networks by using **convex combinations** of data.

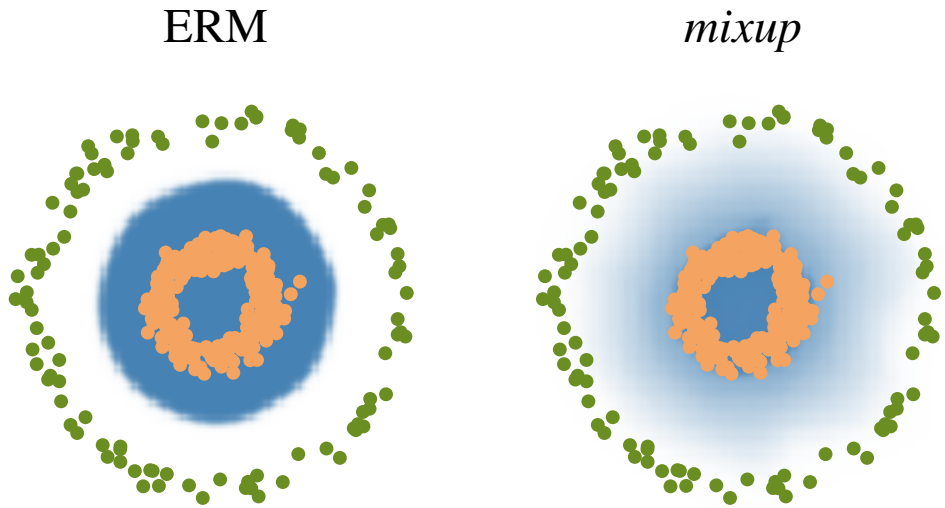


Figure: Comparison of predictions of binary classifiers trained with naive empirical risk minimization (ERM) and mixup[3]. Blue shading indicates $p(y = \text{orange}|x)$.

[3] Zhang et al., mixup: Beyond Empirical Risk Minimization, 2018

What is mixup?

- For a given pair of data and labels (x_1, y_1) and (x_2, y_2), **input mixup**[4] generates

$$(\lambda x_1 + (1 - \lambda)x_2, \lambda y_1 + (1 - \lambda)y_1),$$

  where $\lambda \in [0, 1]$.

- Input mixup encourages the network to predict linearly between training examples.

[4] Zhang et al., mixup: Beyond Empirical Risk Minimization, 2018

Toy experiments on mixup

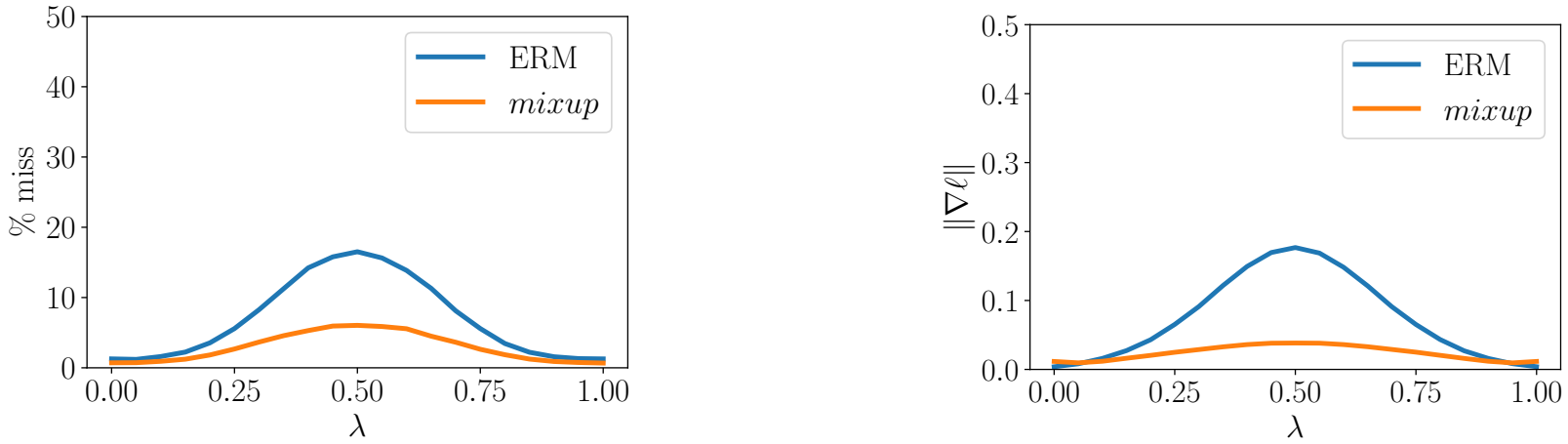


Figure: Performance of models trained with ERM and mixup. (Left) **Prediction errors** on in-between training data, $x = \lambda x_1 + (1 - \lambda)x_2$. The prediction is counted as a miss if it does not belong to $\{y_1, y_2\}$. (Right) **Norm of the gradients of the network** given the in-between training data x. The network trained with mixup has smaller gradient norms, indicating the stability of the network.

Why mixup works?

- Soft label **mitigates** the model from **over-confidence**.
- Since a mixing ratio λ is ranged in [0,1], previously unseen new data is generated at each step, **preventing** the model from **over-fitting** by memorizing the training data.

Generalization - CIFAR

- Mixup improves **generalization** with various models.

Dataset	Model	ERM	*mixup*
CIFAR-10	PreAct ResNet-18	5.6	**4.2**
	WideResNet-28-10	3.8	**2.7**
	DenseNet-BC-190	3.7	**2.7**
CIFAR-100	PreAct ResNet-18	25.6	**21.1**
	WideResNet-28-10	19.4	**17.5**
	DenseNet-BC-190	19.0	**16.8**

Table: Test errors for ERM and mixup on the CIFAR datasets.

Generalization - Speech data

- Mixup can be independently applied to various types of data and tasks.

Model	Method	Validation set	Test set
LeNet	ERM	**9.8**	**10.3**
	mixup ($\alpha = 0.1$)	10.1	10.8
	mixup ($\alpha = 0.2$)	10.2	11.3
VGG-11	ERM	5.0	4.6
	mixup ($\alpha = 0.1$)	4.0	3.8
	mixup ($\alpha = 0.2$)	**3.9**	**3.4**

Table: Speech recognition performance of mixup and ERM. The table reports classification errors of ERM and mixup on the Google commands dataset.

Robustness on label corruption

- Mixup prevents the network from **memorization**. As we can see from the table, network trained with mixup does not over-fit to data with corrupted labels.

Label corruption	Method	Test error		Training error	
		Best	Last	Real	Corrupted
20%	ERM	12.7	16.6	0.05	0.28
	ERM + dropout ($p = 0.7$)	8.8	10.4	5.26	83.55
	mixup ($\alpha = 8$)	**5.9**	6.4	2.27	86.32
	mixup + dropout ($\alpha = 4, p = 0.1$)	6.2	**6.2**	1.92	85.02

Table: Results on the corrupted label experiments for the best models. 20% of the labels in the training set is randomly corrupted. (Real) train error on normal 80% of the data (Corrupted) train error on corrupted 20%

Adversarial Robustness

- In addition, mixup is more robust on adversarial examples than ERM.

Metric	Method	FGSM	I-FGSM
Top-1	ERM	90.7	99.9
	mixup	**75.2**	99.6
Top-5	ERM	63.1	93.4
	mixup	**49.1**	95.8

Table: Classification errors of ERM and mixup tested on adversarial examples.

Outline

Manifold mixup

- **Manifold mixup**[5]:

$$\lambda f(x_1) + (1 - \lambda) f(x_2)$$

  for some hidden representation f.

- Manifold mixup generalizes input mixup and mixes data in the latent activation space (*i.e.* i-th conv layer activations).

- Manifold mixup empirically improves generalization performance better than input mixup.

[5] Verma et al., Manifold Mixup: Better Representations by Interpolating Hidden States, 2019

CutMix

- **CutMix**[6]:

$$\mathbb{1}_B \odot x_1 + (1 - \mathbb{1}_B) \odot x_2$$

for a binary rectangular mask $\mathbb{1}_B$, where the size of B is randomly chosen per each minibatch.

- CutMix is motivated from existing two data augmentation methods, **input mixup** and **Cutout**.

Figure: Figure illustrating mixup method. (Left) Clean image. (Middle Left) Image from input mixup. (Middle Right) Image from Cutout. (Right) Image from CutMix.

[6] CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features

Generalization

- CutMix empirically improves **generalization** more than other baselines.

Model	# Params	Top-1 Err (%)	Top-5 Err (%)
ResNet-101 (Baseline)	44.6 M	21.87	6.29
ResNet-101 + Cutout	44.6 M	20.72	5.51
ResNet-101 + Mixup	44.6 M	20.52	5.28
ResNet-101 + CutMix	44.6 M	**20.17**	**5.24**
ResNeXt-101 (Baseline)	44.1 M	21.18	5.57
ResNeXt-101 + CutMix	44.1 M	**19.47**	**5.03**

Table: Test errors for CutMix and other baselines on ImageNet datasets.

Localization

- CAM activation map selectively highlights relevant classes for CutMix

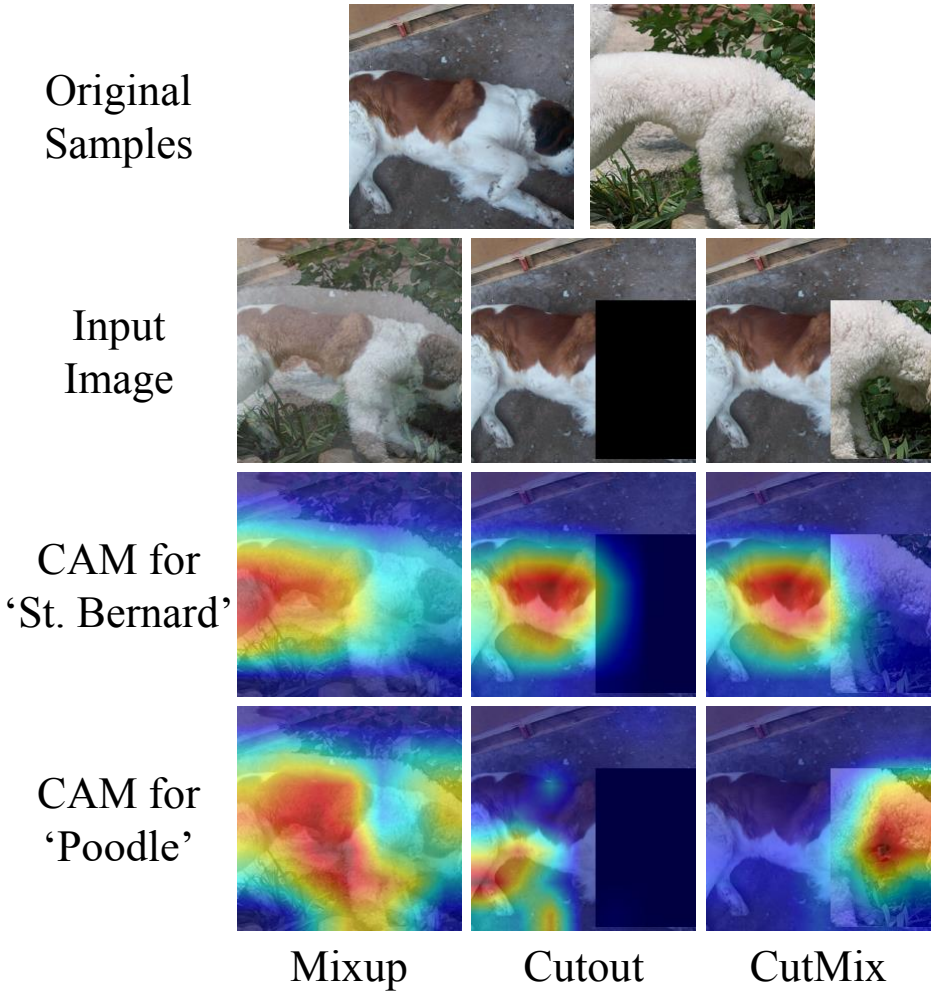


Figure: Class activation mapping (CAM) visualization. Compared to input mixup or Cutout, CutMix helps model to find where the object is located.

Localization

- CAM activation map selectively highlights relevant classes for CutMix

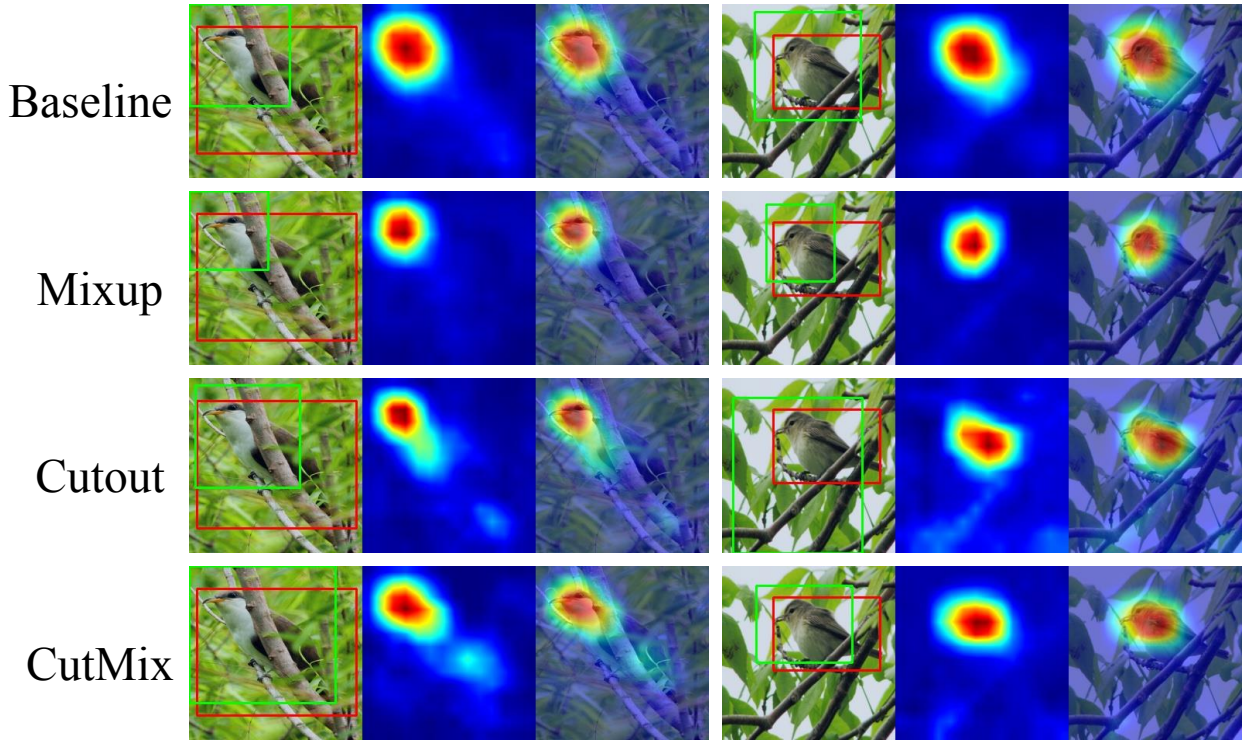


Figure: Qualitative comparison for weakly supervised object localization task on CUB-200-2011 dataset. Red: Ground truth. Green: Predicted bounding box. CutMix generates bounding box closest to the ground truth.

Localization

- Quantitative result comparing localization performance.

Method	CUB200-2011 Loc Acc (%)	ImageNet Loc Acc (%)
VGG-GAP + CAM	37.12	42.73
VGG-GAP + ACoL*	45.92	45.83
VGG-GAP + ADL*	52.36	44.92
GoogLeNet + HaS*	-	45.21
InceptionV3 + SPG*	46.64	48.60
VGG-GAP + Mixup	41.73	42.54
VGG-GAP + Cutout	44.83	43.13
VGG-GAP + CutMix	**52.53**	**43.45**
ResNet-50 + CAM	49.41	46.30
ResNet-50 + Mixup	49.30	45.84
ResNet-50 + Cutout	52.78	46.69
ResNet-50 + CutMix	**54.81**	**47.25**

Figure: Quantitative comparison for weakly supervised object localization task on CUB-200-2011 and ImageNet dataset.

Adversarial Robustness

- In addition, CutMix is more robust on FGSM adversarial examples than other baselines.

	Baseline	Mixup	Cutout	CutMix
Top-1 Acc (%)	8.2	24.4	11.5	**31.0**

Table: Classification errors of CutMix and other baselines adversarial examples.

Out of Distribution Detection

- Out-of-distribution (OOD) detection is about determining whether the sample is in- or out-of-distribution by thresholding based on maximum softmax probability.
- For example, if the model is trained with CIFAR-100 dataset, images from CIFAR-10 or uniform noise are out-of-distribution samples.
- CutMix shows good performance on OOD detection.

Method	TNR at TPR 95%	AUROC	Detection Acc.
Baseline	26.3 (+0)	87.3 (+0)	82.0 (+0)
Mixup	11.8 (-14.5)	49.3 (-38.0)	60.9 (-21.0)
Cutout	18.8 (-7.5)	68.7 (-18.6)	71.3 (-10.7)
CutMix	**69.0** (+42.7)	**94.4** (+7.1)	**89.1** (+7.1)

Figure: Out-of-distribution (OOD) detection results with CIFAR-100 trained models. Results are averaged on seven out-of-distribution samples including CIFAR-10, TinyImageNet, LSUN, uniform noise, etc.

Outline

Puzzle Mix: Exploiting Saliency and Local Statistics for Optimal Mixup

Jang-Hyun Kim, Wonho Choo, Hyun Oh Song

Seoul National University

ICML20

Problem of existing methods

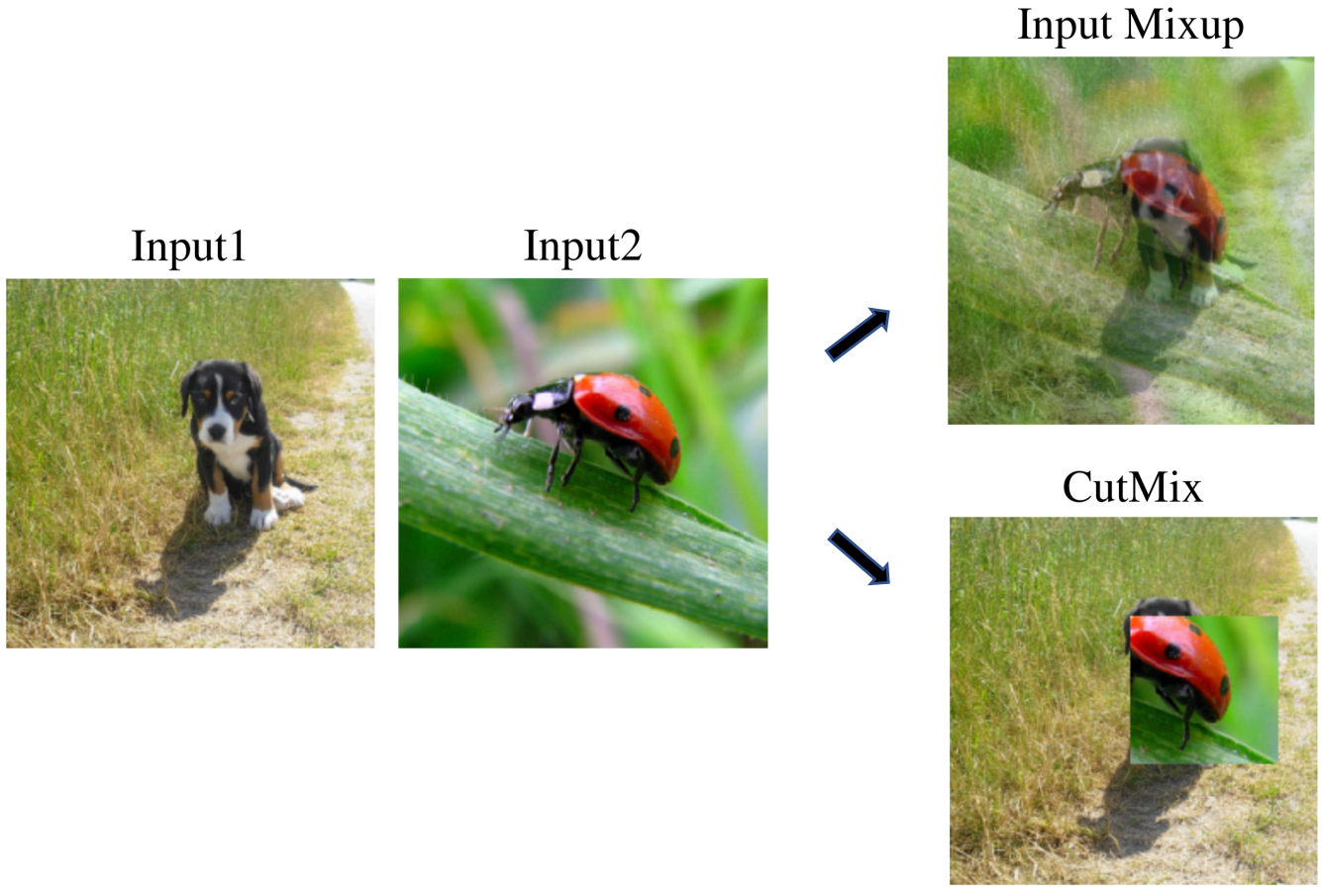


Figure: Image samples of existing mixup methods. Input mixup does not preserve local statistics (*e.g.*, *color*), and CutMix does not preserve salient information.

Step 1. Saliency

- First, we calculate a **saliency map**[7] of data.

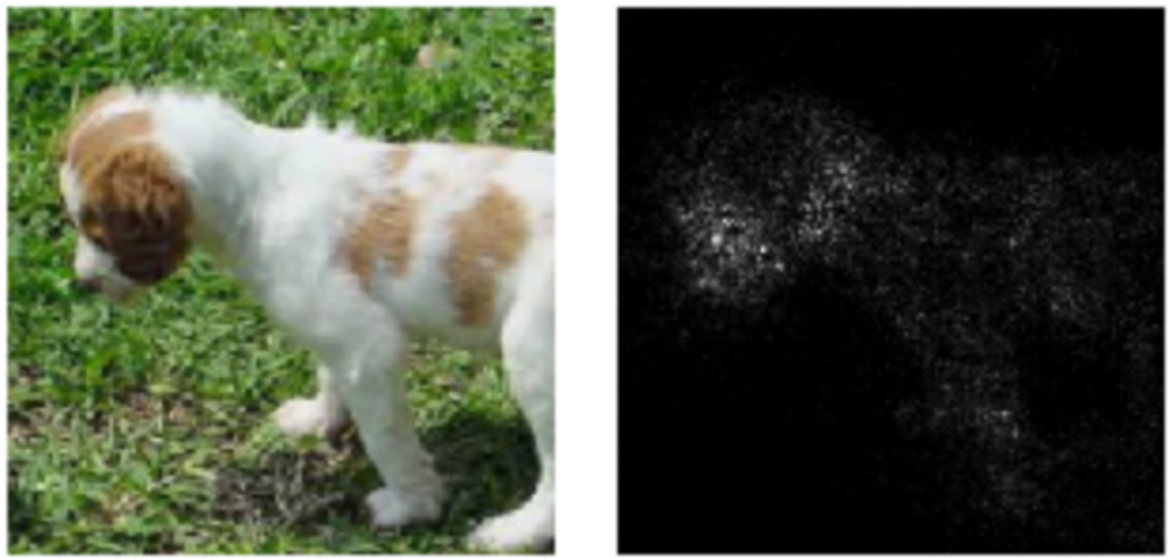


Figure: The saliency map(right) of the given image(left).

[7] Simonyan et al., Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013

Step 1. Saliency

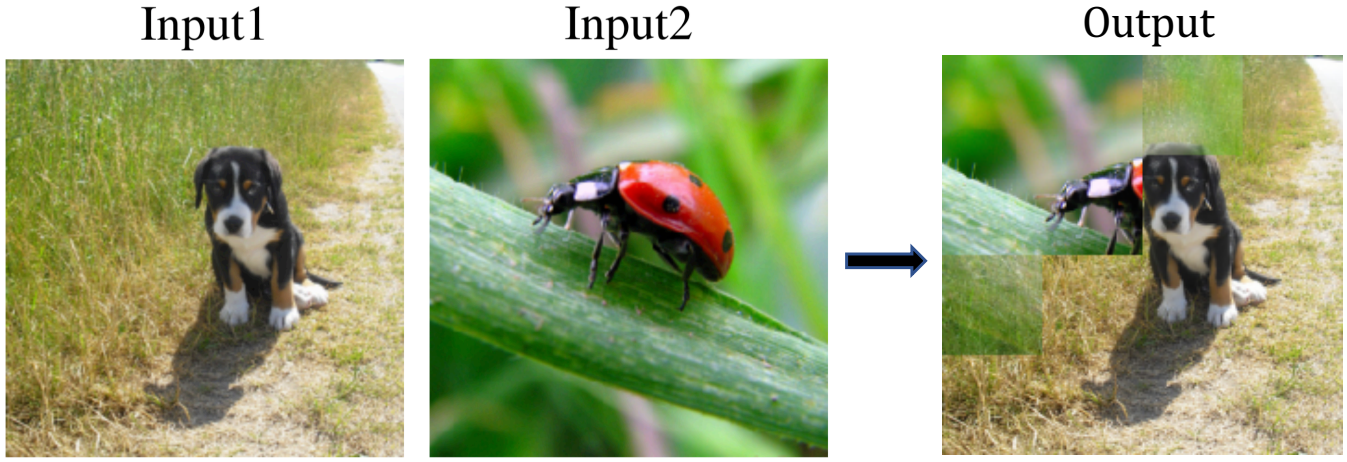
- First, we calculate a down-sampled **saliency map**[8] of data.

Figure: The down-sampled saliency map(right) of the given image(left).

[8] Simonyan et al., Deep inside convolutional networks: Visualising image classification models and saliency maps, 2013

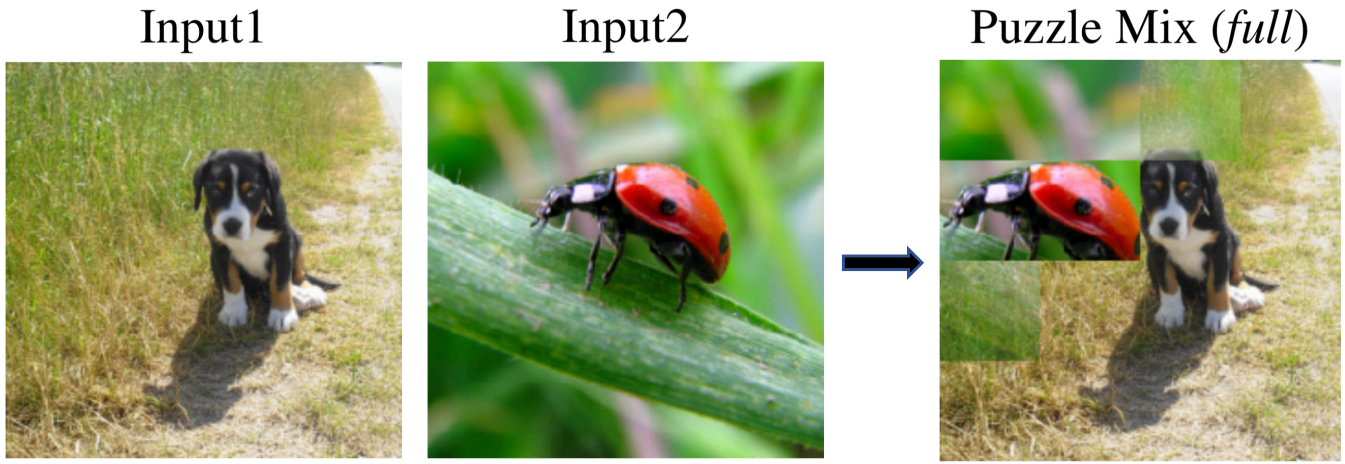
Step 2. Label assignment

- Based on the saliency, we assign **labels** indicating a **ratio between input sources** at each location of the mixup output.



Step 3. Transport

- Finally, we perform **transport** for each input to maximize saliency information.



Mixing Formula

- For a given pair of inputs $x_1, x_2 \in \mathbb{R}^n$, our **mixup formula** is

$$z \odot \Pi_1^\intercal x_1 + (1 - z) \odot \Pi_2^\intercal x_2,$$

where z represents a **mask** in $\mathcal{L}^n$ for $\mathcal{L} = \{\frac{k}{m} \mid k = 0, 1, \ldots, m\}$, and Π_k represents a $n \times n$ **transportation plan**. $\odot$ represents element-wise product.

Mixing Formula

- For a given pair of inputs $x_1, x_2 \in \mathbb{R}^n$, our **mixup formula** is

$$z \odot \Pi_1^\intercal x_1 + (1 - z) \odot \Pi_2^\intercal x_2,$$

  where z represents a **mask** in $\mathcal{L}^n$ for $\mathcal{L} = \{\frac{k}{m} \mid k = 0, 1, \ldots, m\}$, and Π_k represents a $n \times n$ **transportation plan**. $\odot$ represents element-wise product.

- Note, a transportation plan Π satisfies $\Pi 1_n = 1_n$ and $\Pi^\intercal 1_n = 1_n$, where Π_{ij} encodes how much mass moves from location i to j.

Objective

- Our main objective is to **maximize saliency** s_k of each input after mixup as follows:

$$\begin{aligned} \underset{\substack{z\in\mathcal{L}^n \\ \Pi_1,\Pi_2\in\{0,1\}^{n\times n}}}{\text{minimize}} \quad & -\|z\odot\Pi_1^\intercal s_1\|_1-\|(1-z)\odot\Pi_2^\intercal s_2\|_1 \\ \text{subject to} \quad & \Pi_k 1_n=1_n,\ \Pi_k^\intercal 1_n=1_n \quad \text{for } k=1,2. \end{aligned}$$

Objective

- To ensure **local statistics** of mixup data, we add **smoothness terms**; ψ for label smoothness and ϕ for data smoothness.

$$\begin{aligned}\underset{\substack{z\in\mathcal{L}^n\\ \Pi_1,\Pi_2\in\{0,1\}^{n\times n}}}{\text{minimize}} \quad & -\|z\odot\Pi_1^\intercal s_1\|_1-\|(1-z)\odot\Pi_2^\intercal s_2\|_1 \\ & +\beta\sum_{(i,j)\in\mathcal{N}}\psi(z_i,z_j)+\gamma\sum_{(i,j)\in\mathcal{N}}\phi_{i,j}(z_i,z_j) \\ \text{subject to} \quad & \Pi_k 1_n=1_n,\ \Pi_k^\intercal 1_n=1_n \quad \text{for } k=1,2.\end{aligned}$$

Figure: Puzzle Mix images with increasing smoothness coefficient β and γ.

Objective

- As the other mixup methods, we model a **mixing ratio** λ between inputs as a prior p defined as $mz_i \sim B(m, \lambda)$.

$$\underset{\substack{z\in\mathcal{L}^n \\ \Pi_1,\Pi_2\in\{0,1\}^{n\times n}}}{\text{minimize}} \quad -\|z\odot\Pi_1^\intercal s_1\|_1 - \|(1-z)\odot\Pi_2^\intercal s_2\|_1$$
$$+\beta\sum_{(i,j)\in\mathcal{N}}\psi(z_i,z_j)+\gamma\sum_{(i,j)\in\mathcal{N}}\phi_{i,j}(z_i,z_j)$$
$$-\eta\sum_i \log p(z_i)$$
$$\text{subject to}\ \ \Pi_k 1_n = 1_n,\ \Pi_k^\intercal 1_n = 1_n \quad \text{for } k=1,2.$$

Figure: Puzzle Mix images with increasing mixing ratio λ.

Objective

- Finally, we model **transport cost** to preserve local statistics of data.

$$\begin{aligned}\underset{\substack{z\in\mathcal{L}^n\\ \Pi_1,\Pi_2\in\{0,1\}^{n\times n}}}{\text{minimize}} \quad & -\|z\odot\Pi_1^\intercal s_1\|_1-\|(1-z)\odot\Pi_2^\intercal s_2\|_1 \\ & +\beta\sum_{(i,j)\in\mathcal{N}}\psi(z_i,z_j)+\gamma\sum_{(i,j)\in\mathcal{N}}\phi_{i,j}(z_i,z_j) \\ & -\eta\sum_i\log p(z_i)+\xi\sum_{k=1,2}\langle\Pi_k,C\rangle \\ \text{subject to} \quad & \Pi_k 1_n=1_n,\ \Pi_k^\intercal 1_n=1_n \quad \text{for } k=1,2.\end{aligned}$$

Algorithm

- We propose **2-stage alternating** optimization.
  - Optimizing mask z given the fixed transport Π_k
  - Optimizing transport Π_k given the fixed mask z

Algorithm

- We propose **2-stage alternating** optimization.
  - Optimizing mask z given the fixed transport Π_k
  - Optimizing transport Π_k given the fixed mask z
- Empirically, in most cases, the algorithm converges after one cycle.

Algorithm–Optimizing z

- The main objective with respect to z can be represented as a **submodular minimization** problem:

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

where $u_i(z_i) = (1 - z_i)\ (\Pi_1^\intercal s_1)_i + z_i\ (\Pi_2^\intercal s_2)_i$.

[9] Boykov et al., Fast approximate energy minimization via graph cuts, 2001

Algorithm–Optimizing z

- The main objective with respect to z can be represented as a **submodular minimization** problem:

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

  where $u_i(z_i) = (1 - z_i)\ (\Pi_1^\intercal s_1)_i + z_i\ (\Pi_2^\intercal s_2)_i$.

- We apply **alpha-beta swap**[9] algorithm to optimize z.

[9] Boykov et al., Fast approximate energy minimization via graph cuts, 2001

Algorithm–Optimizing Π

- The main objective with respect to Π_k under the mask z becomes

$$\underset{\Pi_1,\Pi_2\in\{0,1\}^{n\times n}}{\text{minimize}} \; -\left\|z\odot\Pi_1^\intercal s_1\right\|_1-\left\|(1-z)\odot\Pi_2^\intercal s_2\right\|_1$$
$$+\xi\sum_{k=1,2}\langle\Pi_k,C\rangle$$
$$\text{subject to}\;\; \Pi_k 1_n=1_n,\;\Pi_k^\intercal 1_n=1_n \quad \text{for } k=1,2.$$

Algorithm–Optimizing Π

- The main objective with respect to Π_k under the mask z becomes

$$\begin{aligned}\underset{\Pi_1,\Pi_2\in\{0,1\}^{n\times n}}{\text{minimize}} \quad & -\left\|z\odot\Pi_1^\intercal s_1\right\|_1-\left\|(1-z)\odot\Pi_2^\intercal s_2\right\|_1 \\ & +\xi\sum_{k=1,2}\langle\Pi_k,C\rangle \\ \text{subject to} \quad & \Pi_k 1_n=1_n,\ \Pi_k^\intercal 1_n=1_n \quad \text{for } k=1,2.\end{aligned}$$

- We develop an **approximate algorithm** that can be parallelized on GPUs and efficiently computed in batches.

Generalization

- Calculating saliency requires an additional forward-backward evaluation of the network, we additionally compare the performances at **a fixed number of network evaluations** (half).

Method	**Top-1 Error(%)**	**Top-5 Error(%)**
Vanilla	21.14	6.33
Input	18.27	4.98
Manifold	**17.40**	**4.37**
Manifold†	18.04	-
CutMix	17.50	4.69
Puzzle Mix		
Puzzle Mix (half)		

Table: Top-1 / Top-5 error rates on CIFAR-100 of WRN28-10 trained with 400 epochs (200 epochs for half). † denotes the result reported in the original paper.

Generalization

- Calculating saliency requires an additional forward-backward evaluation of the network, we additionally compare the performances at **a fixed number of network evaluations** (half).

Method	**Top-1 Error(%)**	**Top-5 Error(%)**
Vanilla	21.14	6.33
Input	18.27	4.98
Manifold	17.40	4.37
Manifold†	18.04	-
CutMix	17.50	4.69
Puzzle Mix	**15.95**	3.92
Puzzle Mix (half)	16.23	**3.90**

Table: Top-1 / Top-5 error rates on CIFAR-100 of WRN28-10 trained with 400 epochs (200 epochs for half). † denotes the result reported in the original paper.

Extension

- After the saliency calculation, we can obtain the followings:
  - **gradient** information with respect to **input data**
  - **gradient** information with respect to **network weights**

Adversarial training

- There is a **trade-off**[10] between generalization and adversarial robustness.

[10] Madry et al., Towards deep learning models resistant to adversarial attacks, 2017

Adversarial training

- There is a **trade-off**[10] between generalization and adversarial robustness.
- To prevent possible degradation of generalization, we **probabilistically** add adversarial perturbation to input data.

[10] Madry et al., Towards deep learning models resistant to adversarial attacks, 2017

Utilizing clean data

- We utilize gradient information with respect to network weights from clean data as **regularization**:

$$\nabla_\theta \ell(x_{mixup}; \theta) + \frac{1}{2}\lambda_c(\nabla_\theta \ell(x_1; \theta) + \nabla_\theta \ell(x_2; \theta)),$$

  where ℓ denotes the objective function of neural networks.

Utilizing clean data

- We utilize gradient information with respect to network weights from clean data as **regularization**:

$$\nabla_\theta \ell(x_{mixup}; \theta) + \frac{1}{2}\lambda_c(\nabla_\theta \ell(x_1; \theta) + \nabla_\theta \ell(x_2; \theta)),$$

  where ℓ denotes the objective function of neural networks.

- The regularization **stabilizes** training and **improves** generalization in some cases.

Tiny-ImageNet

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**
Puzzle Mix			
Puzzle Mix (half)			
Puzzle Mix (adv)			
Puzzle Mix (half, adv)			

Table: Top-1 / Top-5 / FGSM error rates on Tiny-ImageNet dataset for PreActResNet18. *adv* denotes an adversarially trained model, and *half* indicates a model trained for half the epochs.

Tiny-ImageNet

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
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: Top-1 / Top-5 / FGSM error rates on Tiny-ImageNet dataset for PreActResNet18. *adv* denotes an adversarially trained model, and *half* indicates a model trained for half the epochs.