Catch-Up Mix: Catch-Up Class for Struggling Filters in CNN

Minsoo Kang[*1,2], **Minkoo Kang**[*1], **Suhyun Kim**[†1]

[1]Korea Institute of Science and Technology, Republic of Korea
[2]Korea University, Republic of Korea
{minsoo.kang0918, mit240083, dr.suhyun.kim}@gmail.com

Abstract

Deep learning has made significant advances in computer vision, particularly in image classification tasks. Despite their high accuracy on training data, deep learning models often face challenges related to complexity and overfitting. One notable concern is that the model often relies heavily on a limited subset of filters for making predictions. This dependency can result in compromised generalization and an increased vulnerability to minor variations. While regularization techniques like weight decay, dropout, and data augmentation are commonly used to address this issue, they may not directly tackle the reliance on specific filters. Our observations reveal that the heavy reliance problem gets severe when slow-learning filters are deprived of learning opportunities due to fast-learning filters. Drawing inspiration from image augmentation research that combats over-reliance on specific image regions by removing and replacing parts of images, our idea is to mitigate the problem of over-reliance on strong filters by substituting highly activated features. To this end, we present a novel method called Catch-up Mix, which provides learning opportunities to a wide range of filters during training, focusing on filters that may lag behind. By mixing activation maps with relatively lower norms, Catch-up Mix promotes the development of more diverse representations and reduces reliance on a small subset of filters. Experimental results demonstrate the superiority of our method in various vision classification datasets, providing enhanced robustness.

1 Introduction

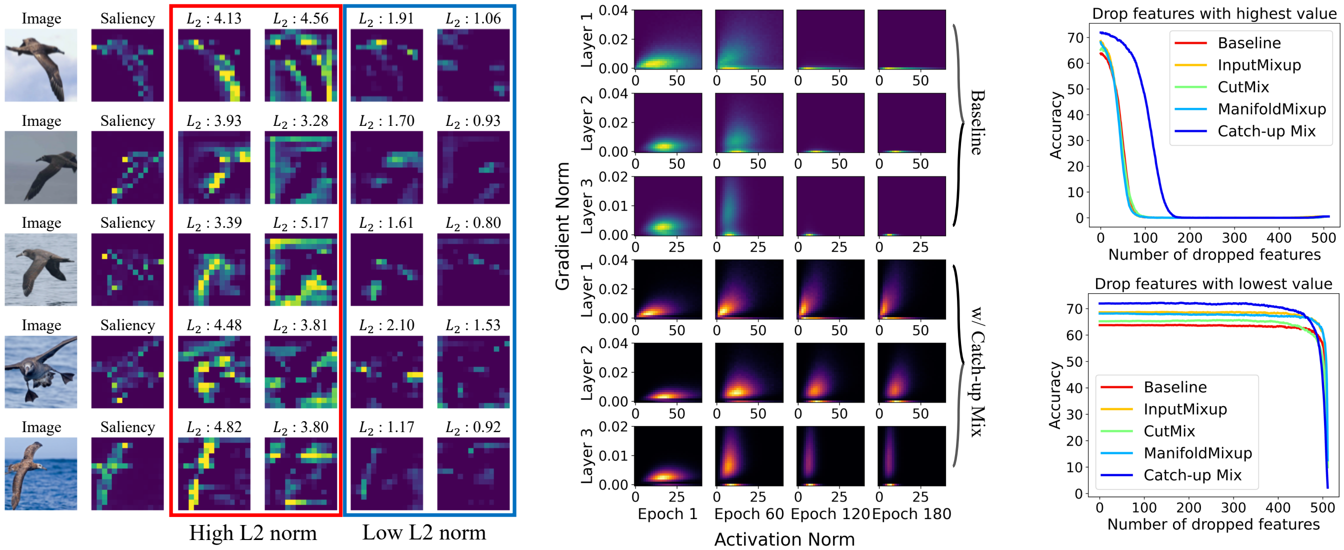
There have been remarkable improvements in deep learning algorithms, especially in computer vision fields. Since the ImageNet challenge (Krizhevsky, Sutskever, and Hinton 2012), modern image classifiers have achieved near-human level accuracy in the image recognition task. Although models may achieve high accuracy, they often face challenges related to their complexity and overfitting. Even slight variations or perturbations in input images, such as rotation, rescaling (Goodfellow, Bengio, and Courville 2016), corruption (Hendrycks and Dietterich 2019), or adversarial attacks (Szegedy et al. 2013; Goodfellow, Shlens, and Szegedy 2014), can lead to unexpected model behavior. For instance, a classifier trained on clean images from a sunny day may underperform with inputs from rainy conditions. Such vulnerabilities underscore the importance of robustness, especially when models tackle real-world scenarios subject to distributional shifts (Hendrycks and Dietterich 2019).

There are several factors that can contribute to a model's limited generalization and robustness. One particular phenomenon we focus on is the convolutional neural network (CNN) model's heavy reliance on a small subset of convolutional filters for making predictions. Models relying on a limited number of filters often exhibit poor performance on test data and lack robustness against slight variations. Furthermore, a concerning aspect is that rapidly trained weights can potentially make inaccurate judgments due to their tendency to learn dataset-specific biases (Nam et al. 2020). These biases may capture object-irrelevant patterns (e.g., background) rather than focusing on the object of interest.

To address this issue, regularization methods such as weight decay, dropout, and data augmentation are widely employed. Dropout (Srivastava et al. 2014) tempers the model's tendency to over-rely on particular units by randomly deactivating them. Weight decay (Ng 2004), known as ℓ_2 regularization, encourages the model to have smaller weights by adding a penalty for large weights. Data augmentation (DeVries and Taylor 2017; Hinton et al. 2012; Simonyan and Zisserman 2014; Kang and Kim 2023) encourages learning more robust and generalized representations by generating training data variants through assorted transformations. However, those regularization methods do not directly address the model's reliance on specific filters.

Our observations indicate that the heavy reliance on a small subset of filters in CNN can occur when slow-learning filters lack adequate opportunities to learn, especially when a small number of the fast-learning filters are sufficient to classify the training dataset accurately. It is supported by the visualization of activation maps (Figure 1a) and histograms, which illustrate the relation between the ℓ_2 norms of activation maps and corresponding gradients (Figure 1b). Figure 1a provides insights into filters' learning progress and impact during training. We notice that fast-learning filters tend to exhibit higher ℓ_2 norms of activations, often capturing biases like the background. On the other hand, lower-magnitude filters struggle to extract meaningful information from the input.

[*]These authors contributed equally.
[†]Corresponding Author.



(a) Images, saliencies, and activation maps with high L2 norm (red) and low L2 norm (blue)

(b) Changes in gradient norm and activation norm distribution across layers during training

(c) A number of dropped features - Acc (%, ↑) plot.

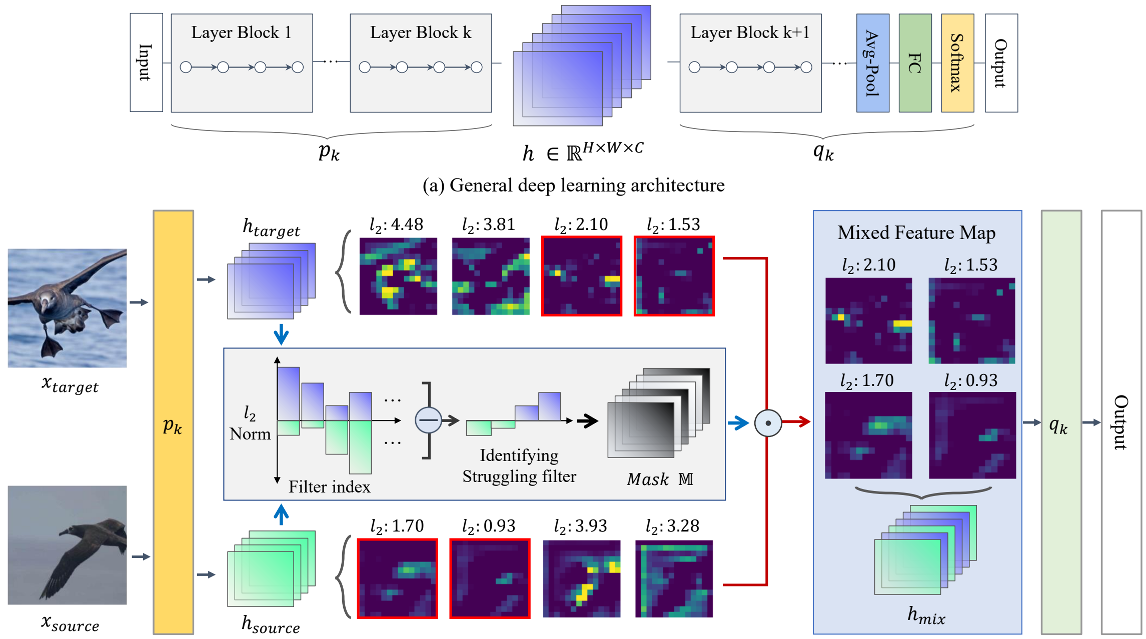
Figure 1: We visualize activation maps, ℓ_2 norm, and gradients to understand how filters behave during training. (a) shows visualizations of images, saliency maps, and activation maps. Activation maps are obtained from the third layer block output of ResNet-18 while training CUB-200 at 120 epochs. (b) represents how distributions of activation norm (x-axis) and gradient norm (y-axis) change over epochs. For example, among rows of 'Baseline,' the gradient scale decreases significantly after 60 epochs, making it difficult to update the weight properly for the remaining process. (c) depicts the model's accuracy as latent vectors are sequentially dropped by their value. This highlights that our method prompts the model to use diverse features.

The problem is that slow-learning filters not only lag behind in their learning progress but are also deprived of future learning opportunities in the remaining epochs. The main reason is that the current model already reaches a top-1 training accuracy of 95.81% at 120 epochs. Thus, despite many filters failing to extract features properly at 120 epochs (Figure 1a), only small gradients will be generated, indicating minimal updates to these filters (Figure 1b). This imbalance causes poor generalization performance, as evidenced by the top-1 validation accuracy rates of 48.21%, 50.00%, and 51.50% achieved at 60, 120, and 150 epochs, respectively. As a result, unfortunately, a significant portion of the training process becomes unproductive for these slow learners, hindering the model from acquiring generalized representations.

Throughout this analysis, we observe that slow-learning filters tend to lag behind during the training and have limited opportunities for the model to improve for the rest of the training process. Our design is inspired by image augmentation techniques which train models on defective images by removing (CutOut (DeVries and Taylor 2017)) or replacing (CutMix (Yun et al. 2019)) parts of the image to learn different aspects of the data without relying on any particular region. Transposing this philosophy to the feature space, we intentionally exclude and supplant well-trained filters from a stochastic layer. By training on these defective feature maps, we encourage the model to learn different features of data utilizing the '*struggling filters*' without relying solely on a small number of strong filters. We denote this concept as a '*catch-up class*,' allowing slow-learners to be trained and contribute to the model's decision-making process.

In this paper, we present Catch-up Mix, a novel feature-level mixup method designed to offer learning opportunities to a wide range of filters during training, focusing on filters that may fall behind. Catch-up Mix ensures broader training exposure for less-evolved filters by mixing activation maps of pairs with relatively low ℓ_2 norms and updating gradients by excluding filters with high magnitudes. Note that choosing a 'relative' smaller norm doesn't mean that every selected filter is under-trained; instead, it provides a variety of filter combinations to mitigate over-reliance issues. Consequently, the trained model makes robust predictions without relying heavily on a small subset of filters.

The experiments show that our method outperforms other methods on various vision classification datasets. Furthermore, we designed Catch-up Mix to utilize a greater number of filters to identify and capture the characteristics and patterns in the input data, leading us to expect high levels of robustness and feature diversity. To validate this, we assessed its robustness through various means, such as adversarial attacks, deformation, and data corruption. Since our main concern, the problem of struggling filters, is intrinsically tied to the diversity of the training dataset and the capacity of the model, we assess outcomes both with limited datasets and large-scale datasets. Moreover, the results from out-of-distribution detection and loss landscape visualization further emphasize its robustness and generalization capabilities.



(a) General deep learning architecture

(b) Catch-up Mix procedure

Figure 2: The overall framework and augmentation process of Catch-up Mix. (a) represents a general deep learning architecture, and (b) outlines the procedure of Catch-up Mix. First, we compare the magnitudes of activation maps using their ℓ_2 norms. A mask $\mathbb{M}$ is generated to mix the activation maps from the selected pair with relatively low ℓ_2 norms.

2 Method

2.1 Preliminaries

Mixup augmentation is performed on a pair of input images or feature maps of the network to enhance generalization capabilities. A model with input-level mixup can be expressed as $f(mix_\lambda(x, x'))$, where $x, x' \in X$ represents paired images from the input batch. $f : X \to \mathbb{R}^N$ is a given network where N is the number of classes, and λ is a mixing ratio. Usually, λ is randomly sampled from $Beta(\alpha, \alpha)$ distribution, where α is a hyper-parameter. For example, Input-Mixup (Zhang et al. 2018) can be expressed as follows:

$$mix_\lambda(x, x') = \lambda \cdot x + (1 - \lambda) \cdot x'. \quad (1)$$

To formalize the network with feature-level mixup, we define two functions as illustrated in Figure 2: $p_k : X \to \mathbb{R}^{C \times H \times W}$ and $q_k : \mathbb{R}^{C \times H \times W} \to \mathbb{R}^N$, that are mapping from input image to the feature map of layer k and from feature map of layer k to model output. Throughout this paper, we denote $h_k \in \mathbb{R}^{C \times H \times W}$ as the feature map of layer k and $h_k^i \in \mathbb{R}^{H \times W}$ as the activation map of ith filter in layer k. Thus, p_k and q_k satisfy $f(x) = q_k(p_k(x))$ and $h_k = p_k(x)$. Those mixup methods $f_\lambda(x, x')$ can be expressed as follows:

$$\begin{cases} f(mix_\lambda(x, x')), & \text{Input-level mixup,} \\ q_k(mix_\lambda(p_k(x), p_k(x'))), & \text{Feature-level mixup.} \end{cases} \quad (2)$$

In the image classification task, each image sample $x \in X$ has a one-hot encoded label vector $y \in \{0, 1\}^N$ where N is the number of classes. Labels should also be interpolated when a mixup method is used. The following equation is the mixed label determined by ground truth labels weighted by the mixing ratio λ:

$$y_{mix} = \lambda \cdot y + (1 - \lambda) \cdot y'. \quad (3)$$

2.2 Proposed Method

In this section, we introduce Catch-up Mix, a method designed to address the over-reliance on small subsets of filters and enhance the model's generalization capability. Our observation is that activations with low ℓ_2 norms tend to be relatively underdeveloped during the training process (Figure 1a). Also, if a model with less developed filters can still achieve high training accuracy, there will be a scarcity of gradients (Figure 1b). This indicates that less-developed filters may not have sufficient learning opportunities in the subsequent epochs and not reach their full potential.

Based on this observation, our approach randomly selects a layer and mixes activation maps that exhibit relatively less development for a given input. By making uncertain predictions with low confidence based on these mixed features, we can update the corresponding filters that do not capture object characteristics well, thereby offering them more substantial training opportunities. Catch-up Mix enables the model to prioritize the improvement of less developed filters and mitigates its reliance on a few highly influential filters. As a result, Catch-up Mix enhances the model's generalization ability and reduces its vulnerability to overfitting.

For each iteration of Catch-up Mix, a random layer k is selected, and the batch input is passed through the model up to that k-th layer to obtain the source and target feature maps, denoted as h_k and h'_k. For a given input, the influence of each filter on predictions is assessed by computing the ℓ_2 norm of its activations h_k^c, denoted as Filter Influence (FI),

$$FI = \{FI_c | FI_c = ||h_k^c||_2, \quad \text{for all } c \in \mathcal{C}\}. \tag{4}$$

Next, we compare the sum-to-one normalized FIs of source and target mixup pairs, and the activation maps with the relatively lower filter influence (Relative Filter Influence; RFI) are selected. This normalization is necessary because the scale of the activation map, and thus the scale of the FIs, can vary significantly depending on the input. If RFI_i is high, i-th filter has a stronger influence on predicting the source image compared to the target image, and vice versa.

$$RFI = \{RFI_c | RFI_c = \frac{FI_c}{\sum_{c=0}^{\mathcal{C}}(FI_c)} - \frac{FI'_c}{\sum_{c=0}^{\mathcal{C}}(FI'_c)}, \text{ for all } c \in \mathcal{C}\}. \tag{5}$$

Using RFI and mixing ratio λ, we generate a binary mixup mask that determines which filters' activation map should be propagated to the next layer. Given the number of filters in the layer and λ, we calculate the number of filters to retain from source feature map $N_{mix} = \lfloor \lambda \times |\mathcal{C}| \rfloor$. Next, we generate filter-wise activation mask $\mathbb{M}$ that drops the activation maps with top $|\mathcal{C}| - N_{mix}$ RFI values. The sum of the mask elements equals N_{mix}. Finally, the mixed feature map is computed as follows:

$$h_{mix} = \mathbb{M} \odot h + (\mathbb{I} - \mathbb{M}) \odot h', \tag{6}$$

where $\mathbb{I} \in \{1\}^C$ is a binary mask filled with ones, $\odot$ is a filter-wise multiplication operation. The output of network f_λ with Catch-up Mix at layer k can be expressed as follows:

$$f_\lambda(x, x') = q_k(\mathbb{M} \odot p_k(x) + (\mathbb{I} - \mathbb{M}) \odot p_k(x')). \tag{7}$$

For better understanding, we explain how Catch-up Mix operates using ResNet (He et al. 2016a), which consists of five-layer blocks: the first convolutional layer as the first layer block and four stages of ResNet as the other four layer blocks. Here, we uniformly sample the mixup layer k from the set $K = \{0, 1, 2, 3, 4, 5\}$ every iteration, where layer 0 corresponds to the input layer. If $k = 0$, CutMix (Yun et al. 2019) is applied, and if $k > 0$, Catch-up Mix is applied to feature maps of kth layer block. Layer blocks for applying Catch-up Mix can be set arbitrarily. For models with many layers or multi-stage, we can use Catch-up Mix, applying an appropriate mixup layer set. After, we assign a new label for the mixed feature map according to the mixing ratio λ.

3 Experiments

In this section, we extensively evaluate the performance and robustness of Catch-up Mix on various datasets. First, we compare the generalization performance of the model trained with proposed methods on general classification datasets: CIFAR-100 (Krizhevsky 2009), and Tiny-ImageNet (Chrabaszcz, Loshchilov, and Hutter 2017). In addition, we measure the augmentation overhead of mixup methods. Next, we evaluate the robustness of classifiers against adversarial attacks, data corruption, and deformation to validate the improvement achieved by Catch-up Mix.

Method	Top-1 Err (%)	FGSM Err (%)	Time/epoch (s, ↓)	mCE (%)
Baseline	22.98	84.93	103.8±0.5	51.43
InputMixup	19.26	75.72	104.4±0.5	44.84
CutMix	19.95	73.89	104.5±0.7	53.74
PuzzleMix	18.55	79.33	275.8±2.5	46.21
SaliencyMix	19.43	81.25	-	47.85
Co-Mixup	20.06	87.94	894.0±6.5	53.72
Manifold	19.32	81.96	107.1±0.4	<u>44.32</u>
MoEx	19.45	80.41	105.9±0.5	52.52
RecursiveMix	19.42	89.58	-	48.52
AlignMixup	<u>19.20</u>	<u>69.84</u>	118.1±0.5	44.71
Catch-up Mix	**17.76**	**68.16**	107.8±0.6	**43.76**

Table 1: Top-1 / FGSM error rate (%, ↓), and computational cost (s, ↓) of training PreActResNet-18 with mixup methods on CIFAR-100. Also, we evaluate the mean Corruption Error (mCE, %, ↓) rate on CIFAR-100-C using a pre-trained model on CIFAR-100 using corresponding mixup methods.

Furthermore, we validate that our performance improvements are not limited to specific datasets and architectures. We also evaluate our methods on standard fine-grained datasets: CUB-200-2011 (CUB) (Wah et al. 2011), Stanford-Cars (Cars) (Krause et al. 2013), and FGVC-Aircraft (Aircraft) (Maji et al. 2013). Note that achieving high performance on fine-grained datasets requires the model to have the capability of capturing the fine details in images. Since the '*struggling filter*' problem we want to address is related to the volume of the dataset, we compare and analyze the performance of training on small datasets and a large dataset, ImageNet (Krizhevsky, Sutskever, and Hinton 2012). Also, we evaluate the out-of-distribution detection performance using ImageNet-O (Hendrycks et al. 2021) to assess the generalization performance.

Throughout this paper, we present the results of experiments aimed at evaluating the effectiveness of Catch-up Mix across various architectures: ResNet (He et al. 2016a), PreActResNet (He et al. 2016b), DenseNet (Huang et al. 2017), PyramidNet (Han, Kim, and Kim 2017), VGGNet (Simonyan and Zisserman 2014), MobileNetv2 (Sandler et al. 2018), Wide-ResNet (Zagoruyko and Komodakis 2016), and ResNeXt (Xie et al. 2017). Lastly, we compare the loss landscape to visualize the regularization effect of Catch-up Mix.

3.1 Performance on General Classification Datasets

We evaluate the performance of our method on two benchmark datasets: CIFAR-100 and Tiny-ImageNet. We use PreActResNet-18 as the backbone architecture and reproduce existing mixing data-based augmentation methods: InputMixup (Zhang et al. 2018), CutMix (Yun et al. 2019), PuzzleMix (Kim, Choo, and Song 2020), SaliencyMix (Uddin et al. 2020), Co-Mixup (Kim et al. 2020), SnapMix (Huang, Wang, and Tao 2021), ManifoldMixup (Manifold) (Verma et al. 2019), MoEx (Li et al. 2021), AlignMixup (Venkataramanan et al. 2022), and Recur-

Method	General			Deformation Performance Accuracy (%, ↑)							
	Top-1 Err (%, ↓)	FGSM Err (%, ↓)	Time/epoch (s, ↓)	Random Rotate ±20°	Random Rotate ±40°	Random Sheering ±28.6°	Random Sheering ±57.2°	Zoom In/Out 60%	Zoom In/Out 80%	Zoom In/Out 120%	Zoom In/Out 140%
Baseline	36.98	89.34	21.2±0.1	53.96	42.99	54.14	37.62	20.41	47.51	50.67	40.76
InputMixup	35.32	91.02	21.3±0.2	53.20	42.45	54.94	39.44	17.88	47.14	52.54	43.43
CutMix	34.58	87.00	21.4±0.2	56.63	45.13	57.70	43.00	12.69	44.72	47.43	38.65
PuzzleMix	32.98	87.22	64.4±0.8	53.93	40.90	56.02	40.69	9.26	38.94	47.10	36.43
SaliencyMix	34.76	93.75	-	54.78	44.14	56.75	41.80	15.18	44.85	49.48	41.45
Co-Mixup	34.58	91.11	244.7±1.6	53.12	39.89	52.22	37.86	7.72	33.52	40.15	32.05
Manifold	35.46	88.74	21.8±0.2	54.37	43.65	55.78	39.42	15.51	46.32	49.78	39.12
MoEx	34.27	89.49	21.7±0.2	56.23	44.56	57.79	41.24	14.66	45.52	48.55	40.70
RecursiveMix	33.97	92.88	-	56.01	43.74	56.75	41.40	12.03	44.11	50.06	44.09
AlignMixup	34.03	86.56	35.4±0.3	54.87	43.62	57.25	41.38	12.36	46.07	53.55	42.57
Catch-up Mix	**30.73**	**84.22**	22.2±0.2	**60.20**	**47.87**	**61.52**	**44.06**	**24.03**	**55.46**	**58.34**	**48.32**

Table 2: Top-1 / FGSM error rate (%, ↓), and computational cost (s, ↓) of training the model with mixup methods on Tiny-ImageNet datasets using PreActResNet-18. Top-1 accuracy rates (%, ↑) on the test datasets of Tiny-ImageNet with various deformations. We evaluate the PreActResNet-18 model pre-trained on the original Tiny-ImageNet dataset.

siveMix (Yang et al. 2022). We train each model for 1200 (Tiny-ImageNet) and 2000 epochs (CIFAR-100) using hyperparameter settings from AlignMixup. For method-specific mixup parameters, we follow the paper and the official code. As shown in Table 1, Catch-up Mix achieves the best generalization performance compared to other mixup baselines. On Tiny-ImageNet, Catch-up Mix outperforms PuzzleMix by about 2%, and on CIFAR-100, it outperforms AlignMixup by about 1.5%. PuzzleMix and AlignMixup were the second-best performing methods on each dataset.

Training Overhead Also, we measure the training overhead of mixup methods by training the classifier for 20 epochs on CIFAR-100 with RTX2080 and Tiny-ImageNet with TITAN RTX. For a fair comparison, we measured the time without any other workloads running to avoid interference, and we did not use multiprocessing support to accelerate mixup augmentation. As Catch-up Mix does not require additional forward and backward propagation, the results show that it has only negligible overhead.

Robustness against Adversarial Attacks In this section, we assess the adversarial robustness of classifiers trained with various mixup methods. To evaluate adversarial robustness, we measure the model's performance on adversarial examples generated by applying Fast Gradient Sign Method (FGSM) (Goodfellow, Shlens, and Szegedy 2014) to the test dataset with a 4/255 l_∞ epsilon ball. The results show that Catch-up Mix improves the FGSM Top-1 error rate over the best-performing baseline by 2.34% on Tiny-ImageNet and 1.68% on CIFAR-100, as shown in Table 1 and 2.

Robustness against Data Corruption Also, We evaluate the robustness against data corruption using CIFAR-100-C, synthesized by corrupting the CIFAR-100 test dataset with 19 types of corruptions, such as fog, snow, brightness, blur, and noise, at five different strengths. The mean Corruption Error (mCE, %, ↓), calculated across all types of corruption, is reported in Table 1. Our results demonstrate that Catch-up Mix achieves the highest robustness against data corruption, even without employing additional training processes tailored explicitly for data corruption.

Robustness against Deformation In real-world scenarios, objects within images can be situated at varying distances or orientations, leading to irregularly rotated or zoomed-in/out instances. These deformations necessitate the development of a model with robust generalization capabilities to accommodate various forms of deformation. Following ManifoldMixup (Verma et al. 2019), we evaluate a pre-trained classifier (refer to Table 2) on Tiny-ImageNet datasets subjected to random rotation, random sheering, and zoom in/out to assess deformation robustness. As shown in Table 2, the model pre-trained with Catch-up Mix exhibits high robustness against deformations, successfully classifying test instances across all deformation variations compared to all the other mixup methods.

Ensuring robustness is a critical aspect of reliable deep learning models. It is vital to achieve robustness solely through a training method rather than training with corrupted samples that may limit generalization to specific distortions. For example, when a model is trained exclusively with distorted data, it specializes in recognizing the specific corruptions encountered during training, thereby hindering its ability to generalize to unseen distortions (Geirhos et al. 2018). As our empirical results show, Catch-up Mix, which emphasizes balanced filter utilization, successfully improves robustness without additional effort.

3.2 Performance on Fine-grained Datasets

To show that our improvement is not limited to specific datasets, we conduct evaluations on three standard fine-grained datasets: CUB, Cars, and Aircraft. We use ResNet-18 and DenseNet-121 as a backbone architecture. For the pre-processing, we resized input images to 256×256 resolution and randomly cropped them with 224×224 resolution. Next, we compare mixup methods used in Section 3.1. Here, we follow a hyper-parameter setting in SnapMix (Huang,

Method	CUB		Cars		Aircraft	
	R18	D121	R18	D121	R18	D121
Baseline	63.77	68.88	82.37	85.44	77.92	80.01
InputMixup	67.24	74.28	85.67	88.58	80.38	81.24
CutMix	64.08	74.62	86.57	89.10	80.05	81.15
PuzzleMix	69.33	76.32	86.51	89.96	79.18	82.59
SaliencyMix	61.74	67.53	85.66	88.90	79.78	80.88
Co-Mixup	71.82	76.88	87.31	89.83	80.17	82.14
SnapMix	70.71	75.76	87.55	90.27	80.77	83.29
Manifold	68.29	74.02	85.23	88.43	79.60	82.08
MoEx	65.00	71.05	85.37	88.94	79.21	81.48
RecursiveMix	67.02	73.80	87.36	89.71	79.90	83.35
AlignMixup	71.78	76.07	87.18	89.09	80.89	82.11
Catch-up Mix	**72.44**	**78.77**	**87.71**	**90.82**	**82.21**	**84.70**

Table 3: Top-1 accuracy rates (%, ↑) on various fine-grained datasets using ResNet-18, and DenseNet-121.

Method	CIFAR-100			Tiny-ImageNet		
	10%	25%	50%	10%	25%	50%
Baseline	35.88	60.17	70.48	28.09	43.58	56.13
InputMixup	45.65	63.41	73.19	30.70	44.85	55.18
CutMix	47.03	62.59	72.90	30.83	44.19	54.74
PuzzleMix	43.87	65.75	75.30	32.31	46.78	56.52
SaliencyMix	47.93	65.12	73.93	31.20	48.60	54.98
Co-Mixup	40.59	61.55	72.31	30.90	45.62	55.99
Manifold	49.00	64.80	73.46	33.91	48.53	56.93
MoEx	41.61	65.03	72.91	31.58	48.78	56.46
RecursiveMix	36.72	62.94	72.80	30.84	45.10	56.21
AlignMixup	50.15	67.07	75.60	35.91	49.78	58.61
Catch-up Mix	**51.26**	**69.51**	**76.95**	**36.27**	**51.06**	**60.36**

Table 4: Top-1 accuracy rate (%, ↑) of mixup baselines trained using PreActResNet-18 with a reduced number of images per class on CIFAR-100 and Tiny-ImageNet.

Wang, and Tao 2021). As shown in Table 3, Catch-up Mix shows high performance on a variety of datasets. With ResNet-18, Catch-up Mix achieves 72.44%, 87.71%, and 81.79% on CUB, Cars, and Aircraft, outperforming the best competitor by 0.62%, 0.10%, and 0.90%, respectively.

3.3 Performance on Small Datasets

Generally, training deep neural networks for real-world applications often faces the challenge of overfitting due to insufficient data. Data augmentation has been considered as a key strategy to effectively increase the dataset size and alleviate this issue. In this section, we evaluate the performance of mixup methods under data scarcity. We train PreActResNet-18 on 10%, 25%, and 50% subsets of the CIFAR-100 and Tiny-ImageNet, repeating each training three times with different seeds. Training details are the same as in Table 1 and 2. We report the average accuracy of each method over three runs in Table 4. The results show that Catch-up Mix effectively prevents the overfitting phenomenon and achieves greater performance even when there's a limited number of data per class. Notably, RecursiveMix exhibits inferior performance compared to other mixup methods. We hypothesize that this is due to its reliance on a contrastive learning framework, which requires high data diversity to learn meaningful representations.

Method	ImageNet	ImageNet-O		
	Acc (%, ↑)	FPR95 (%, ↓)	AUROC (↑)	AUPR (↑)
Baseline	76.15	-*	-*	-*
Mixup	77.46	90.06	53.55	16.27
CutMix	78.58	93.53	51.19	15.81
PuzzleMix	78.63	91.08	53.63	16.58
ManifoldMix	77.50	91.66	52.99	16.23
MoEx	79.06	93.99	51.58	15.99
RecursiveMix	**79.14**	88.54	54.99	**16.91**
Catch-up Mix	78.71	**81.92**	**55.56**	16.69

Table 5: Top-1 classification accuracy rate (%) on ImageNet-1k with ResNet-50 and out-of-distribution detection performance (FPR95, AUROC, and AUPR) on ImageNet-O. *ImageNet-O consists of data that vanilla ResNet misclassifies, meaning its detection accuracy is 0%. Here, to perform OOD evaluation, Catch-up Mix is compared with methods that publicly released their pre-trained model weights.

3.4 Performance on Large Dataset

To evaluate our method in a data-rich context, we train ResNet-50 on ImageNet-1k for 300 epochs using Catch-up Mix and compare it with mixup methods that published the ImageNet pre-trained model weights. Catch-up Mix achieves a top-1 accuracy of 78.71. Our motivation aims to leverage the under-exploited model capacity, often left unused due to the early convergence. For expansive datasets such as ImageNet-1K, the inherent diversity and variety within the data naturally promote the model's generalization capabilities, diminishing the potentially unused capacity and the chances of early convergence.

Robustness against OOD We also verify that our feature diversity and robustness improvements are valid for the ImageNet-trained model using ImageNet-O, out-of distribution (OOD) detection dataset (Hendrycks et al. 2021). OOD detection tasks can provide insights into the generalization capability and feature representations of models. From ImageNet-22K examples with the ImageNet-1K class removed, ImageNet-O consists of images that vanilla ResNet-50 confidently classifies as belonging to one of the ImageNet-1K classes. OOD detection performance in Table 5 shows that our method can effectively differentiate between the data it was trained on (in-distribution) and unfamiliar data (OOD). Having the advantage of a variety of features, Catch-up Mix improves the robustness to unseen data and builds a reliable model.

Accuracy-Robustness Trade-off Even when there is plenty of data and little surplus capacity in the model, we encourage a model to utilize diverse features and, if necessary,

Model (Epoch)	CIFAR-100 R18* (400)	CIFAR-100 WRN16 (400)	CIFAR-100 RX50* (400)	TI R18* (400)	TI PR50 (1200)
Baseline	77.73	79.37†	80.24	61.68	62.32
MixUp	79.34	80.12†	82.54	63.86	66.51
CutMix	79.58	80.29†	78.52	65.53	67.02
PuzzleMix	80.82	80.75†	82.84	65.81	71.39
SaliencyMix	79.64	80.41†	78.63	64.60	66.37
Co-Mixup	80.87	80.43†	82.88	65.92	65.26
Manifold	80.18	80.77†	82.56	64.15	66.91
MoEx	N/A	79.31	N/A	N/A	68.98
RecursiveMix	N/A	81.27	N/A	N/A	68.45
AlignMixup	80.80	81.23†	N/A	66.87	69.36
Catch-up Mix	**82.10**	**81.64**	**83.56**	**68.84**	**72.58**

Table 6: Top-1 accuracy rates (%) on CIFAR-100 and Tiny-ImageNet using various architectures. All the results of * refer to (Li et al. 2022). The results of WRN16† are from (Venkataramanan et al. 2022).

sacrifice accuracy for robustness (Table 5). When training ResNet-50 on simpler classification tasks like CIFAR-100 and considering the model's capacity (e.g., number of parameters), encouraging feature diversity can enhance both robustness and accuracy without sacrificing performance. Conversely, when training on larger datasets for more challenging tasks such as ImageNet, prioritizing feature diversity might result in a trade-off between accuracy and robustness.

3.5 Performance on Various Architectures

Here, we evaluate Catch-up Mix on various architectures to show that our performance gain is not limited to specific architectures. We evaluate ResNet-18 (R18), Wide ResNet-16-8 (WRN16), and ResNeXt-50 (RX50) on CIFAR-100 and evaluate ResNet-18 (R18) and PreActResNet-50 (PR50) on Tiny-ImageNet. Experiment settings for R18 and RX50 were adopted from Li et al. (2022), while for WRN16 and PR-50, we referenced Co-Mixup (Kim et al. 2020) and AlignMixup (Venkataramanan et al. 2022). E.g., we train PR-50 for 1200 epochs and the others for 400 epochs. As depicted in Table 6, Catch-up Mix consistently outperformed other mixup methods in various architectures.

3.6 Loss Landscape

The concept of generalization in a neural network pertains to the model's ability to effectively apply the knowledge obtained from the training data to unseen test data. A well-generalizing model reduces the discrepancy called the 'generalization gap' between the performance during training and testing. In this context, numerous studies (Keskar et al. 2017; Foret et al. 2021; Choromanska et al. 2015) have discussed that a network's convergence to flat minima enhances its generalization and robustness performance. To this end, the method generates a flatter loss landscape compared to others would be better generalization and robustness. Following the implementation of Yao et al. (2020), we plot the loss landscape of mixup methods as shown in Figure 3.

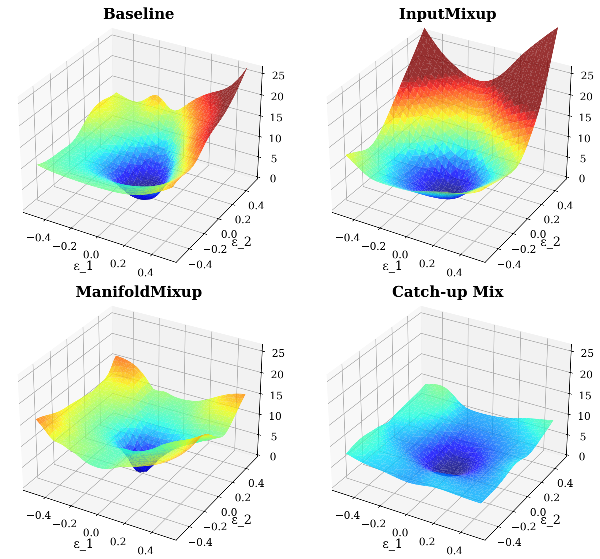


Figure 3: Loss Landscapes of mixup methods on Tiny-ImageNet using PreActResNet-18.

We use InputMixup (Zhang et al. 2018) as the input-level mixup except for 'Baseline' for the fair comparison. Loss values of other mixup methods change extremely, even with a slight change. However, interestingly, the model trained with Catch-up Mix has a flat loss landscape, and its loss value does not explode with small changes compared to InputMixup and ManifoldMixup. As intuitively shown by the loss landscape, Catch-up Mix consistently achieves high robustness and generalization on various benchmarks.

4 Conclusion and Future Work

We have addressed the challenge of models relying heavily on a small subset of filters for making predictions, resulting in limited generalization and robustness. We introduced Catch-up Mix, a technique that enhances learning opportunities for filters that may fall behind during training. By mixing activation maps with lower norms, Catch-up Mix encourages the development of more diverse representations and mitigates over-reliance problems. Our experiments highlight Catch-up Mix's improved performance on vision classification datasets, enhancing robustness against adversarial attacks, data corruption, and deformations.

Previous studies have not investigated the model's unused capacity during training. Our approach aims to bridge this gap, moving from a state where a significant portion of the model's capacity remains idle to an optimized state where every filter actively contributes to predictions. Therefore, this enhancement is more pronounced when the dataset and regularizations cannot sufficiently utilize the model's capacity. However, a challenge arises when all the weights can already be fully optimized (e.g., plenty of training data); utilizing various features becomes a cost for the generalization, resulting in an accuracy-robustness trade-off. In future work, our research will address these considerations, further improving the performance and adaptability of Catch-up Mix.

Acknowledgements

This work was supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No.2021-0-00456, Development of Ultra-high Speech Quality Technology for remote Multi-speaker Conference System), and by the Korea Institute of Science and Technology (KIST) Institutional Program.

References

Choromanska, A.; Henaff, M.; Mathieu, M.; Arous, G. B.; and LeCun, Y. 2015. The loss surfaces of multilayer networks. In *Artificial intelligence and statistics*, 192–204. PMLR.

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

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

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2021. Sharpness-aware Minimization for Efficiently Improving Generalization. In *International Conference on Learning Representations*.

Geirhos, R.; Temme, C. R.; Rauber, J.; Schütt, H. H.; Bethge, M.; and Wichmann, F. A. 2018. Generalisation in humans and deep neural networks. *Advances in neural information processing systems*, 31.

Goodfellow, I.; Bengio, Y.; and Courville, A. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Goodfellow, I. J.; Shlens, J.; and Szegedy, C. 2014. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.

Han, D.; Kim, J.; and Kim, J. 2017. Deep pyramidal residual networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5927–5935.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Identity mappings in deep residual networks. In *European conference on computer vision*, 630–645. Springer.

Hendrycks, D.; and Dietterich, T. 2019. Benchmarking neural network robustness to common corruptions and perturbations. *arXiv preprint arXiv:1903.12261*.

Hendrycks, D.; Zhao, K.; Basart, S.; Steinhardt, J.; and Song, D. 2021. Natural adversarial examples. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15262–15271.

Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. R. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

Huang, G.; Liu, Z.; Van Der Maaten, L.; and Weinberger, K. Q. 2017. Densely Connected Convolutional Networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2261–2269. IEEE.

Huang, S.; Wang, X.; and Tao, D. 2021. SnapMix: Semantically Proportional Mixing for Augmenting Fine-grained Data. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Kang, M.; and Kim, S. 2023. GuidedMixup: an efficient mixup strategy guided by saliency maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 1096–1104.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *International Conference on Learning Representations*.

Kim, J.; Choo, W.; Jeong, H.; and Song, H. O. 2020. Co-Mixup: Saliency Guided Joint Mixup with Supermodular Diversity. In *International Conference on Learning Representations*.

Kim, J.-H.; Choo, W.; and Song, H. O. 2020. Puzzle mix: Exploiting saliency and local statistics for optimal mixup. In *International Conference on Machine Learning*, 5275–5285. PMLR.

Krause, J.; Stark, M.; Deng, J.; and Fei-Fei, L. 2013. 3D Object Representations for Fine-Grained Categorization. In *4th International IEEE Workshop on 3D Representation and Recognition (3dRR-13)*. Sydney, Australia.

Krizhevsky, A. 2009. Learning Multiple Layers of Features from Tiny Images. *Master's thesis, University of Tront*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Li, B.; Wu, F.; Lim, S.-N.; Belongie, S.; and Weinberger, K. Q. 2021. On feature normalization and data augmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 12383–12392.

Li, S.; Wang, Z.; Liu, Z.; Wu, D.; and Li, S. Z. 2022. Openmixup: Open mixup toolbox and benchmark for visual representation learning. *arXiv preprint arXiv:2209.04851*.

Maji, S.; Rahtu, E.; Kannala, J.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Nam, J.; Cha, H.; Ahn, S.; Lee, J.; and Shin, J. 2020. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33: 20673–20684.

Ng, A. Y. 2004. Feature selection, L 1 vs. L 2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, 78.

Sandler, M.; Howard, A.; Zhu, M.; Zhmoginov, A.; and Chen, L.-C. 2018. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 4510–4520.

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

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

Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.; and Fergus, R. 2013. Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.

Uddin, A. S.; Monira, M. S.; Shin, W.; Chung, T.; and Bae, S.-H. 2020. SaliencyMix: A Saliency Guided Data Augmentation Strategy for Better Regularization. In *International Conference on Learning Representations*.

Venkataramanan, S.; Kijak, E.; Amsaleg, L.; and Avrithis, Y. 2022. AlignMixup: Improving Representations By Interpolating Aligned Features. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 19174–19183.

Verma, V.; Lamb, A.; Beckham, C.; Najafi, A.; Mitliagkas, I.; Lopez-Paz, D.; and Bengio, Y. 2019. Manifold mixup: Better representations by interpolating hidden states. In *International Conference on Machine Learning*, 6438–6447. PMLR.

Wah, C.; Branson, S.; Welinder, P.; Perona, P.; and Belongie, S. 2011. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, California Institute of Technology.

Xie, S.; Girshick, R.; Dollár, P.; Tu, Z.; and He, K. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1492–1500.

Yang, L.; Li, X.; Zhao, B.; Song, R.; and Yang, J. 2022. Recursivemix: Mixed learning with history. *arXiv preprint arXiv:2203.06844*.

Yao, Z.; Gholami, A.; Keutzer, K.; and Mahoney, M. W. 2020. Pyhessian: Neural networks through the lens of the hessian. In *2020 IEEE international conference on big data (Big data)*, 581–590. IEEE.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 6023–6032.

Zagoruyko, S.; and Komodakis, N. 2016. Wide Residual Networks. In *British Machine Vision Conference 2016*. British Machine Vision Association.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2018. mixup: Beyond Empirical Risk Minimization. In *International Conference on Learning Representations*.