Logit Mixing Training for More Reliable and Accurate Prediction

Duhyeon Bang[1*], **Kyungjune Baek**[2*], **Jiwoo Kim**[3], **Yunho Jeon**[4], **Jin-Hwa Kim**[5], **Jiwon Kim**[1], **Jongwuk Lee**[3] and **Hyunjung Shim**[6†]
[1]SK T-Brain
[2]School of Integrated Technology, Yonsei University
[3]Department of Software, Sungkyunkwan University
[4]MOFL
[5]NAVER AI Lab
[6] Kim Jaechul Graduate School of AI, KAIST

Abstract

When a person solves the multi-choice problem, she considers not only what is the answer but also what is not the answer. Knowing what choice is not the answer and utilizing the relationships between choices, she can improve the prediction accuracy. Inspired by this human reasoning process, we propose a new training strategy to fully utilize inter-class relationships, namely *LogitMix*. Our strategy is combined with recent data augmentation techniques, *e.g.*, Mixup, Manifold Mixup, CutMix, and PuzzleMix. Then, we suggest using a mixed logit, *i.e.*, a mixture of two logits, as an auxiliary training objective. Since the logit can preserve both positive and negative inter-class relationships, it can impose a network to learn the probability of wrong answers correctly. Our extensive experimental results on the image- and language-based tasks demonstrate that LogitMix achieves state-of-the-art performance among recent data augmentation techniques regarding calibration error and prediction accuracy.

1 Introduction

Humans can improve predictions by knowing how likely non-target classes are misclassified into a target class. For example, suppose that we have an image of *computer with apple logo* and should identify an object out of three choices, *e.g.*, *computer*, *apple* and *orange*. Here, we can exploit the fact that *apple* and *orange* have positive correlations and *computer* and *orange* have no correlation. Since *computer with apple logo* is equally and highly probable for being both *computer* and *apple*, we can utilize the probability of *orange* being an answer; it is a small, non-zero value because orange is similar to apple. Then, both the new information about *orange* and the sum of all probabilities being 1 substantially affect the overall decision; the probability of *apple* being an answer can be less probable than that of *computer*. Consequently, we can correctly predict *computer* as the final answer. This example shows that understanding inter-class relationships can be particularly beneficial when handling more challenging and unusual decision tasks.

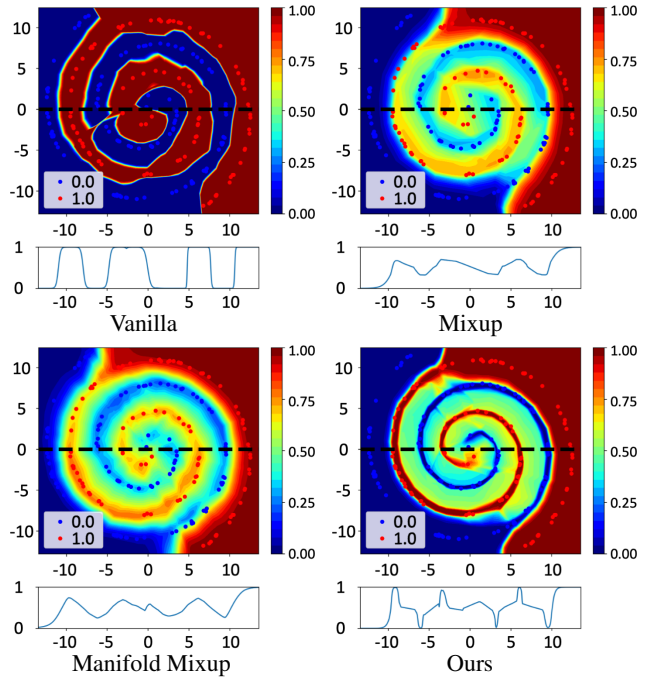

Figure 1: Visualization of the probability distribution on various training strategies on the 2D spiral dataset. Compared to other methods, our method shows well-calibrated prediction and smooth transition between two classes. Please refer to Section 4.1 for details.

Unlike humans, conventional deep neural networks (DNNs) are poor at understanding class relationships. Since the one-hot label induces zero probability for non-target class labels, DNNs learn to de-correlate the relationship between the target class and the other classes [Li and Maki, 2018]. Even worse, when the model overfits to estimate a target class, its prediction tends to be over-confident [Guo *et al.*, 2017], which is particularly vulnerable to high-risk systems, *e.g.*, medical diagnosis and autonomous driving systems.

Inspired by the human reasoning process, we propose a new training strategy using inter-class relationships for holistic recognition. Specifically, we exploit *logits* for design-

*indicates an equal contribution.
†is a corresponding author.

ing an additional learning signal and develop our training strategy in conjunction with mixing-based data augmentation [Verma *et al.*, 2019; Yun *et al.*, 2019; Zhang *et al.*, 2018; Kim *et al.*, 2020]. Given the baseline mixing-based method, we randomly choose two training examples. The data is generated by linearly combining two examples, and the logit is also derived by the same linear combination of two logits (*i.e.*, two logits are the logit of each example). While the existing method imposes the model to map the mixed data to the mixed label, we additionally introduce two supervision and a joint training strategy. In other words, the model should jointly learn to map both 1) from the mixed data to the mixed logit and 2) from the original training samples to the labels.

We name it LogitMix. Specifically, the mixed logit is utilized as weak supervision for considering inter-class relationships (*e.g.*, how likely the wrong classes are the answer). Simultaneously, the original training data is used as the strong supervision for learning class-only information (*e.g.*, relying only on how likely the correct class is the answer). While mixing-based methods also enjoy the effect of learning two-class relationships, they are limited in two aspects. Firstly, the participation of only two-class relationships can be insufficient to resolve ambiguity in multi-choice problems. Secondly, their mixed labels are the result after activation (*i.e.*, softmax), thereby losing negative relationships and precision. LogitMix can effectively learn inter-class relationships (*i.e.*, logit reveals all-class relationships) with high precision (*i.e.*, logit provides negative and positive values with high dynamic range). Moreover, since LogitMix faithfully learns the original training data, we can extract meaningful logits for weak supervision. Then, logits can supervise the model to learn accurate confidence of training data. As a result, our logit mixing strategy can induce a well-calibrated and highly accurate model, as shown in Figure 1. Here, the vanilla model can predict 0 and 1 accurately, but has sharp decision boundaries, causing a poorly calibrated prediction. While existing studies, such as Mixup or Manifold Mixup, induce smooth boundaries but sacrifice confidence in training data, LogitMix enjoys the benefit from both sides, showing the improved boundaries without losing the prediction accuracy on training samples.

In summary, LogitMix is a powerful training strategy for reflecting both positive and negative inter-class relationships in model training. Based on the extensive experiments on three popular image benchmark datasets, LogitMix outperforms state-of-the-art data mixing training techniques, such as Mixup [Zhang *et al.*, 2018], CutMix [Yun *et al.*, 2019], and PuzzleMix [Kim *et al.*, 2020], in terms of prediction accuracy and calibration error. Also, we adopt LogitMix using the famous pre-trained language model, *i.e.*, BERT [Devlin *et al.*, 2018], to solve eight popular NLP tasks on the GLUE benchmark datasets. The experimental results confirm that LogitMix on BERT can show better prediction accuracy than the existing mixing-based method on BERT.

2 Related Work

Mixing-based augmentation. Mixup [Zhang *et al.*, 2018] trains the model using the data generated by the convex combination of two random training samples and their labels. Despite its simplicity, Mixup training achieves robustness toward adversarial examples [Zhang *et al.*, 2018] and improves calibration [Thulasidasan *et al.*, 2019]. Manifold Mixup [Verma *et al.*, 2019] uses two intermediate representations at layer k as examples. To achieve better performance in spatial examples, [Yun *et al.*, 2019] introduce a region-based augmentation strategy. By randomly creating a binary mask, it generates the mixed example by adding the masked image and the inversely masked other images. Unlike CutMix, PuzzleMix [Kim *et al.*, 2020] recently suggests a saliency-aware mixing strategy to mitigate the misguiding examples during training.

Although various mixing-based techniques have been actively studied, their success is mostly limited to computer vision tasks. Most recently, the idea of Mixup has been applied to the pre-trained language model, namely Mixup-Transformer [Sun *et al.*, 2020], which demonstrates the potential of mixing-based techniques on language-based tasks. Note that our method can utilize any mixing-based approaches as the baselines and further improve their performances by exploiting the mixed logit as the additional weak supervision in both computer vision and natural language processing tasks.

Logit regularization. [Szegedy *et al.*, 2016] propose label smoothing that modifies a one-hot label to a mixture of the one-hot label and uniform distribution by dividing a weight factor. [Pereyra *et al.*, 2017] point out the limitation of label smoothing that allocates the same probability across all classes regardless of their relationships. Then, they propose a confidence regularizer by penalizing low entropy predictions. As a pioneering work, [Hinton *et al.*, 2014] propose knowledge distillation, which gives insight into how logits have useful information on the model and data distribution. Motivated by these observations, we exploit the logit to develop a mixing-based training strategy by preserving inter-class correlations.

Confidence calibration. [Naeini *et al.*, 2015] diagnoses the problem of the confidence calibration of modern networks and proposes the expected calibration error (ECE) as a measurement of calibration. [Guo *et al.*, 2017] addresses that the modern network is poorly calibrated and suggests a simple post-processing calibration method to soften the prediction with temperature scaling. Various Bayesian approaches [MacKay, 1992] are commonly used for estimating the uncertainty of the prediction. Nevertheless, these methods are computationally inefficient as they require some modifications to the training procedure. [Lakshminarayanan *et al.*, 2017] approximate the Bayesian method by using an ensemble of networks. [Seo *et al.*, 2019] propose a well-calibrated prediction scheme without multiple stochastic inferences. This method is related to label smoothing [Szegedy *et al.*, 2016; Müller *et al.*, 2019] and confidence penalty [Pereyra *et al.*, 2017] as it makes the network output smooth prediction. [Thulasidasan *et al.*, 2019] have empirically shown that the network trained with Mixup provides better-calibrated results.

3 Proposed Model

3.1 Mixing-Based Augmentation Techniques

Recently, mixing-based data augmentation techniques [Verma *et al.*, 2019; Yun *et al.*, 2019; Zhang *et al.*, 2018] have achieved

highly accurate prediction performance. They generate the input data by a linear combination of two randomly selected training datasets. Likewise, the label is generated by the same linear combination of the two corresponding labels. In this way, they effectively improve prediction accuracy and prevent undesirable behaviors such as memorization and sensitivity to adversarial examples at the same time. Furthermore, recent work [Thulasidasan *et al.*, 2019] reports that Mixup training encourages that the output of DNNs, *i.e.*, the estimated label distribution, can serve as a better indicator of the actual likelihood of a correct prediction. The mixing-based training strategies generate an augmented sample as follows:

$$x_{mix} = \begin{cases} \lambda x_1 + (1-\lambda)x_2 \text{ [Zhang } et\ al.\text{, 2018]}, \\ \mathbf{M} \odot x_1 + (1-\mathbf{M}) \odot x_2 \text{ [Yun } et\ al.\text{, 2019]}, \\ \mathbf{Z} \odot \mathbf{\Pi}_1^T x_1 + (1-\mathbf{Z}) \odot \mathbf{\Pi}_2^T x_2 \text{ [Kim } et\ al.\text{, 2020]}, \end{cases} \quad (1)$$
$$y_{mix} = \lambda y_1 + (1-\lambda) y_2, \quad \lambda \sim \mathrm{Beta}(\alpha, \alpha),$$

where (x_1, y_1) and (x_2, y_2) denote two random training samples. $\mathbf{M} \in \{0,1\}^{W \times H}$ denotes a binary mask indicating where to drop out and fill in from two images, and $\odot$ is an element-wise multiplication. $\mathbf{Z}$ is a soft mask with a value within $[0, 1]$, when $\lambda = \sum_i \mathbf{Z}_i$ and $\mathbf{\Pi}_0$ (or $\mathbf{\Pi}_1$) represents the transport matrix. Here, each element Π_{ij} encodes the ratio of the transport from location i to j (*i.e.* i and j for the block index). $\mathrm{Beta}(\cdot, \cdot)$ implies a Beta distribution, and $\alpha \in (0, \infty)$ is the parameter to control the shape of the Beta distribution. Using the mixed input and the mixed label, the model is trained to minimize the following loss function.

$$\mathcal{L}_{mix} = \lambda \mathcal{H}(\tilde{y}_{mix}, y_1) \ + (1-\lambda)\mathcal{H}(\tilde{y}_{mix}, y_2), \quad (2)$$

where $\mathcal{H}$ is the cross-entropy function. $\tilde{y}_{mix}$ is the predicted label vector by the model, *e.g.* $\tilde{y}_{mix} = \mathtt{softmax}(f(x_{mix}))$, where f denotes the model. That is, $f(x_{mix})$ indicates *logit*, the model output for the mixed input x_{mix} before the softmax activation function.

Whereas the original label y is encoded as a one-hot vector, the mixed label y_{mix} allows two non-zero entries in the label distribution. Here, the model learns two-class relationships encoded in the mixed labels, which empirically yield the label smoothing effect. Lately, [Thulasidasan *et al.*, 2019] show that the label smoothing effect is a crucial factor for achieving accurate predictive uncertainty. Mixing-based techniques can also be regarded as a robust data augmentation scheme. [Thulasidasan *et al.*, 2019] also show that data augmentation alone without mixed labels can improve the prediction accuracy.

3.2 LogitMix

To utilize inter-class relationships (*i.e.*, all-class relationships), LogitMix aims to achieve both high prediction accuracy and reliable prediction confidence. Specifically, we devise the mixed data augmentation trick like [Yun *et al.*, 2019; Zhang *et al.*, 2018] to explicitly consider the inter-class relationships. Unlike existing techniques, we additionally suggest two loss terms: 1) the estimated logit for the mixed data should match the target mixed logit, and 2) the estimated label distribution for each sample should match the one-hot label. In the first loss term, we assigned weak supervision for the logit of mixed data as the mixture of two logits. We refer to it as weak supervision as the mixed logit is not oracle supervision. Since the mixed data is generated by any existing mixing-based technique, we use it as the baseline model and thus incorporate $\mathcal{L}_{mix}$ into our final loss function. Formally, the objective of LogitMix is formulated by the sum of three losses:

$$\begin{aligned} \mathcal{L}_{total} &= \mathcal{L}_{cls} + \mathcal{L}_{sim} + \mathcal{L}_{mix}, \\ \mathcal{L}_{cls} &= \mathcal{H}(\tilde{y}_1, y_1) + \mathcal{H}(\tilde{y}_2, y_2) \\ \mathcal{L}_{sim} &= \|(\lambda f(x_1) + (1-\lambda) f(x_2)) - f(x_{mix})\|_2. \end{aligned} \quad (3)$$

To derive $\mathcal{L}_{sim}$, namely the similarity loss, we generate a mixed logit by linearly combining two logits $f(x_1)$ and $f(x_2)$ for two random samples x_1 and x_2. Then, the mixed logit should be matched to the logit $f(x_{mix})$ of the mixed sample x_{mix}. It is motivated by previous studies [Mikolov *et al.*, 2013] that linear interpolation is an effective way of combining factors. Besides, various studies report linear interpolation between the hidden representations results in the meaningful region of embeddings [Verma *et al.*, 2019]. Therefore, we utilize a simple linear mixture of two logits as weak supervision.

Why LogitMix helps? The idea of utilizing logit as a learning signal is used to develop $\mathcal{L}_{sim}$. This similarity loss enforces the network to map the mixed input to the mixed logit. Then, a natural question can be raised as to why logit should be used, not a probability (*i.e.*, a post-softmax value). Here, we argue that the post-softmax values generally lose the rich information encoding class relationships. This argument can be demonstrated by the following example. Focusing on the N-ary classification task, we denote two mixed inputs as x^1_{mix} and x^2_{mix}, respectively, and their logits as $l_1 = [1.5, 0.5, \ldots]$ and $l_2 = [0.5, -0.5, \ldots]$, respectively. Although the two logit vectors are clearly different, the first two elements of their post-softmax results are the same as $[0.73, 0.27, \ldots]$. When adopting the post-softmax values as our signals, the two different mixed data results in the same class relationships. However, x^2_{mix} in its logit exhibits a negative relationship between class 1 and class 2, which is observable from neither x^1_{mix} nor the post-softmax of x^2_{mix}; this unusual information can be valuable information to train the model [Heo *et al.*, 2019].

Based on this example analysis, we can easily find the disadvantages of the post-softmax values in learning inter-class relationships. Since the softmax reduces the dynamic range ($[+\infty, -\infty] \rightarrow [0, 1]$) of the class activation significantly, valuable information such as negative correlations and precision can be abandoned. On the other hand, the logit can encode both positive and negative all-class relationships with a high dynamic range. Thanks to this desirable property, $\mathcal{L}_{sim}$ is more effective than $\mathcal{L}_{mix}$ in learning class relationships. $\mathcal{L}_{mix}$ and $\mathcal{L}_{sim}$ also induce different types of learning signals. Therefore, the two-loss terms can be used simultaneously and enjoy performance gains from others.

4 Proof-of-Concept Study

4.1 Toy Example

We conducted a 2D classification task using the two-class spiral dataset [Verma *et al.*, 2019] to reveal the impact of LogitMix. To show the decision boundary and hidden representation of each method, we visualize the predictions in

different colors; two-class samples are noted as either red or blue color-coded points. Then, we compare LogitMix with the other mixing augmentation methods. Because it is non-trivial to apply region-based methods such as CutMix and PuzzleMix (a point does not have spatial information), we compared LogitMix with vanilla, Mixup [Zhang *et al.*, 2018], and Manifold Mixup [Verma *et al.*, 2019], as shown in Fig. 1.

Specifically, vanilla training results in irregular boundaries and the predictions for the out-of-distribution samples are over-confident; the predictions for the out-of-distribution samples exhibit as high confidence as in-distribution samples. Also, the sharp transition around the decision boundaries indicates the vulnerability of adversarial attacks [Goodfellow *et al.*, 2015]. In the case of Mixup, we observe a gradual transition in the inter-class area having an intensity of 0.3 to 0.7 around the decision boundaries; however, a part of those observations originates from lower-confidence on training samples. Manifold Mixup has more precise decision boundaries than the two previous cases, especially on the transition. However, overall confidence tends to be lower than others. One possible reason is under-fitting since its complicated training prevents sufficient convergence for the conventional learning procedure [Thulasidasan *et al.*, 2019]. In contrast, LogitMix improves high-confident samples (*i.e.* high confidence on training samples) and shows the gradual transition over inter-class regions as a supportive clue of a well-calibrated confidence estimation model.

4.2 Combination of Losses

Existing mixing-based techniques only use $\mathcal{L}_{mix}$; they utilize the original training samples depending on sampling results from Beta$(\cdot,\cdot)$ but do not learn the original training data simultaneously. Meanwhile, LogitMix utilizes both $\mathcal{L}_{cls}$ and $\mathcal{L}_{sim}$ on top of $\mathcal{L}_{mix}$. Here, to understand the role of each loss term and its synergy with $\mathcal{L}_{mix}$, we investigate various combinations of losses and report their classification accuracy with ResNet50 on the CIFAR100 dataset, which consists of 50,000 images with 100 classes. More training details can be found in Section 5.

Table 1 demonstrates the result of various combinations of losses; LogitMix$_m$ indicates that Mixup is chosen as our baseline mixing strategy. In this study, we have three important observations. Firstly, our similarity loss ($\mathcal{L}_{sim}$) always increases the accuracy. Specifically, both the vanilla model and Mixup are improved by adding $\mathcal{L}_{sim}$. Without $\mathcal{L}_{sim}$, the accuracy of LogitMix, *i.e.*, $\mathcal{L}_{mix} + \mathcal{L}_{cls}$, is significantly degraded; the accuracy decreases by -1.51%p.

We also observe that removing $\mathcal{L}_{mix}$ significantly decreases the accuracy (-1.48%p). It is because the logit can be noisy at the early stage of the training without $\mathcal{L}_{mix}$, therefore, the effect of $\mathcal{L}_{sim}$ can be degraded.

Another interesting observation is that combining $\mathcal{L}_{cls}$ on existing Mixup loss also improves the baseline Mixup. Since $\mathcal{L}_{cls}$ provides more accurate feedback for training data and $\mathcal{L}_{mix}$ serves as a regularization term, their joint training can enjoy the synergy, achieving 0.26%p of gains over the original Mixup. However, we observe that training with $\mathcal{L}_{mix} + \mathcal{L}_{cls}$ is less stable (*i.e.* high variance) than all other combinations.

Finally, LogitMix utilizes all three losses and records the best accuracy among all combinations, also a low variance. Specifically, LogitMix$_m$ achieves 1.77%p of the improvement over Mixup. Mixup shows a 1.5%p gain over the vanilla model; overall, our model is 3.27%p higher than the vanilla model. In this study, we confirm that each component of LogitMix is useful and effective, but combining all components best synergizes the model training.

Name	Loss	Accuracy(%)
Vanilla	$\mathcal{L}_{cls}$	78.32 ± 0.07
Mixup	$\mathcal{L}_{mix}$	79.82 ± 0.08
LogitMix$_m$	$\mathcal{L}_{cls} + \mathcal{L}_{sim} + \mathcal{L}_{mix}$	$\mathbf{81.59 \pm 0.09}$
$(-)$ Mixup	$\mathcal{L}_{cls} + \mathcal{L}_{sim}$	80.11 ± 0.09
$(-)$ Cross entropy	$\mathcal{L}_{mix} + \mathcal{L}_{sim}$	80.51 ± 0.08
$(-)$ Similarity loss	$\mathcal{L}_{mix} + \mathcal{L}_{cls}$	80.08 ± 0.49

Table 1: Accuracy of CIFAR-100 under various combinations of losses. We achieved consistently better accuracy by adding the proposed loss $\mathcal{L}_{sim}$ on existing losses. To analyze the effect of each loss term, we subtract each loss during training and observe its performance. The combination of all three losses reports the best accuracy.

5 Experiments

In this section, we evaluate LogitMix compared to competitors in various tasks. We stress that LogitMix can be combined with any mixing-based data augmentation techniques [Verma *et al.*, 2019; Yun *et al.*, 2019; Zhang *et al.*, 2018; Kim *et al.*, 2020] and enjoy their performance gains. We first assess the ability to improve prediction accuracy and confidence calibration. The performance gain in both aspects implies that the trained model better predicts the true posterior distribution; thus it better reveals the inter-class correlations of the data distribution. To show the effectiveness of LogitMix on various tasks, we tackle two problems; image classification and natural language processing (NLP) tasks. Additionally, we examine the robustness of the model trained with LogitMix. This result can quantify how LogitMix is successful in learning the vicinity of boundaries. We also conduct an ablation study by changing α for the empirical understanding of LogitMix.

Model architecture. For image classification tasks, we select five CNN architectures as backbone networks: three of them are conventional CNNs (i.e., VGGNet [Simonyan and Zisserman, 2015], ResNet [He *et al.*, 2016], and ResNeXt [Xie *et al.*, 2017]), and the others are light-weight CNNs (i.e., MobileNetV2 [Sandler *et al.*, 2018] and ShuffleNet [Ma *et al.*, 2018]). For the NLP tasks, we choose two baselines, BERT$_{BASE}$ and BERT$_{LARGE}$ [Devlin *et al.*, 2018] because the Mixup-Transformer [Sun *et al.*, 2020] demonstrated the effectiveness of Mixup on these pre-trained models. Inspired by their success, we use the same backbone models for building and evaluating LogitMix.

Datasets. We validate the effectiveness of LogitMix on three benchmark image datasets and eight natural language datasets. The image datasets include CIFAR100 [Krizhevsky and Hinton, 2009] (32×32 RGB images in 100 classes), TinyImageNet

Dataset	Network	Metric	Vanilla	Mixup	LogitMix$_m$	CutMix	LogitMix$_c$	PuzzleMix	LogitMix$_p$
CIFAR100	VGG16	Acc	74.30	75.02	76.22 (+1.20)	75.34	76.10 (+0.76)	75.92	**76.38** (+0.46)
		ECE	0.176	0.060	**0.035** (-0.025)	0.051	0.062 (+0.011)	0.121	0.100 (-0.021)
		OE	0.154	0.035	0.025 (-0.010)	0.022	**0.008** (-0.014)	0.011	0.049 (+0.038)
	ResNet50	Acc	78.32	79.82	81.59 (+1.77)	80.57	81.02 (+0.45)	82.57	**83.76** (+1.19)
		ECE	0.087	0.040	**0.014** (-0.026)	0.078	0.073 (-0.005)	0.092	0.215 (+0.123)
		OE	0.073	0.028	0.003 (-0.025)	0.064	0.060 (-0.004)	0.015	**0.000** (-0.015)
	ResNeXt50	Acc	79.18	81.10	81.63 (+0.53)	81.16	81.46 (+0.30)	81.40	**82.13** (+0.73)
		ECE	0.069	0.042	**0.021** (-0.021)	0.059	0.032 (-0.027)	0.092	0.220 (+0.128)
		OE	0.057	0.001	**0.000** (-0.001)	0.047	0.023 (-0.024)	0.017	0.001 (-0.016)
	MobileNetV2	Acc	69.69	69.98	73.90 (+3.92)	68.82	69.91 (+1.09)	75.77	**75.99** (+0.22)
		ECE	0.061	0.091	**0.048** (-0.043)	0.050	0.049 (-0.001)	0.097	0.100 (+0.003)
		OE	0.042	**0.000**	**0.000** (0.000)	**0.000**	**0.000** (0.000)	0.022	0.009 (-0.013)
	ShuffleNetV2	Acc	72.17	74.17	75.53 (+1.36)	73.60	73.73 (+0.13)	76.18	**76.75** (+0.57)
		ECE	0.079	0.060	0.042 (-0.018)	**0.016**	0.023 (+0.007)	0.126	0.094 (-0.032)
		OE	0.060	**0.000**	**0.000** (-0.000)	0.002	**0.000** (-0.002)	0.014	0.001 (-0.013)
TinyImageNet	ResNet50	Acc	66.6	68.34	**70.71** (+2.37)	69.08	69.87 (+0.79)	69.71	70.15 (+0.44)
		ECE	0.098	0.032	0.030 (-0.002)	**0.029**	0.034 (+0.005)	0.121	0.131 (+0.010)
		OE	0.076	0.022	0.010 (-0.012)	0.015	**0.005** (-0.010)	0.012	0.012 (0.000)
	MobileNetV2	Acc	57.62	59.55	62.12 (+2.57)	53.54	57.66 (+4.12)	64.08	**65.30** (+1.22)
		ECE	0.073	0.091	**0.032** (-0.059)	0.094	0.082 (-0.012)	0.112	0.104 (-0.008)
		OE	0.045	0.019	**0.000** (-0.019)	**0.000**	**0.000** (0.000)	0.034	0.016 (-0.018)
ILSVRC2015	ResNet50	Acc	76.13	77.37	78.38 (+1.01)	78.43	**78.51** (+0.08)	75.63	77.47 (+1.84)
		ECE	0.370	0.041	0.028 (-0.013)	0.028	**0.020** (-0.008)	0.120	0.117 (-0.003)
		OE	0.030	0.003	**0.001** (-0.002)	0.029	0.029 (0.000)	0.053	0.056 (+0.003)

Table 2: Comparison Acc, ECE, OE on CIFAR100, TinyImageNet and ILSVRC2015. LogitMix$_m$, LogitMix$_c$, LogitMix$_p$ represents the combination of $\mathcal{L}_{cls}$, $\mathcal{L}_{sim}$ losses with Mixup, CutMix, or PuzzleMix, respectively. The best score among different mixing methods is in **bold**. We train the model for PuzzleMix on ILSVRC2015 with *fast* setting that is provided by the official source code of PuzzleMix.

(64×64 RGB images in 100 classes) and ILSVRC2015 [Russakovsky *et al.*, 2015] (256×256 RGB images in 1000 classes). Additionally, the General Language Understanding Evaluation (GLUE) benchmark [Wang *et al.*, 2018].

Implementation details. All CNN networks are trained by SGD optimization with momentum of 0.9. Except for the PuzzleMix [Kim *et al.*, 2020] and LogitMix$_p$, we follow the same training schedule as CutMix [Yun *et al.*, 2019] for CIFAR100, TinyImageNet and ILSVRC datasets. Since light-weight models have a different training practice, we follow the procedure described in their original papers. In the language-based tasks, we use a pre-trained language model, *i.e.* BERT [Devlin *et al.*, 2018]. As Mixup-Transformer [Sun *et al.*, 2020] finetunes BERT using a Mixup training strategy for downstream tasks, we also follow the same practice using the same hyperparameters as [Sun *et al.*, 2020]. When finetuning BERT$_{BASE}$ (or BERT$_{LARGE}$), the batch size is 8, the learning rate is $2e-5$, the max sequence length is 128, and the number of the training epochs is 3 for all eight tasks. We use a beta distribution with $\alpha = 3.0$ for λ. To train the models on all the datasets except for ILSVRC2015, we use a single Titan XP GPU with 12 GB memory. For ILSVRC2015, we utilize four V100 GPU.

Metrics for confidence calibration. For the quantitative analysis of the confidence calibration, we used two popular metrics, the expected calibration error (ECE) [Naeini *et al.*, 2015] and the overconfidence error (OE) [Thulasidasan *et al.*, 2019]. Note that the two measures are calculated on validation datasets. Specifically, the ECE represents an average difference between true confidence and predicted confidence. If it is zero, it means the network is perfectly calibrated. The OE is similar to ECE, but it only measures the confidence difference when it is over-confident. Over-confidence is a critical factor in high-risk systems. Thus, the OE is a good indicator to assess system reliability for high-risk applications.

5.1 Image Classification Tasks

We showed that our method improved classification accuracy (*i.e.* high confidence on training samples) and helped to learn the gradual transition of the probability between the two classes (*i.e.*, appropriate confidence on samples from off-the-data distribution) with a toy example. In this section, we evaluate the proposed method on image classification tasks for handling more realistic application scenarios.

Table 2 reports the experimental results using various CNN networks and image benchmark datasets. In general, we achieve better accuracy and confidence calibration after combining LogitMix. In particular, our gain in prediction accuracy is substantial where the gap between the baseline and the baseline combined with LogitMix is as large as the gap between the vanilla and the other competitors. Besides, our method enjoys the advantage of the baseline and effectively improves the baseline performances.

When LogitMix$_p$ is used to train ResNet50 or ResNeXt50 on CIFAR100, we observe large gains in the prediction accuracy but substantial errors in ECE. Despite the trade-off between the prediction accuracy and the calibration error,

Model	MNLI-mm	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	Average
BERT$_{BASE}$ [Devlin *et al.*, 2018]	**84.73**	**91.25**	91.43	**93.12**	57.82	89.43	**87.75**	68.95	83.06
Mixup on BERT$_{BASE}$ [Sun *et al.*, 2020]	84.29	91.15	91.36	**93.12**	58.82	89.44	87.50	67.87	82.94
LogitMix on BERT$_{BASE}$	**84.73** (0.00)	**91.25** (0.00)	**91.58** (+0.15)	**93.12** (0.00)	**58.85** (+0.03)	**89.45** (+0.01)	87.50 (-0.25)	**70.04** (+1.09)	**83.32** (+0.26)
BERT$_{LARGE}$ [Devlin *et al.*, 2018]	85.99	90.20	92.20	92.89	60.88	89.9	87.75	73.29	84.14
Mixup on BERT$_{LARGE}$ [Sun *et al.*, 2020]	86.02	90.09	92.42	92.66	61.86	**90.02**	88.24	73.29	84.33
LogitMix on BERT$_{LARGE}$	**86.10** (+0.08)	**90.95** (+0.75)	**92.60** (+0.18)	**93.58** (+0.69)	**63.89** (+2.03)	89.98 (-0.04)	**89.22** (+0.98)	**74.37** (+1.08)	**85.09** (+0.76)

Table 3: Performance comparison on NLP tasks (GLUE tasks). Matthew's correlations are reported for CoLA. Spearman correlations are reported for STS-B. The accuracies are reported for the other tasks. We compare BERT and Mixup on BERT with LogitMix on BERT. The gain over the best score (*i.e.* the maximum value of two competitors) is reported in parentheses, and the best score per task is highlighted in **bold**.

	VWCI [Seo *et al.*, 2019]	Mixup [Zhang *et al.*, 2018]	LogitMix$_m$ (Ours)
Acc	73.87 (+0.09)	75.02 (+0.72)	76.22 (+1.92)
ECE	0.098 (-0.089)	0.057 (-0.116)	0.035 (-0.141)

Table 4: Comparison with other calibration methods.

LogitMix$_p$ achieves nearly 0 in OE, indicating that our training strategy is a safe and reliable solution for practical applications. On ILSVRC2015, LogitMix consistently achieves meaningful gains over the baseline mixing-based methods; finding the best ratio of the three-loss terms (currently, they are equally weighted) would further improve the performance. Without weight tuning, nevertheless, LogitMix$_c$ still achieves state-of-the-art performances on ILSVRC2015 evaluations. Considering that achieving the improvement over the state-of-the-art performances is particularly challenging, our gains on PuzzleMix (CIFAR100 and TinyImageNet) or CutMix (ILSVRC2015) are sufficiently meaningful. As a result, LogitMix surpasses the performance of existing methods in most experimental conditions.

One interesting remark can be made via the experiment with a compact model such as MobileNetV2. Generally speaking, Mixup-like approaches act as an augmentation method, which populates training examples to prevent over-fitting. However, if it injects examples far from the training distribution (*e.g.*, missing the salient object in mixed data), such augmentation can induce under-fitting. Under-fitting normally does not degrade the performance of high-capacity networks, but it can hurt the performance of low-capacity networks. When CutMix is used for training MobileNetV2 on TinyImageNet, we observe severe performance degradation. We interpret that their mixed data rather provides misguiding learning signals and thus induces under-fitting. Unlike CutMix, PuzzleMix largely outperforms Mixup and CutMix on MobileNetV2 because PuzzleMix carefully considers the saliency when mixing the data, mitigating the misleading mixed data.

When LogitMix is combined with CutMix, LogitMix$_c$, the effect of under-fitting is substantially reduced, filling the degradation gap introduced by CutMix. From this result, we argue that our method provides a reasonable interpretation for understanding the example far from training distribution. LogitMix helps the model to understand the hidden relationships with weak supervision, and it can create synergy when combined with the existing mixing-based technique. For the same reason, LogitMix$_p$ shows clear improvements over PuzzleMix. This observation is coherent with our motivation that the inter-class correlation (*i.e.* mixed logit) with sufficient class information (*i.e.* label) helps to improve both the accuracy and the estimate of predictive confidence.

5.2 Natural Language Processing Tasks

Table 3 summarizes the performances of the vanilla model (baseline BERT), Mixup (Mixup-Transformer), and LogitMix under eight different NLP tasks. In this experiment, we only focus on evaluating the prediction performances because not all metrics handle the accuracy (*i.e.*, Matthew's correlation for CoLA and Spearman correlations for STS-B). Besides, CutMix and PuzzleMix are excluded as the competitors because it is non-trivial to create mixing data for text by following their region-based mixing principle.

In this experiment, we observe that Mixup on BERT achieves -0.12 %p and +0.18 %p of average differences over BERT$_{BASE}$ and BERT$_{LARGE}$, respectively. Adopting LogitMix on the baseline BERT, we mostly achieve performance gains over not only the baseline BERT but also Mixup on BERT. Overall, LogitMix on BERT further promotes the performances of baseline models on eight NLP tasks, approximately 0.26%p and 0.95%p of average gains over BERT$_{BASE}$ and BERT$_{LARGE}$, respectively, and 0.37%p and 0.76%p of average gains over Mixup on BERT$_{BASE}$ and Mixup on BERT$_{LARGE}$, respectively.

One interesting remark is that the performance gains of LogitMix on BERT$_{LARGE}$ are more pronounced than those of LogitMix on BERT$_{BASE}$. This is an encouraging result because improving the complex model is more challenging and truly improves state-of-the-art performances. Since LogitMix provides a useful learning signal, we speculate that the large model could benefit more. These experimental results demonstrate that LogitMix is a powerful training strategy and it can easily be used for various tasks.

5.3 Confidence Calibration

We focus on evaluating our calibration performances. As our method calibrates the prediction with a single inference, we exclude Bayesian approaches [MacKay, 1992] or stochastic approaches with multiple inferences [Lakshminarayanan *et al.*, 2017] for comparisons. Instead, we identify Mixup and

Method	Metric	$\alpha = 0.2$	$\alpha = 0.4$	$\alpha = 0.6$	$\alpha = 1.0$	$\alpha = 2.0$	$\alpha = 3.0$	$\alpha = 4.0$
Mixup	Acc	79.30	79.32	79.28	79.07	78.15	77.92	76.81
	ECE	0.020	0.040	0.400	0.056	0.080	0.099	0.120
	OE	0.012	0.008	0.001	0.000	0.000	0.000	0.000
LogitMix$_m$ $(-)$ $\mathcal{L}_{cls}$	Acc	80.21	79.74	79.49	79.66	79.98	79.76	80.21
	ECE	0.026	0.024	0.087	0.071	0.101	0.110	0.101
	OE	0.018	0.002	0.000	0.000	0.000	0.000	0.000
LogitMix$_m$	Acc	80.47	81.10	80.72	80.34	81.23	81.59	81.31
	ECE	0.039	0.025	0.019	0.013	0.013	0.014	0.014
	OE	0.029	0.014	0.011	0.005	0.003	0.002	0.002

Table 5: Ablation study for the effect of α. Unlike Mixup, LogitMix$_m$ can handle mixed data with a large α value successfully; the high accuracy and the small OE. Besides, LogitMix$_m$ is less sensitive to the choice of α. The large α tends to degrade ECE because the out-of-distribution samples by the large α strictly penalize over-confident, leading to the under-confident model.

variance-weighted confidence-integrated Loss (VWCI) [Seo *et al.*, 2019] as a single inference-based competitor. In order to match the same recipe in the results reported by VWCI, we compare the methods for CIFAR100 using VGG16 (Table 4). The result of VWCI is obtained from the paper, and both the accuracy and ECE with the gains over the baseline model (in parentheses) are summarized because the reported performance from VWCI is slightly different from our results. Compared with the others, LogitMix achieves the highest accuracy and the lowest calibration error; comparing with performance gains, our result is also the best among the three.

5.4 Searching for the Loss Weights

We search for the best weight to some extent. Currently, w_{mix} and w_{cls} are set to 1 for reducing the searching cost. Then, the best weight for $\mathcal{L}_{sim}$ is determined as one of $\{0.1, 0.5, 1.0, 2.0\}$ by testing them on CIFAR100. As a result, we set w_{sim} to 1 for LogitMix$_m$, 0.1 for LogitMix$_c$ and 0.5 for LogitMix$_p$ to perform the best. Then, the same weights were used regardless of the datasets. The different weights may be caused by different regularization effects per mixing-based augmentation. Although the weights might not be optimal for each dataset, LogitMix consistently improves the baselines.

5.5 The Effect of Alpha

In all mixing-based approaches, the mixing rate is controlled by α. In particular, using a large α value degrades the accuracy largely in Mixup [Thulasidasan *et al.*, 2019; Zhang *et al.*, 2018]. With the large α, the training sample is likely to be mixed in half (*e.g.*, near boundaries), far from the training distribution. Using large α also leads the network to rarely observe the original sample. Then, it raises a natural question: *is a large α also bad for LogitMix?*

To analyze the effect of the α, the performance of the network (*i.e.*, accuracy and ECE) is investigated by changing α using ResNet50 on CIFAR100 as reported in Table 5. As expected, the accuracy starts to drop with high α in Mixup. Meanwhile, the accuracy generally increases as the α value increases in our method. Here, we discard the effect of $\mathcal{L}_{cls}$ from LogitMix and observe that LogitMix without $\mathcal{L}_{cls}$ shows a similar tendency. This implies that our similarity loss can provide a meaningful interpretation by implicit supervision. LogitMix can successfully handle half-mixing samples (*i.e.* samples created with a large α value). This phenomenon is coherent with the advantage of LogitMix on the low-capacity network (*e.g.* MobileNetV2 on TinyImageNet); weak supervision provides a reasonable interpretation for understanding the out-of-distribution samples, and it is effective to handle under-fitting as well. Nevertheless, extreme α values are also harmful to LogitMix because they prevent the original training samples (*i.e.* in-distribution samples) from participating in model training. Based on this experimental result, we conclude $\alpha = 3$ as the default value.

Regarding the confidence calibration, ECE is degraded, and OE decreases as α increases. This means that ECE increases due to being under-confident; the network is overly regularized with out-of-the-distribution samples. This is consistent with the tendency for accuracy. Unlike Mixup, LogitMix generally achieves good calibration results, less sensitive to α.

6 Conclusion

This study is motivated by the human reasoning capability that improves the prediction by better understanding the wrong choices. Based on this analogy, we proposed LogitMix, a novel joint training strategy that utilizes the mixed logit for considering the class relationships and the original training data for class-only information.

Based on extensive evaluations using various network architectures on both image and language classification tasks, empirical analysis, and ablation study, we have demonstrated various useful properties of LogitMix; 1) creating performance synergy with any mixing-based techniques, and thus achieving state-of-the-art performances, 2) applicable to various backbones, datasets, and tasks, 3) improving the decision boundaries without losing the confidence accuracy on training data, and 4) being insensitive to α.

Acknowledgements

This research was partially supported by the Basic Science Research Program through NRF funded by the MSIP (NRF-2022R1A2C3011154), IITP grant funded by the Korea government(MSIT) and KEIT grant funded by the Korea government(MOTIE) (No. 2022-0-00680), and Korea Medical Device Development Fund grant funded by the Korea government (Project Number: 202011D06).

References

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Goodfellow *et al.*, 2015] Ian Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. In *ICLR*, 2015.

[Guo *et al.*, 2017] Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2017.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[Heo *et al.*, 2019] Byeongho Heo, Jeesoo Kim, Sangdoo Yun, Hyojin Park, Nojun Kwak, and Jin Young Choi. A comprehensive overhaul of feature distillation. In *ICCV*, pages 1921–1930, 2019.

[Hinton *et al.*, 2014] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the Knowledge in a Neural Network. In *NIPS 2014 Deep Learning Workshop*, 2014.

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

[Krizhevsky and Hinton, 2009] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Master's thesis, Department of Computer Science, University of Toronto*, 2009.

[Lakshminarayanan *et al.*, 2017] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *NeurIPS*, pages 6402–6413, 2017.

[Li and Maki, 2018] Vladimir Li and Atsuto Maki. Feature contraction: New convnet regularization in image classification. In *BMVC*, page 213, 2018.

[Ma *et al.*, 2018] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *ECCV*, pages 116–131, 2018.

[MacKay, 1992] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[Mikolov *et al.*, 2013] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NeurIPS*, pages 3111–3119, 2013.

[Müller *et al.*, 2019] Rafael Müller, Simon Kornblith, and Geoffrey Hinton. When does label smoothing help? *arXiv preprint arXiv:1906.02629*, 2019.

[Naeini *et al.*, 2015] Mahdi Pakdaman Naeini, Gregory Cooper, and Milos Hauskrecht. Obtaining well calibrated probabilities using bayesian binning. In *AAAI Conference on Artificial Intelligence*, 2015.

[Pereyra *et al.*, 2017] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton. Regularizing neural networks by penalizing confident output distributions. In *ICLR*, 2017.

[Russakovsky *et al.*, 2015] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 115(3):211–252, 2015.

[Sandler *et al.*, 2018] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pages 4510–4520, 2018.

[Seo *et al.*, 2019] Seonguk Seo, Paul Hongsuck Seo, and Bohyung Han. Learning for single-shot confidence calibration in deep neural networks through stochastic inferences. In *CVPR*, pages 9030–9038, 2019.

[Simonyan and Zisserman, 2015] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[Sun *et al.*, 2020] Lichao Sun, Congying Xia, Wenpeng Yin, Tingting Liang, Philip S Yu, and Lifang He. Mixup-transformer: Dynamic data augmentation for nlp tasks. *arXiv preprint arXiv:2010.02394*, 2020.

[Szegedy *et al.*, 2016] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, pages 2818–2826, 2016.

[Thulasidasan *et al.*, 2019] Sunil Thulasidasan, Gopinath Chennupati, Jeff Bilmes, Tanmoy Bhattacharya, and Sarah Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. In *NeurIPS*, 2019.

[Verma *et al.*, 2019] Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *ICML*, pages 6438–6447. PMLR, 2019.

[Wang *et al.*, 2018] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. arXiv preprint 1804.07461, 2018.

[Xie *et al.*, 2017] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017.

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

[Zhang *et al.*, 2018] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018.