From Confusion to Clarity: Generalized $mixup$

Tucker C. Simpson
Dartmouth College
tucker.c.simpson.24@dartmouth.edu

Harrison F. Stropkay
Dartmouth College
harrison.f.stropkay.25@dartmouth.edu

Abdullah Al Maruf
Dartmouth College
abdullah.al.maruf.gr@dartmouth.edu

David C. Mason
Dartmouth College
david.c.mason.25@dartmouth.edu

Anastasia C. Johnson
Dartmouth College
anastasia.c.johnson.25@dartmouth.edu

Abstract

Data augmentation significantly enhances the ability of machine learning models to generalize on unseen data sets. The traditional $mixup$ technique, which linearly interpolates between training examples and their labels, has been effective in increasing model robustness but lacks focus on specific learning challenges. In this work, we introduce a novel adaptation of $mixup$ that integrates a dynamic, confusion matrix-based selection strategy for pairing examples. This method prioritizes frequently misclassified combinations, thereby targeting critical areas where the model underperforms. Our proposed approach, tested extensively on CIFAR-10 and CIFAR-100 datasets with ResNet18, shows that adaptive interpolation based on the confusion matrix notably improves model accuracy. Results demonstrate that lower γ values, which adjust the influence of the confusion matrix, yield the best performance, enhancing generalization while avoiding underexposure to pairs of images from specific classes. Our method, a generalization of $mixup$, addresses current challenges in machine learning and makes strides in both theoretical and practical aspects of model training.

1 Introduction

Data augmentation is a critical technique in machine learning employed to enhance model robustness. Introduced by (12), $mixup$ creates new training examples by computing linear interpolations of

pairs of examples and their corresponding labels. This technique prevents overfitting and promotes smoother decision boundaries between data points.

The core idea behind mixup is Vicinal Risk Minimization (VRM). Unlike Empirical Risk Minimization (ERM), which aims to minimize errors on the only training data and can lead to overfitting, VRM focuses on reducing errors in the neighborhood of the training examples. Under $mixup$, the model is trained on convex combinations of training examples.

While $mixup$ has been successful, its indiscriminate approach to selecting pairs for interpolation does not consider the underlying structure or specific characteristics of the data. Other approaches, namely DMix, (6), and CoMixup, (4), seek to exploit innate traits of a dataset to more efficiently apply standard $mixup$. Building on these advancements, this paper introduces a novel approach that utilizes a variant of the confusion matrix to guide the selection of pairs for $mixup$. Our method targets misclassifications during training by upsampling the pairs of classes that are often mistaken for each other. Our approach aims to utilize $mixup$ to smooth the decision boundary between classes that a model finds difficult to distinguish. Our strategy of adaptive sampling leads to faster convergence and improved test accuracy. We show that standard $mixup$ is a special case of our method; thus, our approach is a generalization of $mixup$. The code used to run our experiments can be found here or at the following link:

`https://github.com/harrison-f-stropkay/78-final-project.git`

In this paper, we present a comprehensive description of our method, including its theoretical framework, implementation details, and empirical evaluation. By synthesizing insights from prior $mixup$ variations and incorporating a mechanism to adapt based on error analysis, we contribute a novel perspective to the field of data augmentation, aiming to provide a more targeted and effective approach to enhancing model generalization.

2 Preliminaries

The original $mixup$ algorithm involves linearly interpolating the features of two training examples and their labels. This process prevents models with high Vapnik-Chervonenkis complexity from memorizing features. The degree to which the mixed example resembles the first example in its pair is controlled by λ, a value between 0 and 1 drawn from the beta distribution. (12) derives a new, mixed data point $\hat{x}$ and its label $\hat{y}$:

$$\hat{x} = \lambda x_i + (1 - \lambda) x_j$$

$$\hat{y} = \lambda y_i + (1 - \lambda) y_j$$

where x_i and x_j are raw input vectors and y_i and y_j are their corresponding labels.

For each instance where a pair of inputs is mixed, λ is selected from the distribution $\beta(\alpha, \alpha)$, where $\alpha \in (0, \infty)$. When $\alpha = 0$, the distribution only takes the values 0 and 1, and as $\alpha \to \infty$, the distribution only takes the value 0.5. At α = 1, the distribution is uniform between 0 and 1.

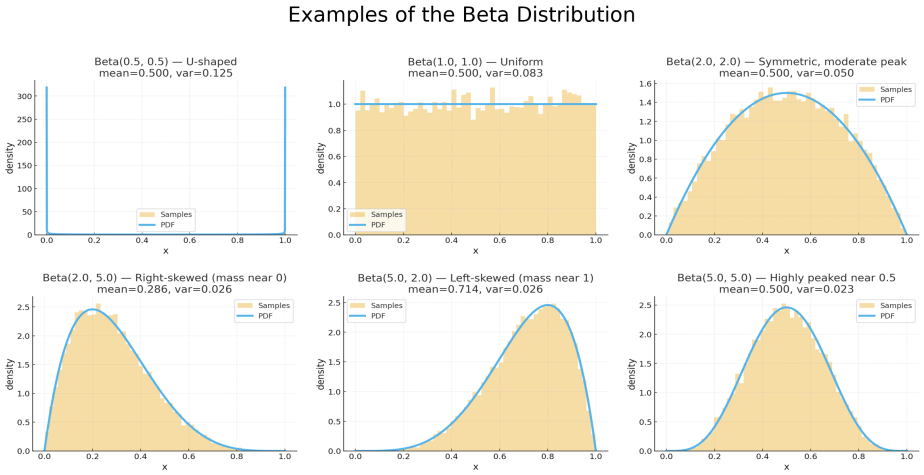

Figure 1: Variations of the beta distribution.

The original $mixup$ algorithm combines data points indiscriminately, ignoring any trends in the data. There have been attempts to improve $mixup$'s performance by being more selective about which pair of inputs is linearly interpolated; for example, the aforementioned DMix (6) combines data points only if the hyperbolic tangential distance from each other is significantly high.

The model introduced in this paper selects which data points are mixed together using a different criterion. After each epoch of training, we use a variant of the confusion matrix to upsample examples from classes C_i and C_j if examples from C_i are often attributed to C_j. Thus, we do not modify how a pair of inputs is mixed, but hypothesize that changing how the pair of inputs is selected will decrease the time to convergence and improve accuracy.

3 Proposed Method

We propose a method that generalizes $mixup$. (12) shows that $mixup$ smooths the decision boundary between data points. We hypothesize that a model can benefit from $mixup$ between some pairs of data points more than others. Specifically, it could be especially helpful to perform $mixup$ on data points from pairs of classes that a model often mistakes for one another. Our method follows this logic, upsampling pairs of data points belonging to pairs of classes that the model often confuses.

Standard $mixup$ pairs examples by mixing two copies of the dataset: the first copy is the dataset in its original order, and the second copy is a permutation of the dataset chosen uniformly at random.

Our version of generalized $mixup$ creates a permutation of the dataset using the confusion matrix. For each class C with n examples, we sample without replacement n examples from the dataset to mix with the examples of C via a probability distribution P_C over all the classes, where $P_C(C')$ is relatively high when the model often mislabels examples of class C as C'. The probability distributions P_C can each be represented as a vector of length C, and there are C total P_C. This construction naturally lends itself to a $C \times C$ matrix M. We organize M such that each P_C is a row in M. Because the rows of M each represent a probability distribution, each row must sum to 1. Each column must also sum to 1 so that the probability that two examples are mixed is independent of the sampling order. [1]

The confusion matrix is a matrix representation of the predictions of a multi-class classification model. For each prediction that the model makes, exactly one element of the confusion matrix is incremented, namely the element in the row of the true class and column of the predicted class of the data point. The $mixup$ algorithm turns a multi-class task into a multi-label task: two one-hot labels are interpolated into a vector that generally has two non-zero elements. There is not a standard way to construct a two-dimensional confusion matrix for a multi-label task, so we do the following: for our interpolated example $\hat{x} = \lambda x_i + (1 - \lambda)x_j$ and interpolated label vector $\hat{y} = \lambda y_i + (1 - \lambda)y_j$, where x_i is from class C_a and x_j is from class C_b, we distribute the predicted label vector $\hat{y}_p$ by incrementing the $a^{\texttt{th}}$ row in the confusion matrix by $\lambda\hat{y}_p$ and incrementing the $b_{\texttt{th}}$ row by $(1 - \lambda)\hat{y}_p$. Equivalently, for each true label vector $\hat{y}$ and predicted label vector $\hat{y}_p$, we increment the confusion matrix by the vector outer product $\hat{y}\hat{y}_p^T$.

At the model's initialization, by random chance, examples of C_a could often be mislabeled as C_b (assume $i \neq j$). This would then cause upsampling of examples from C_a and C_b, which would then likely lead to an increase in the four cells in rows a and b and columns a and b. These start conditions ultimately cause more upsampling between classes C_a and C_b, and so on. To prevent this feedback loop, we modify the result of our vector outer product such that for an example mixed from examples from classes C_a and C_b, we set all values in columns a and b to 0. In other words, our confusion matrix only contains the values attributed to incorrect labels.

Now, we must modify our confusion matrix so that it is doubly stochastic, satisfying the criteria of M. By excluding in the confusion matrix the predicted value of the true classes, the diagonal of our confusion matrix only contains values of 0. It is beneficial to mix between examples of the same class, as (12) showed that mixing between same classes alone beats ERM. Thus, we first scale each row such that the sum of all values equals $\frac{c-1}{c}$, where c is the number of classes. We then set

[1] Consider an alternative. Assume that the first row sampled is coincidentally all zeros, except for a value of 1 for class C. Therefore, all examples of C will be sampled before we sample for any subsequent row. If any later rows have a nonzero value for class C, there are no examples of class C to sample, and thus sampling is impacted by row order.

each element along the diagonal of the confusion matrix to $\frac{1}{c}$. Therefore, each row of our confusion matrix now gives us a probability distribution from which we can sample examples—a probability distribution that places increased importance on the classes that the model often mislabels.

To make the matrix doubly stochastic, we use the Sinkhorn-Knopp algorithm, which alternates between normalizing the rows and columns. Empirically, we find that this process does not significantly impact our matrix. For $\gamma = 1$, the average difference between each matrix element before and after Sinkhorn-Knopp is roughly 0.017.

We then iterate through the rows of the confusion matrix, sampling without replacement from the training data. At each row a, we sample n examples to be mixed with the examples of class C_i. We define a parameter γ that amplifies or reduces the effects of the confusion matrix on the sampling. Before we make our matrix doubly stochastic, we raise each element to the power of γ. As $\gamma \to \infty$, the largest elements dominate. As $\gamma \to 0$, all elements of the confusion matrix approach $\frac{1}{c}$. In this case, when we sample from a row a, each example in the dataset is equally likely to be sampled. Therefore, the permutation is completely random. Thus, as $\gamma \to 0$, we recover standard $mixup$.

This reasoning justifies our choice to initialize our matrix such that every element is $\frac{1}{c}$. We compute our matrix after each training epoch using the training predictions and exponential averaging of the confusion matrix with momentum 0.5. In practice, (12) used batching and see no detriments in performance, so we do the same. Because sampling from our doubly stochastic matrix relies on having an equal number of examples from each class, we use stratified sampling for each batch.

Algorithm 1 summarizes the full training loop, including the construction of the confusion-guided pairing matrix M, the partner assignment sampled without replacement, and the Sinkhorn-normalized update with guidance parameter γ.

4 Experiments

4.1 Dataset

For our experiments, we utilize datasets specifically designed for benchmarking image classification models (5). The CIFAR-10 and CIFAR-100 datasets comprise 60,000 color images, each measuring 32×32 pixels, distributed across 10 and 100 classes, respectively. Each dataset is split into 50,000 training images and 10,000 testing images. These datasets contain images from various natural categories, making them ideal for evaluating the performance of deep learning models in recognizing and differentiating between diverse visual patterns. Detailed characteristics of these datasets can be found on the CIFAR dataset website[2].

[2] `https://www.cs.toronto.edu/~kriz/cifar.html`

Algorithm 1 Generalized *mixup* Training Procedure

Input: Training dataset $\mathcal{D} = \{(x_k, y_k)\}_{k=1}^{N}$ with c classes, model f_θ, number of epochs E, batch size B, mixup parameter α, confusion influence parameter γ, momentum $m = 0.5$
Initialize confusion matrix $CM \in \mathbb{R}^{c \times c}$ with all entries $\frac{1}{c}$
Normalize CM to be doubly stochastic using Sinkhorn-Knopp algorithm
for each epoch $e = 1$ to E **do**
    Initialize temporary confusion matrix $CM_{temp} \leftarrow \mathbf{0}_{c \times c}$
    for each mini-batch $\mathcal{B}$ sampled stratified from $\mathcal{D}$ (equal examples per class) **do**
        for each class index a in the classes present in $\mathcal{B}$ **do**
            Let $n_a =$ number of examples in $\mathcal{B}$ from class a
            Sample n_a examples $\{x_j\}_{j=1}^{n_a}$ from $\mathcal{B}$ (without replacement) according to probabilities in row a of CM
            Pair them with the n_a examples $\{x_i\}_{i=1}^{n_a}$ from class a in $\mathcal{B}$
        end for
        Form permuted batch $\mathcal{B}'$ from the sampled pairs
        Sample $\lambda \sim \text{Beta}(\alpha, \alpha)$
        Compute mixed batch: $\hat{x} = \lambda x_i + (1 - \lambda) x_j$, $\hat{y} = \lambda y_i + (1 - \lambda) y_j$ for pairs $(x_i, y_i), (x_j, y_j) \in \mathcal{B} \times \mathcal{B}'$
        Compute predictions $\hat{y}_p = f_\theta(\hat{x})$
        Compute loss $\mathcal{L}$ and update model parameters θ
        for each mixed example from classes C_a, C_b **do**
            Compute outer product increment: $\Delta = \hat{y} \cdot \hat{y}_p^T$
            Set $\Delta[:, a] = 0$ and $\Delta[:, b] = 0$ (exclude correct class columns)
            $CM_{temp} \leftarrow CM_{temp} + \Delta$
        end for
    end for
    Normalize CM_{temp} rows to sum to $\frac{c-1}{c}$, set diagonal to $\frac{1}{c}$
    Update $CM \leftarrow m \cdot CM + (1 - m) \cdot CM_{temp}$ (exponential averaging)
    Raise non-diagonal elements of CM to power γ (diagonal remains $\frac{1}{c}$)
    Normalize CM to be doubly stochastic using Sinkhorn-Knopp algorithm
end for
Output: Trained model f_θ

4.2 Results

We conduct experiments on γ to evaluate the effect of γ on model performance. As γ increases, pairs of examples from classes that the model confuses are more likely to be mixed. We perform the γ experiments on ResNet18 with a learning rate of 0.1. We decrease the learning rate to 0.01 after 50 epochs and 0.001 after 75 epochs. We train for 100 total epochs and use a weight decay of 0.0001. Our $mixup$ parameter α is set to 1. For CIFAR-10, we use a batch size of 250, and for CIFAR-100, we use a batch size of 1000. Results are captured in Table 1 and Figure 2.

Figure 3 shows heatmaps of the average confusion matrix for $\gamma = 0.5$ and $\gamma = 2$. For high values of γ, pairs of classes are more likely to contain classes that a model often confuses. For example, automobiles and trucks are often sampled as pairs. The same can be said for airplanes and ships and dogs and cats. For $\gamma = 2$, this effect is amplified compared to $\gamma = 0.5$.

We show how the standard deviation of the confusion matrix evolves over epochs in Figure 4. Upon initialization, every value in the confusion matrix is $\frac{1}{c}$, where c is the number of classes. The model

γ	**Test error: CIFAR-10**	**Test error: CIFAR-100)**
0.063	0.064	0.277
0.125	0.062	**0.271**
0.25	**0.059**	0.273
0.5	0.063	0.276
1	0.062	0.276
2	0.068	0.305
Standard *mixup*	0.065	0.276

Table 1: Test accuracies on CIFAR10 and CIFAR100 for varying γ values.

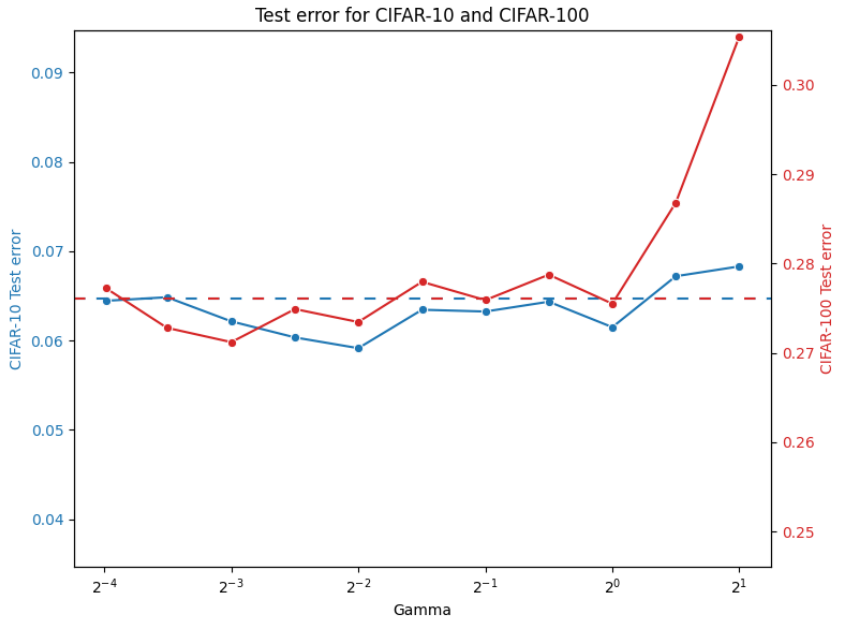

Figure 2: Test error as a function of γ for CIFAR-10 and CIFAR-100. The dotted line is the test error of standard $mixup$.

is initially a poor classifier, so the standard deviation of the confusion matrix increases quickly, only hindered by the exponential averaging momentum value of 0.5. As model performance improves throughout training, we assume that the model mislabels examples of a class to another class less frequently, so the standard deviation slowly decreases. This could be compounded by the effect of our method. Because we are intentionally upsampling pairs of classes that are often confused, the model creates a more linear decision boundary between such pairs. As training continues, the model is less likely to continue making the same mistakes. The average standard deviation for each value of γ is smaller for CIFAR-100 than for CIFAR-10, but the general trend is the same.

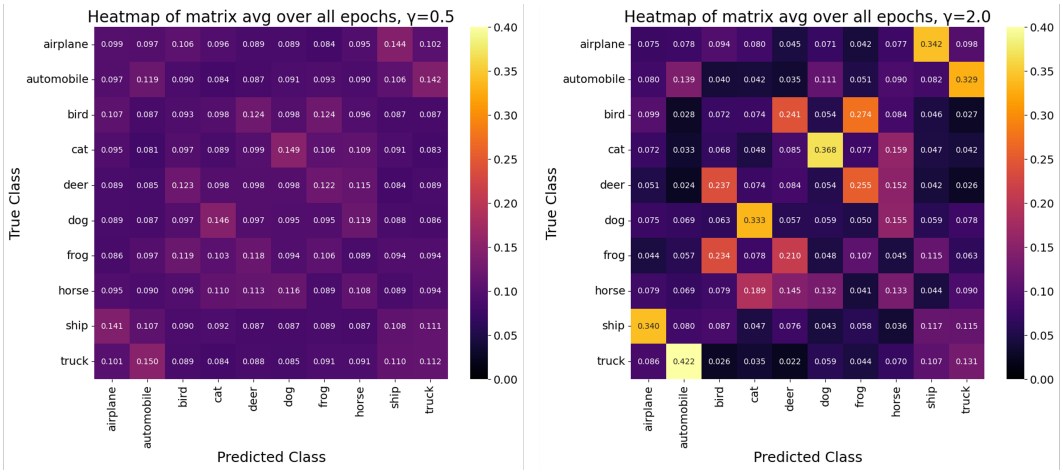

Figure 3: Heatmaps of the confusion matrix for $\gamma = 0.5$ and $\gamma = 2$ for CIFAR-10 averaged over all epochs.

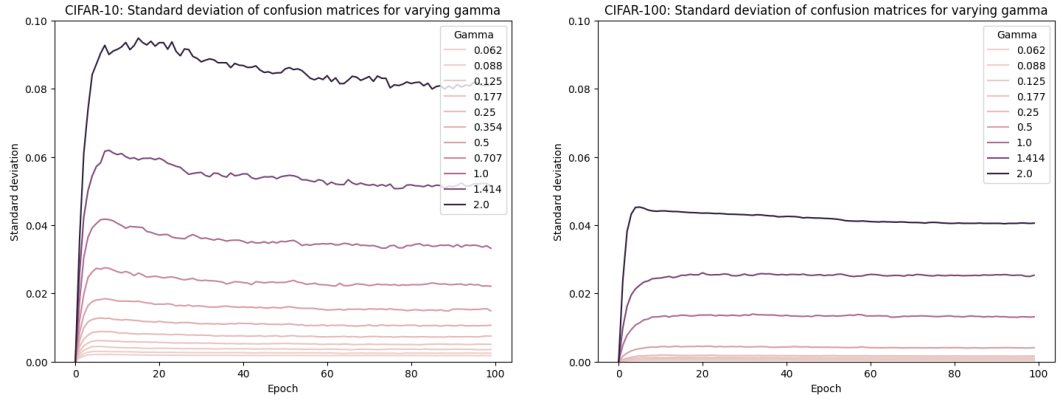

Figure 4: The standard deviation of the confusion matrix as a function of epoch for varying values of γ for both CIFAR-10 and CIFAR-100.

5 Related Work

Our method is inspired by previous scholarship. Since the original $mixup$ paper was published in 2018, work has focused on improving the implementation and adapting it to modality specific tasks.

5.1 Augmentation based on Input Saliency

Although the original $mixup$ algorithm is highly effective as a regularizer, many other approaches have made improvements by exploiting the relationships between data inputs. RICAP, presented in (7), samples four random inputs from a dataset, randomly crops them, and then forms a new image from four crops. SaliencyMix, introduced by (8), uses a saliency map to crop "important" crops from paired images. It seeks to improve upon previous approaches by pasting the most relevant patch of the second image to the first. They argue this new method is better than other random cut and mix

strategies because the augmented image better reflects the intrinsically meaningful information from both inputs.

Similarly, Attentive Cut-Mix, introduced by (9), uses a feature extractor based on an attention maps of a transformer model. After finding an input's most "descriptive" aspects, it implements Cut-Mix (11), improving upon that model's random cut and paste between the input pair. In this example, the resulting augmented image of a dog and cat could include the body of a dog with the eyes, nose, and ears of a cat. Co-$mixup$ strategically blends features from different samples, emphasizing saliency and diversity through an optimization framework involving submodular and supermodular functions (4).

Finally, CowMask utilizes on the filtering found in most CNNs to augment photos (2). It randomly applies a series of masking filters over a given image, preserving the overall image structure. This pattern of masking resembles the spots on a cow, hence the approach's name.

5.2 Applying *mixup* based on Input Saliency

The previous approaches diverge sharply from our method. For one, most of them augment based on varying implementations of Cut-Mix (11). However, other algorithms seek to improve the original $mixup$ algorithm by changing which images are mixed. For example, Recursive-mix (10) mixes pairs of inputs, similar to the original implementation of $mixup$. However, mixes are also used as input to later mixes, making the algorithm recursive. MixStyle seeks to mix inputs, specifically images, with different styles (13). For example, it would iterate through a set of paintings and form a feature map identifying the Mona Lisa, Sistine Chapel, and the Birth of Venus as 'Renaissance Art.' Afterward, the algorithm mixes the style features of images and uses the mixed style statistics to reconstruct new images.

AugMix abandons mixing based on a permutation of the input set and instead performs a series of transformations on a given input and then mixes it with the original image (3). In this approach, all inputs are mixed with variations of themselves, rather than other members of the input set. Finally, ReMix, proposed by (1), operates in a similarly to CoMix, introduced in (4). It mixes the most important parts of two inputs using a saliency map; however, unlike regular $mixup$ and CoMix, it decouples the mixing of features and labels. While the inputs are mixed, the labels are set with a bias toward the minority class. This small alteration improves the model's performance on classes with a smaller number of inputs, which improves generalization by eliminating the larger class's influence on the model's output.

6 Conclusion

Standard $mixup$ mixes the list of training examples with a random permutation of itself. We have empirically found that our method improves upon standard $mixup$. Specifically, standard $mixup$ is a special case of our method when $\gamma = 0$. We show that there exist non-zero values of γ that improve upon standard $mixup$. For CIFAR-10, $\gamma = 0.25$ improves upon standard $mixup$ by 0.6%. For CIFAR-100, $\gamma = 0.125$ improves upon standard $mixup$ by 0.5%.

We find that the model confuses intuitively similar classes of images like trucks and automobiles. We upsample the pairs of such similar classes to make the decision boundary between them more linear. This claim is supported by the decrease in standard deviation over epochs and our performance improvement over standard $mixup$. This defends our hypothesis that some pairs of classes can benefit from mixing more than others, and that our method of upsampling is one effective method of doing so.

The results indicate upsampling can be overdone. For larger values of γ, the model likely sees a much smaller range of class combinations, leading to decreased robustness. Our proposed method represents a step forward in the field of data augmentation. It not only uses the principles of Vicinal Risk Minimization but also introduces an adaptive approach that is absent in standard $mixup$. The ability to integrate error analysis directly into the augmentation process opens new avenues for research and application, particularly in domains where data is limited or highly complex. Future work should focus on expanding this method's applicability to other forms of data beyond images. Work could also be done to find general rules of which value of γ most improves model performance given the dataset and model architecture.

References

[1] Chou, H.-P., Chang, S.-C., Pan, J.-Y., Wei, W., and Juan, D.-C. (2020). Remix: rebalanced mixup. In *Computer Vision–ECCV 2020 Workshops: Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 95–110. Springer.

[2] French, G., Oliver, A., and Salimans, T. (2020). Milking cowmask for semi-supervised image classification. *arXiv preprint arXiv:2003.12022*.

[3] Hendrycks, D., Mu, N., Cubuk, E. D., Zoph, B., Gilmer, J., and Lakshminarayanan, B. (2019). Augmix: A simple data processing method to improve robustness and uncertainty. *arXiv preprint arXiv:1912.02781*.

[4] Kim, J.-H., Choo, W., Jeong, H., and Song, H. O. (2021). Co-mixup: Saliency guided joint mixup with supermodular diversity. *arXiv preprint arXiv:2102.03065*.

[5] Krizhevsky, A. (2009). Learning multiple layers of features from tiny images. *Technical Report*.

[6] Sawhney, R., Thakkar, M., Pandit, S., Soun, R., Jin, D., Yang, D., and Flek, L. (2022). Dmix: Adaptive distance-aware interpolative mixup. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 606–612.

[7] Takahashi, R., Matsubara, T., and Uehara, K. (2018). Ricap: Random image cropping and patching data augmentation for deep cnns. In *Asian conference on machine learning*, pages 786–798. PMLR.

[8] Uddin, A., Monira, M., Shin, W., Chung, T., Bae, S.-H., et al. (2020). Saliencymix: A saliency guided data augmentation strategy for better regularization. *arXiv preprint arXiv:2006.01791*.

[9] Walawalkar, D., Shen, Z., Liu, Z., and Savvides, M. (2020). Attentive cutmix: An enhanced data augmentation approach for deep learning based image classification. *arXiv preprint arXiv:2003.13048*.

[10] Yang, L., Li, X., Zhao, B., Song, R., and Yang, J. (2022). Recursivemix: Mixed learning with history. *Advances in Neural Information Processing Systems*, 35:8427–8440.

[11] 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*, pages 6023–6032.

[12] Zhang, H., Cisse, M., Dauphin, Y. N., and Lopez-Paz, D. (2017). mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*.

[13] Zhou, K., Yang, Y., Qiao, Y., and Xiang, T. (2021). Domain generalization with mixstyle. *arXiv preprint arXiv:2104.02008*.