Learn2Mix: Training Neural Networks Using Adaptive Data Integration

Shyam Venkatasubramanian
Duke University
shyam.venkatasubramanian@duke.edu

Vahid Tarokh
Duke University
vahid.tarokh@duke.edu

Abstract

Accelerating model convergence in resource-constrained environments is essential for fast and efficient neural network training. This work presents *learn2mix*, a new training strategy that adaptively adjusts class proportions within batches, focusing on classes with higher error rates. Unlike classical training methods that use static class proportions, learn2mix continually adapts class proportions during training, leading to faster convergence. Empirical evaluations on benchmark datasets show that neural networks trained with learn2mix converge faster than those trained with existing approaches, achieving improved results for classification, regression, and reconstruction tasks under limited training resources and with imbalanced classes. Our empirical findings are supported by theoretical analysis.

1 Introduction

Deep neural networks have become essential tools across various applications of machine learning, including computer vision [Krizhevsky et al., 2012, Simonyan and Zisserman, 2014, He et al., 2016], natural language processing [Vaswani et al., 2017, Devlin et al., 2018, Radford et al., 2019, Touvron et al., 2023], and speech recognition [Hinton et al., 2012, Baevski et al., 2020]. Despite their ability to learn and model complex, nonlinear relationships, deep neural networks often require substantial computational resources during training. In resource-constrained environments, this demand poses a significant challenge [Goyal et al., 2017], making the development of efficient and scalable training methodologies increasingly crucial to fully leverage the capabilities of these models.

Training deep neural networks relies on the notion of empirical risk minimization [Vapnik and Bottou, 1993], and typically involves optimizing a loss function using gradient-based algorithms [Rumelhart et al., 1986, Bottou, 2010, Kingma and Ba, 2014]. Techniques such as regularization [Srivastava et al., 2014, Ioffe and Szegedy, 2015] and data augmentation [Shorten and Khoshgoftaar, 2019], learning rate scheduling, [Smith, 2017] and early stopping [Prechelt, 1998], are commonly employed to enhance generalization and prevent overfitting. However, the efficiency of the training process itself remains a critical concern, particularly in terms of convergence speed and computational resources.

Within this context, adaptive training strategies, which target enhanced generalization by modifying aspects of the training process, have emerged as promising approaches. Methods such as curriculum learning [Bengio et al., 2009, Graves et al., 2017, Wang et al., 2021] adjust the order and difficulty of training samples to facilitate more effective learning. Insights from these adaptive training strategies can be extended to the class imbalance problem [Wang et al., 2019], where underrepresented classes are intrinsically harder to learn due to data scarcity [Buda et al., 2018], a challenge intensified in adversarial settings where safe data collection is severely limited [Wang and Gursoy, 2023]. These methods are typically categorized into data-level methods, such as oversampling and undersampling [Chawla et al., 2002] and algorithm-level schemes, including class-balanced loss functions [Lin et al.,

GitHub repository: https://github.com/shyamven/Learn2Mix.

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

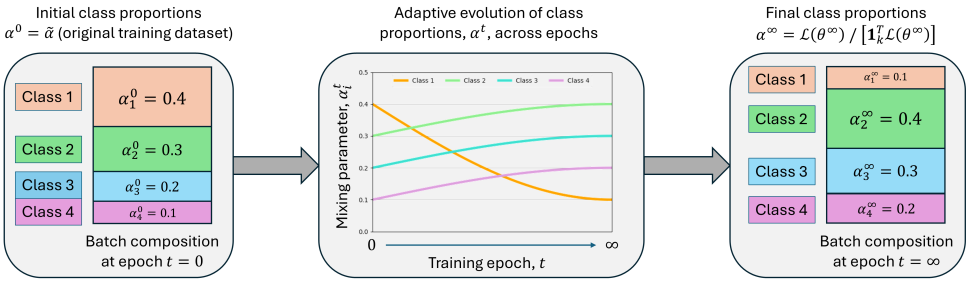


Figure 1: Illustration of the learn2mix training mechanism. The class-wise composition of batches is adaptively modified during training using instantaneous class-wise error rates.

2017]. However, developing adaptive neural network training methodologies that *accelerate* model convergence, while ensuring robustness to class imbalance, remains an open problem.

Building upon these insights, a critical aspect of training efficiency lies in the composition of batches used during stochastic gradient descent. Classical training paradigms maintain approximately fixed class proportions within each shuffled batch, mirroring the overall class distribution in the training dataset [Buda et al., 2018, Peng et al., 2019]. However, this static approach fails to account for the varying levels of difficulty associated with different classes, which can hinder optimal convergence rates. For example, classes with higher error rates or those that are inherently more challenging may require greater emphasis during training to enhance model performance. While existing approaches address class imbalance by adjusting sample weights or dataset resampling, they do not dynamically change the class-wise composition of batches during training via real-time performance metrics.

This observation motivates the central question of this paper: *Can we adaptively adjust the proportion of classes within batches, across training epochs, to accelerate model convergence?* Addressing this question involves developing approaches that dynamically modify class proportions using real-time performance metrics, directing learning towards underperforming classes. Such batch construction has the potential to enhance convergence rates and training efficiency, especially in scenarios with imbalanced classes or heterogeneous class difficulties [Liu et al., 2008, Ren et al., 2018].

To address these considerations, in this work, we introduce *learn2mix*, a novel training strategy that dynamically modifies class proportions in batches by emphasizing classes with higher instantaneous error rates. In contrast with classical training schemes that utilize fixed class proportions, learn2mix continually adapts these proportions during training via real-time class-wise error rates. This dynamic adjustment facilitates faster convergence and improved performance across various tasks, including classification, regression, and reconstruction. An illustration of the learn2mix training methodology is provided in Figure 1, demonstrating the adaptive class-wise composition of batches.

This paper is organized as follows. In Section 2, we formalize learn2mix, and prove relevant properties. In Section 3, we detail the algorithmic implementation of the learn2mix training methodology. In Section 4, we present empirical evaluations on benchmark datasets, demonstrating the efficacy of learn2mix in accelerating model convergence and enhancing performance. Finally, in Section 5, we summarize our paper. Our main contributions are outlined as follows:

1. We propose *learn2mix*, a novel adaptive training strategy that dynamically adjusts class proportions within batches, utilizing class-wise error rates, to accelerate model convergence.
2. We prove that neural networks trained using *learn2mix* converge faster than those trained using classical approaches when certain properties hold, such that the class proportions converge to a stable distribution proportional to the optimal class-wise error rates.
3. We empirically validate that neural networks trained using *learn2mix* consistently observe accelerated convergence, outperforming existing training methods in terms of convergence speed across classification, regression, and reconstruction tasks.

Related Work. The landscape of neural network training methods comprises various approaches aiming to enhance model performance and training efficiency. Handling class imbalance has been extensively studied, with methods such as importance sampling [Katharopoulos and Fleuret, 2018], oversampling [Chawla et al., 2002], undersampling [Tahir et al., 2012], and class-balanced loss

functions [Lin et al., 2017, Ren et al., 2018] being proposed to mitigate biases towards majority classes. In parallel, curriculum learning [Bengio et al., 2009] and reinforcement learning approaches [Florensa et al., 2017] have introduced methods to facilitate more effective learning trajectories. Meta-learning, or *learn2learn* methodologies [Arnold et al., 2020], including model-agnostic meta-learning (MAML) [Finn et al., 2017], focus on optimizing the learning process itself to enable rapid adaptation to new tasks. Additionally, adaptive data sampling strategies [Liu et al., 2008] and boosting algorithms [Freund and Schapire, 1997] emphasize the significance of prioritizing harder or misclassified examples to improve model robustness. Despite these advances, most existing training methods either adjust sample weights, resample datasets, or modify the sequence of training examples without specifically altering the class proportions within batches in an adaptive manner. Our proposed *learn2mix* strategy distinguishes itself by adapting batch class proportions throughout the training process, targeting classes with higher error rates to accelerate convergence. This approach offers a unified framework by addressing class imbalance through adaptive training principles.

2 Theoretical Results

Consider the random variables $X \in \mathbb{R}^d$ and $Y \in \mathbb{R}^k$, where X denotes the feature vector, Y are the labels, and k is the number of classes. We consider the *original training dataset*, $J = \{(x_j, y_j)\}_{j=1}^N$, where $(x_j, y_j) \overset{\text{i.i.d.}}{\sim} (X, Y)$, $\forall j \in \{1, \ldots, N\}$. The class proportions for this dataset are given by the vector of fixed-proportion mixing parameters, $\tilde{\alpha} = [\tilde{\alpha}_1, \ldots, \tilde{\alpha}_k]^T$, reflecting the distribution of classes. We define $\alpha = [\alpha_1, \ldots, \alpha_k]^T$ as a variable denoting the vector of *mixing parameters*, where $\alpha_i \in [0, 1]$ and $\sum_{i=1}^k \alpha_i = 1$. The value of α determines the class proportions used during training, and can vary depending on the chosen training mechanism. In *classical training*, $\alpha = \alpha^t$ is constant over time and reflects the class proportions within the original training dataset, wherein $\alpha^t = \tilde{\alpha}$, $\forall t \in \mathbb{N}$. In *learn2mix training*, $\alpha = \alpha^t$ is time-varying, and is initialized at time $t = 0$ as $\alpha^0 = \tilde{\alpha}$.

Let $\mathcal{H} \subset \{h : \mathbb{R}^d \to \mathbb{R}^k\}$ be the class of hypothesis functions that model the relationship between X and Y. For our empirical setting, we let $\mathcal{H}$ denote the set of neural networks that have predetermined architectures. We note $\mathcal{H}$ is fully defined by a vector of parameters, $\theta \in \mathbb{R}^m$, where $\mathcal{H} = h_\theta$ denotes a set of parameterized functions. The generalized form of the loss function for classical training and the loss function form under learn2mix training are given below.

Definition 2.1 (Loss Function for Classical Training)**.** *Consider $\tilde{\alpha} \in [0, 1]^k$ as the vector of fixed-proportion mixing parameters, and let $\mathcal{L}(\theta^t) \in \mathbb{R}^k$ denote the vector of class-wise losses at time t. The loss for classical training at time t is given by:*

$$\mathcal{L}(\theta^t, \tilde{\alpha}) = \sum_{i=1}^{k} \tilde{\alpha}_i \mathcal{L}_i(\theta^t) = \tilde{\alpha}^T \mathcal{L}(\theta^t). \tag{1}$$

Definition 2.2 (Loss Function for Learn2Mix Training)**.** *Consider $\alpha^t, \alpha^{t-1} \in [0, 1]^k$ as the vector of mixing parameters at time t and time $t-1$, and let $\mathcal{L}(\theta^t), \mathcal{L}(\theta^{t-1}) \in \mathbb{R}^k$ denote the respective class-wise loss vectors at time t and time $t-1$. Consider $\gamma \in (0, 1)$ as the mixing rate. The loss for learn2mix training at time t is given by the following:*

$$\mathcal{L}(\theta^t, \alpha^t) = \sum_{i=1}^{k} \alpha_i^t \mathcal{L}_i(\theta^t) = (\alpha^t)^T \mathcal{L}(\theta^t), \tag{2}$$

$$\textit{Where: } \alpha^t = \alpha^{t-1} + \gamma \left(\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T \mathcal{L}(\theta^{t-1})} - \alpha^{t-1} \right), \tag{3}$$

We note that the denominator, $\mathbb{1}_k^T \mathcal{L}(\theta^{t-1})$, is the sum of losses across all classes, and dividing by it converts $\mathcal{L}(\theta^{t-1})$ into a probability distribution. We update α^{t-1} by nudging the mixing parameters toward this probability distribution, so classes with higher losses receive a larger share of samples in the next time step. The scalar mixing rate, γ, is a user-defined step size hyperparameter that controls how aggressively α^{t-1} moves. We note that classical training is recovered by setting $\gamma = 0$.

Suppose that $\mathcal{H}$ is sufficiently expressive and can represent the true conditional expectation function, wherein there exists $\theta^* \in \mathbb{R}^m$ with $h_{\theta^*}(X) = \mathbb{E}[Y \mid X]$ almost surely. In the following proposition, we demonstrate that via gradient-based optimization under learn2mix training, the parameters converge to θ^*, with the mixing proportions converging to a stable distribution that reflects the relative difficulty of each class under the optimal parameters.

Proposition 2.3. *Let $\mathcal{L}(\theta^t), \mathcal{L}(\theta^*) \in \mathbb{R}^k$ denote the respective class-wise loss vectors for the model parameters at time t and for the optimal model parameters. Suppose each class-wise loss $\mathcal{L}_i(\theta) \in \mathbb{R}$ is strongly convex in θ, with strong convexity parameter $\mu_i \in \mathbb{R}_{>0}$, $\forall i \in \{1, \ldots, k\}$, and each class-wise loss gradient $\nabla_\theta \mathcal{L}_i(\theta) \in \mathbb{R}^m$ is Lipschitz continuous in θ, having Lipschitz constant $L_i \in \mathbb{R}_{\geq 0}$, $\forall i \in \{1, \ldots, k\}$. Let $\mu^* = \min_{i \in \{1,\ldots,k\}} \mu_i$, $L^* = \max_{i \in \{1,\ldots,k\}} L_i$. Then, if the model parameters at time $t+1$ are obtained via the gradient of the loss for learn2mix training, where:*

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta \mathcal{L}(\theta^t, \alpha^t), \qquad \textit{with:} \quad \eta \in \mathbb{R}_{>0}, \tag{4}$$

It follows that for learning rate, $\eta \in (0, 2/L^*)$, and for mixing rate, $\gamma \in (0, 1)$:

$$\lim_{t \to \infty} \theta^t = \theta^*, \qquad \textit{and:} \quad \lim_{t \to \infty} \alpha^t = \alpha^* = \frac{\mathcal{L}(\theta^*)}{\mathbb{1}_k^T \mathcal{L}(\theta^*)}. \tag{5}$$

The complete proof of Proposition 2.3 is provided in Section A of the Appendix. We now detail the convergence behavior of the learn2mix and classical training strategies, and suppose that $\alpha^{t-1} = \tilde{\alpha}$. We first present Corollary 2.4, which will be used to prove the convergence result in Proposition 2.5. This corollary leverages Lipschitz continuity and strong convexity to bound the loss gradient norm.

Corollary 2.4. *Let $\mathcal{L}(\theta^t) \in \mathbb{R}^k$ denote the class-wise loss vector at time t. Suppose each class-wise loss, $\mathcal{L}_i(\theta) \in \mathbb{R}$, is strongly convex in θ, with strong convexity parameter $\mu_i \in \mathbb{R}_{>0}$, $\forall i \in \{1, \ldots, k\}$, and suppose each class-wise loss gradient $\nabla_\theta \mathcal{L}_i(\theta) \in \mathbb{R}^m$ is Lipschitz continuous in θ with Lipschitz constant $L_i \in \mathbb{R}_{\geq 0}$, $\forall i \in \{1, \ldots, k\}$. Let $\mu^* = \min_{i \in \{1,\ldots,k\}} \mu_i$, $L^* = \max_{i \in \{1,\ldots,k\}} L_i$. Then, the following condition and inequality hold, $\forall \alpha \in [0, 1]^k$ where $\sum_{i=1}^k \alpha_i = 1$:*

$$\frac{\mu^*}{2} \|\theta^t - \theta^*\| \leq \|\nabla_\theta \mathcal{L}(\theta^t, \alpha)\| \leq L^* \|\theta^t - \theta^*\|, \tag{6}$$

$$\textit{Wherein:} \ \|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\| + \|\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\| \leq 2L^* \|\theta^t - \theta^*\|. \tag{7}$$

The proof of Corollary 2.4 is provided in Section A of the Appendix — we note that the inequality in Eq. (7) relates the loss gradient norm under classical training with that under learn2mix training. We now present Proposition 2.5, which demonstrates that under the condition expressed in Eq. (8), updates obtained via the gradient of the loss for learn2mix training bring the model parameters closer to the optimal solution than those obtained via the gradient of the loss for classical training.

Proposition 2.5. *Let $\mathcal{L}(\theta^t), \mathcal{L}(\theta^*) \in \mathbb{R}^k$ denote the respective class-wise loss vectors for the model parameters at time t and for the optimal model parameters. Suppose each class-wise loss, $\mathcal{L}_i(\theta) \in \mathbb{R}$ is strongly convex in θ with strong convexity parameter $\mu_i \in \mathbb{R}_{>0}$, $\forall i \in \{1, \ldots, k\}$, and each class-wise loss gradient $\nabla_\theta \mathcal{L}_i(\theta) \in \mathbb{R}^m$ is Lipschitz continuous in θ, having Lipschitz constant $L_i \in \mathbb{R}_{\geq 0}$, $\forall i \in \{1, \ldots, k\}$. Moreover, suppose the loss gradient $\nabla_\theta \mathcal{L}(\theta, \alpha) \in \mathbb{R}^m$ is Lipschitz continuous in α, having Lipschitz constant $L_\alpha \in \mathbb{R}_{\geq 0}$, and let $\mu^* = \min_{i \in \{1,\ldots,k\}} \mu_i$, $L^* = \max_{i \in \{1,\ldots,k\}} L_i$. Then, if and only if the following condition holds:*

$$\Big[\Big(\frac{\mu^*}{2} - L^*\Big)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\Big]\Big[\|\theta^t - \theta^*\| - (\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\Big] > 0, \tag{8}$$

It follows that for every learning rate, $\eta > 0$, and for every mixing rate, $\gamma \in (0, \beta]$:

$$\left\|\left(\theta^t - \eta \nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\right) - \theta^*\right\| \leq \left\|\left(\theta^t - \eta \nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\right) - \theta^*\right\|. \tag{9}$$

The complete formula for β can be found in Section A of the Appendix.

The complete proof of Proposition 2.5 is provided in Section A of the Appendix.

3 Algorithm

In this section, we outline our approach for training neural networks using learn2mix. The learn2mix mechanism comprises a bilevel optimization procedure, where we first update the neural network parameters, θ^t, before updating the mixing parameters, α^t, using the vector of class-wise losses, $\mathcal{L}(\theta^t)$. Considering the original training dataset, J, define $J_i = \{(x_j, y_j)\}_{j=1}^{\tilde{\alpha}_i N}$, $\forall i \in \{1, \ldots, k\}$ as each class-specific training dataset, with $J = \bigcup_{i=1}^k J_i$. These k class-specific training datasets are leveraged to speed up batch formation under learn2mix. We consider neural network training using

Algorithm 1: Neural Network Training Via Learn2Mix

Input: J (Original Training Dataset), θ (Initial NN Parameters), $\tilde{\alpha}$ (Initial Mixing Parameters), η (Learning Rate), γ (Mixing Rate), M (Batch Size), P (No. of Batches), E (Epochs)

Output: θ (Trained NN Parameters)

for $i = 1, 2, \dots k$ **do**
  $J_i \leftarrow \{(x_j, y_j)\}_{j=1}^{\alpha_i N}$ (Initialize class-specific training datasets)
  $\alpha_i \leftarrow \tilde{\alpha}_i$ (Initialize time-varying mixing parameters)
for $epoch = 1, 2, \dots, E$ **do**
  for $i = 1, 2, \dots, k$ **do**
    $J_i \leftarrow \texttt{Shuffle}(J_i)$ (Randomly shuffle each class-specific training dataset)
  for $p = 1, 2, \dots, P$ **do**
    for $i = 1, 2, \dots, k$ **do**
      $S_i^p \leftarrow \texttt{Sample}(J_i, \alpha_i M)$ (Select $\alpha_i M$ distinct examples from J_i)
    $S^p \leftarrow \biguplus_{i=1}^{k} S_i^p$ (Aggregate to form batch S^p)
    $\mathcal{L}^p(\theta, \alpha) \leftarrow \frac{1}{M} \sum_{(x_j, y_j) \in S^p} \ell(h_\theta(x_j), y_j)$ (Compute loss on batch S^p)
  $\mathcal{L}(\theta, \alpha) \leftarrow \frac{1}{P} \sum_{p=1}^{P} \mathcal{L}^p(\theta, \alpha)$ (Obtain total loss)
  $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}(\theta, \alpha)$ (Update model parameters, θ)
  for $i = 1, 2, \dots, k$ **do**
    $\mathcal{L}_i(\theta) \leftarrow \frac{1}{P} \sum_{p=1}^{P} \frac{1}{\alpha_i M} \sum_{(x_j, y_j) \in S_i^p} \ell(h_\theta(x_j), y_j)$ (Compute loss for class i)
  $\alpha \leftarrow \texttt{Update\_Mixing\_Params}(\alpha, \mathcal{L}(\theta), \gamma)$
return θ

batched stochastic gradient descent, where for training epoch, t, the empirical loss is computed over $P = N/M$ total batches, where $M \in \mathbb{Z}^+$ is the batch size. Each batch is formed by sampling $\alpha_i^t M$ distinct examples from the ith class-specific training dataset, denoted as $S_i^p \subseteq J_i$, for $S^p = \biguplus_{i=1}^{k} S_i^p$, where $\biguplus$ is the set union operator that preserves duplicate elements. For learn2mix training, the class-wise errors, $\mathcal{L}_i(\theta^t), \forall i \in \{1, \dots, k\}$, at training epoch t are empirically computed as:

$$\mathcal{L}_i(\theta^t) = \frac{1}{P} \sum_{p=1}^{P} \left[\frac{1}{\alpha_i^t M} \sum_{(x_j, y_j) \in S_i^p} \ell(h_{\theta^t}(x_j), y_j) \right], \tag{10}$$

Where $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ is a bounded per-sample loss function and computes the error between the model prediction, $h_{\theta^t}(x_j)$, and the true label, y_j. Accordingly, the overall empirical loss at training epoch, t, under the learn2mix training mechanism is given by:

$$\mathcal{L}(\theta^t, \alpha^t) = \sum_{i=1}^{k} \alpha_i^t \mathcal{L}_i(\theta^t) = \sum_{i=1}^{k} \alpha_i^t \left[\frac{1}{P} \sum_{p=1}^{P} \left[\frac{1}{\alpha_i^t M} \sum_{(x_j, y_j) \in S_i^p} \ell(h_{\theta^t}(x_j), y_j) \right] \right]. \tag{11}$$

Utilizing the empirical loss formulation from Eq. (11), we now detail the algorithmic implementation of the learn2mix training methodology on a per-sample basis, for consistency with the mathematical preliminaries in Section 2. We note that the batch processing equivalent of this procedure is a trivial extension to the domain of matrices, and was used to generate the empirical results from Section 4. Algorithm 1 outlines the primary training loop, where for each epoch, the class-specific datasets, J_i, are shuffled. Within each epoch, we iterate over the P total batches, forming each batch by choosing $\alpha_i M$ examples from every J_i. The empirical loss within each batch is computed and aggregated to obtain the overall loss, $\mathcal{L}(\theta, \alpha)$, which is then used to update the neural network parameters through gradient descent. Lastly, the vector of class-wise losses, $\mathcal{L}(\theta)$, is calculated to inform the adjustment of the mixing parameters, α, using Algorithm 2.

Algorithm 2 outlines the method for adjusting class proportions using the mixing parameters, α, based on the computed class-wise losses. For each class, $i \in \{1, \dots, k\}$, we first calculate the normalized loss L_i by dividing the class-specific loss $\mathcal{L}_i(\theta)$ by the total cumulative loss summed over all classes. Each mixing parameter, α_i, is then updated incrementally towards this normalized loss value L_i.

Algorithm 2: Updating Mixing Parameters Via Learn2Mix

Input: α (Previous Mixing Parameters), $\mathcal{L}(\theta)$ (Class-wise loss vector), γ (Mixing Rate)
Output: α (Updated Mixing Parameters)

for $i = 1, 2, \ldots, k$ **do**
  $L_i \leftarrow \frac{\mathcal{L}_i(\theta)}{\sum_{j=1}^{k} \mathcal{L}_j(\theta)}$ (Compute normalized losses)
  $\alpha_i \leftarrow \alpha_i + \gamma\,(L_i - \alpha_i)$ (Update mixing parameters)
return α

The magnitude of the update step is controlled by the mixing rate, γ, determining how quickly the proportions adapt. Thus, classes exhibiting higher relative losses are progressively given greater emphasis in subsequent training epochs, ensuring a balanced reduction of errors across all classes.

Finally, we recall that during the batch construction phase, for each class, $i \in \{1, \ldots, k\}$, we select $\alpha_i M$ examples from each J_i to form the subset $S_i^p \subseteq J_i$. Given the dynamic nature of the mixing parameters, α, it is possible that this cumulative selection across batches may exhaust all the samples within a particular J_i before the epoch concludes. To address this, we incorporate a cyclic selection mechanism. Formally, we define an index τ_i^p, $\forall i \in \{1, \ldots, k\}$ and $p \in \{1, \ldots, P\}$, such that:

$$\tau_i^p = \left(\tau_i^{p-1} + \alpha_i M\right) \mod \tilde{\alpha}_i N, \tag{12}$$

Where $\tau_i^0 = 0, \forall i \in \{1, \ldots, k\}$. Accordingly, when selecting S_i^p, if $\tau_i^{p-1} + \alpha_i M > \tilde{\alpha}_i N$, we wrap around to the beginning of J_i, effectively resetting the selection index, τ_i^p — this ensures that every example in J_i is selected uniformly and repeatedly as needed throughout the training process. Thus, the selection procedure to construct S_i^p is defined as:

$$S_i^p = \biguplus_{w=0}^{\alpha_i M - 1} J_i \left[(\tau_i^{p-1} + w) \mod \tilde{\alpha}_i N \right]. \tag{13}$$

This cyclic selection procedure ensures that the required number of samples, $\alpha_i M$, for each class in every batch is maintained, even as α_i is adaptively updated across epochs.

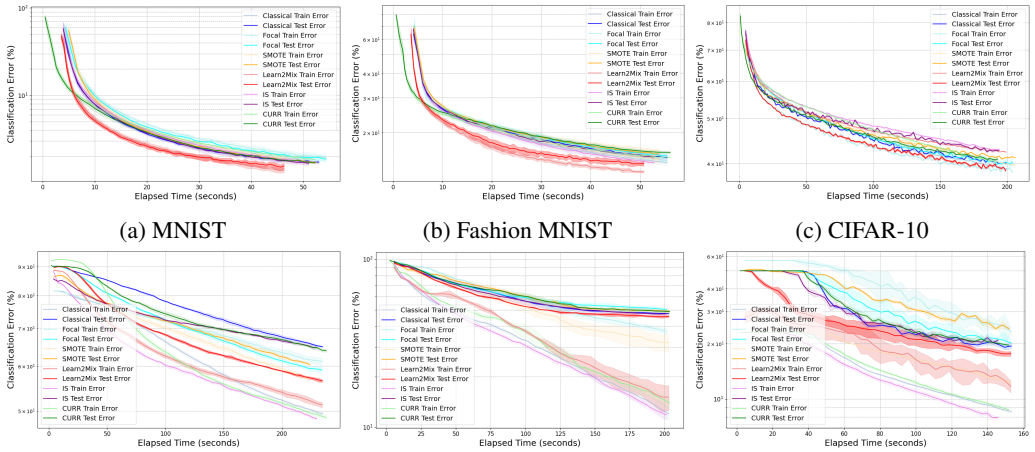
4 Empirical Results

In this section, we present our empirical results on classification, regression, and image reconstruction tasks, across both benchmark and modified class imbalanced datasets. We first present the classification results on three benchmark datasets (MNIST [Deng, 2012], Fashion-MNIST [Xiao et al., 2017], CIFAR-10 [Krizhevsky et al., 2009]), and three standard datasets with manually imbalanced classes (Imagenette [Howard, 2020], CIFAR-100 [Krizhevsky et al., 2009], and IMDB [Maas et al., 2011]). We note that for the imbalanced case, we only introduce the manual class-imbalancing to the training dataset, J, where the test dataset, $K = \{(x_j, y_j)\}_{j=1}^{N_{\text{test}}}$, is not changed. This choice ensures that the generalization performance of the network is benchmarked in a class-balanced setting. Next, for the regression task, we study two benchmark datasets with manually imbalanced classes (Wine Quality [Cortez et al., 2009], and California Housing [Géron, 2022]), and a synthetic mean estimation task, where the manual class-imbalancing parallels that of the classification case. Finally, we reconsider the MNIST, Fashion MNIST and CIFAR-10 datasets for image reconstruction, again with manual class-imbalancing. The comprehensive description of these datasets and class-imbalancing strategies is in Section C of the Appendix. For further performance verification, we include ablation studies on architecture, optimizer, batch size, learning rate, and worst-class error in Section B of the Appendix.

The intuition behind the application of learn2mix to regression and reconstruction tasks stems from its ability to adaptively handle different data distributions. For regression tasks with a categorical variable taking k distinct values, the samples from J that correspond to each of the k values, can be aggregated to obtain each class-specific training dataset, J_i. Here, each J_i denotes a distinct underlying data distribution. As in the classification case, learn2mix will adaptively adjust the class-specific dataset proportions during training. For image reconstruction, we can similarly treat the k distinct classes being reconstructed as the values taken by a categorical variable, paralleling the regression context. This formulation supports the adaptive adjustment of class proportions under learn2mix training.

Table 1: Test accuracies for learn2mix (L2M), classical (CL), FCL, SMOTE, IS, CURR training.

Elapsed Time: $\underline{t} = 0.25\underline{E}$ s						
Dataset	**MNIST**	**Fsh. MNIST**	**CIFAR-10**	**Imagenette**	**CIFAR-100**	**IMDB**
Acc (L2M)	95.42±0.28	77.62±0.69	51.34±0.13	24.12±0.46	30.03±1.30	70.28±1.66
Acc (CL)	93.14±0.47	74.13±0.73	49.26±0.15	15.55±0.13	23.23±1.99	50.13±0.15
Acc (FCL)	91.32±0.57	74.08±0.75	47.90±0.22	20.11±0.37	27.15±1.13	50.30±0.67
Acc (SMOTE)	92.41±0.71	73.67±0.61	47.76±0.15	23.19±0.46	23.93±2.35	50.94±0.05
Acc (IS)	92.44±0.63	74.23±0.29	47.40±0.51	23.10±0.39	27.97±0.67	58.48±0.64
Acc (CURR)	93.30±0.54	75.06±0.63	49.11±0.25	18.82±0.37	27.15±0.10	50.02±0.04
Elapsed Time: $\underline{t} = 0.5\underline{E}$ s						
Dataset	**MNIST**	**Fsh. MNIST**	**CIFAR-10**	**Imagenette**	**CIFAR-100**	**IMDB**
Acc (L2M)	97.61±0.15	83.16±0.87	55.84±0.19	33.64±0.42	46.80±0.54	76.02±2.77
Acc (CL)	96.74±0.10	79.75±0.83	54.50±0.34	23.63±0.33	43.00±0.73	74.99±0.57
Acc (FCL)	95.92±0.13	78.94±0.82	54.15±0.06	29.44±0.43	40.26±0.55	68.30±3.21
Acc (SMOTE)	96.51±0.16	79.17±0.50	53.72±0.16	28.90±0.43	39.10±1.63	62.72±0.54
Acc (IS)	96.60±0.25	79.65±0.38	52.56±0.39	28.52±0.32	42.61±2.61	74.00±0.81
Acc (CURR)	96.53±0.16	79.08±0.58	53.49±0.33	27.26±0.79	39.48±2.22	71.68±0.55
Elapsed Time: $\underline{t} = \underline{E}$ s						
Dataset	**MNIST**	**Fsh. MNIST**	**CIFAR-10**	**Imagenette**	**CIFAR-100**	**IMDB**
Acc (L2M)	98.46±0.14	85.85±0.47	60.49±0.26	42.95±0.33	54.50±0.73	82.38±0.59
Acc (CL)	98.14±0.14	84.23±0.60	59.62±0.16	34.53±0.33	52.30±0.36	80.84±0.71
Acc (FCL)	97.86±0.08	83.68±0.61	59.37±0.64	40.60±0.71	49.33±0.97	79.09±2.58
Acc (SMOTE)	98.09±0.07	83.57±1.06	58.46±0.15	39.59±0.29	50.63±1.02	74.64±1.28
Acc (IS)	98.14±0.14	84.33±0.29	57.44±0.42	35.33±0.43	52.83±0.34	79.08±0.57
Acc (CURR)	98.13±0.05	83.32±0.43	59.15±0.42	35.26±0.48	50.88±0.79	80.04±0.25



(a) MNIST (b) Fashion MNIST (c) CIFAR-10

(d) Imagenette (e) CIFAR-100 (f) IMDB

Figure 2: Comparing model classification errors for learn2mix, classical, FCL, SMOTE, IS, and CURR training. The x-axis is the elapsed [training] time, while the y-axis is the classification error.

For the evaluations that follow, all training was performed on an NVIDIA GEForce RTX 3090 GPU. To ensure a fair comparison between learn2mix and classical training, we utilize the same learning rate, η, and neural network architecture with initialized parameters, θ, across all experiments for a given dataset. Additionally, we train each neural network through learn2mix (with mixing rate γ) and classical training for $\underline{E}$ seconds (or E epochs), where $\underline{E}$ is dataset dependent [1]. In classification tasks, we also benchmark learn2mix and classical training versus 'FCL training', 'SMOTE training', 'IS training', and 'CURR training' (training using focal loss [Lin et al., 2017], SMOTE oversampling [Chawla et al., 2002], importance sampling [Katharopoulos and Fleuret, 2018], and curriculum learning [Hacohen and Weinshall, 2019] — see Sections D.3, D.4, D.5, and D.6 of the Appendix). The complete list of model architectures and hyperparameters is in Section D of the Appendix.

[1] Practically, we observe that choosing $\gamma \in [0.01, 0.1]$ improves performance (see Section B of the Appendix).

4.1 Classification Tasks

As illustrated in Table 1 and Figure 2, we observe a consistent trend across all considered classification benchmarks, whereby neural networks trained using learn2mix converge faster than their classically-trained, FCL-trained, SMOTE-trained, IS-trained, and CURR-trained counterparts. We first consider MNIST, and train LeNet-5 [Lecun et al., 1998] via the Adam optimizer [Kingma and Ba, 2014] and Cross Entropy Loss for $\underline{E} = 50$ s, leveraging learn2mix, classical, FCL, SMOTE, IS, and CURR training. We see that the learn2mix-trained CNN converges faster, eclipsing a test accuracy of 98% within 30 s, whereas the remaining CNNs achieve this test accuracy after 40 s. We next consider the more challenging Fashion MNIST dataset, and train Large LeNet-5 for $\underline{E} = 50$ s with the Adam optimizer and Cross Entropy Loss, leveraging learn2mix, classical, FCL, SMOTE, IS, and CURR training. Paralleling MNIST, we observe that the learn2mix-trained CNN converges faster, yielding a test accuracy of 83% within 20 s, whereas the other CNNs achieve this test accuracy after 33 s. The last class-balanced benchmark dataset we investigate is CIFAR-10, which offers a greater challenge than MNIST and Fashion MNIST. We train Large LeNet for $\underline{E} = 200$ s using the Adam optimizer and Cross Entropy Loss, utilizing learn2mix, classical, FCL, SMOTE, IS, and CURR training. We observe that the learn2mix-trained CNN achieves faster convergence, yielding a test accuracy of 60% after 170 s, whereas the remaining CNNs exceed this test accuracy after 200 s. Cumulatively, these evaluations demonstrate the efficacy of learn2mix training in settings with balanced classes, wherein the adaptive adjustment of class proportions accelerates convergence.

We now consider several class-imbalanced training datasets. We first benchmark Imagenette, which comprises a subset of 10 classes from ImageNet [Deng et al., 2009], and modify the training dataset so the number of samples from each class, $i \in \{1, \ldots, k\}$, in J decreases linearly. We train ResNet-18 [He et al., 2016] with the Adam optimizer and Cross Entropy Loss for $\underline{E} = 230$ s, using learn2mix, classical, FCL, SMOTE, IS, and CURR training. We see the learn2mix-trained ResNet-18 converges faster, achieving a test accuracy of 40% after 185 s, whereas only the FCL-trained model achieves this test accuracy after 230 s. We now consider CIFAR-100, and modify the training dataset so the number of samples from each class, $i \in \{1, \ldots, k\}$, in J decreases logarithmically. We train MobileNet-V3 Small [Howard et al., 2019] for $\underline{E} = 200$ s leveraging the Adam optimizer and Cross Entropy Loss, using learn2mix, classical, FCL, SMOTE, IS, and CURR training. We see that the learn2mix-trained MobileNet-V3 Small model converges faster, achieving a test accuracy of 50% within 120 s, whereas the other models exceed this test accuracy after 140 s. As the $k = 100$ mixing parameters are a small fraction of the total model parameters, this overhead is negligible. For IMDB, we modify the training dataset so the positive class keeps 30% of its original samples. We train a transformer for $\underline{E} = 150$ s with the Adam optimizer and Cross Entropy Loss, using learn2mix, classical, FCL, SMOTE, IS, and CURR training. We see the learn2mix-trained transformer converges faster, reaching a test accuracy of 70% within 35 s, whereas the other models exceed this test accuracy after 60 s.

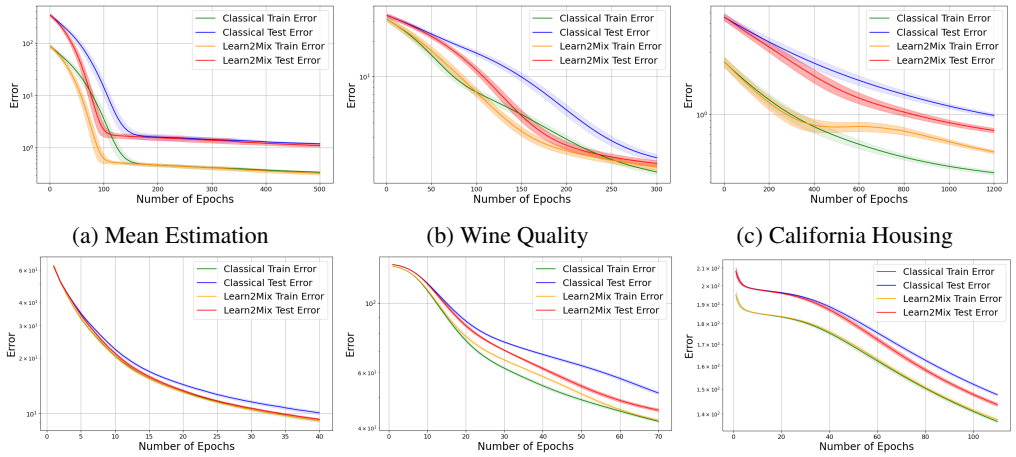
In the above evaluations, we see learn2mix not only accelerates convergence, but also has a tighter alignment between training and test errors versus classical training. This correspondence indicates reduced overfitting, as learn2mix inherently adjusts class proportions based on class-specific error rates, L_i. By biasing the optimization procedure away from the original class distribution and towards L_i, learn2mix achieves improved generalization. We note this property is not unique to classification and also applies to regression and reconstruction (see Sections 4.2 and 4.3).

4.2 Regression Tasks

As illustrated in Table 2 and Figure 3, we observe that learn2mix maintains accelerated convergence in the regression context, wherein all the considered datasets are class imbalanced. We first consider the synthetic Mean Estimation dataset, which comprises sets of samples gathered from $k = 4$ unique distributions and their associated means. Using the Adam optimizer and Mean Squared Error (MSE) Loss, we train a fully connected network for $E = 500$ epochs on Mean Estimation via learn2mix and classical training. We see that the learn2mix-trained neural network converges rapidly, achieving a test error below 2.0 after 100 epochs, at which point the classically-trained network has a test error of 13.0. For the Wine Quality dataset, we modify the training dataset so the white wine class has 10% of its original samples. Using the Adam optimizer and MSE Loss, we train a fully connected network for $E = 300$ epochs on Wine Quality via learn2mix and classical training. We observe that the learn2mix-trained neural network yields faster convergence, achieving a test error below 2.5 after 200 epochs, at which point the classically-trained network has a test error of 5.0. Finally, on the

Table 2: Test mean squared error for learn2mix (L2M) and classical (CL) training.

Dataset	Epoch $t = 0.25E$		Epoch $t = 0.5E$		Epoch $t = E$	
	Err (L2M)	**Err (CL)**	**Err (L2M)**	**Err (CL)**	**Err (L2M)**	**Err (CL)**
Mean Estim.	1.81 ± 0.84	6.51 ± 1.52	1.45 ± 0.26	1.52 ± 0.27	1.07 ± 0.09	1.17 ± 0.06
Wine Quality	17.7 ± 1.64	19.8 ± 1.51	4.26 ± 1.55	9.72 ± 1.94	1.75 ± 0.21	2.03 ± 0.18
Cali. Housing	2.52 ± 0.68	2.95 ± 0.67	1.33 ± 0.32	1.82 ± 0.39	0.77 ± 0.08	0.99 ± 0.10
MNIST	19.6 ± 0.81	20.8 ± 0.93	12.9 ± 0.39	14.0 ± 0.52	9.31 ± 0.24	10.1 ± 0.56
Fsh. MNIST	89.3 ± 2.63	91.9 ± 2.37	65.1 ± 1.21	70.9 ± 1.28	45.5 ± 1.21	51.6 ± 1.60
CIFAR-10	193 ± 1.23	194 ± 1.98	175 ± 2.85	179 ± 3.87	144 ± 1.71	148 ± 1.37



(a) Mean Estimation (b) Wine Quality (c) California Housing

(d) MNIST (e) Fashion MNIST (f) CIFAR-10

Figure 3: Comparing model performance errors for classical training and learn2mix training. The x-axis is the number of elapsed training epochs, while the y-axis is the mean squared error (MSE).

California Housing dataset, we modify the training dataset such that three of the classes have 5% of their original samples. Using the Adam optimizer and MSE Loss, we train a fully connected network for $E = 1200$ epochs on California Housing via learn2mix and classical training. We again see that the learn2mix-trained network converges faster, achieving a test error below 0.8 after 1200 epochs, while the classically-trained network has a test error of 0.99. These empirical evaluations support our previous intuition pertaining to the extension of learn2mix to imbalanced regression settings.

4.3 Image Reconstruction Tasks

Per Table 2 and Figure 3, we note that the class-imbalanced image reconstruction tasks also observe faster convergence using learn2mix. For the MNIST case, we modify the training dataset such that half of the classes retain 20% of their original samples. Leveraging the Adam optimizer and MSE Loss, we train an autoencoder for $E = 40$ epochs on MNIST using learn2mix and classical training. We observe that the learn2mix-trained autoencoder exhibits improved convergence, achieving a test error less than 1.0 after 35 epochs, which the classically-trained autoencoder achieves after 40 epochs. Correspondingly, for Fashion MNIST, we modify the training dataset such that half of the classes retain 20% of their original samples (paralleling MNIST). Using the Adam optimizer and MSE Loss, we train an autoencoder for $E = 70$ epochs on Fashion MNIST, leveraging learn2mix and classical training. We observe that the learn2mix-trained autoencoder converges faster, achieving a test error below 54.0 after 50 epochs, which the classically-trained autoencoder achieves after 65 epochs. We also consider CIFAR-10, wherein we modify the training dataset such that all but two classes retain 20% of their original samples. Utilizing the Adam optimizer and MSE Loss, we train an autoencoder for $E = 110$ epochs on CIFAR-10, leveraging learn2mix and classical training. We observe that the learn2mix-trained autoencoder also converges faster and achieves a test error below 148.0 after 100 epochs, which the classically-trained autoencoder achieves after 110 epochs.

5 Conclusion

In this work, we presented *learn2mix*, a new training strategy that adaptively modifies class proportions in batches via real-time class-wise error rates, accelerating model convergence. We formalized the learn2mix mechanism through a bilevel optimization framework, and outlined its theoretical advantages in aligning class proportions with optimal error rates. Empirical evaluations across classification, regression, and reconstruction tasks on both balanced and imbalanced datasets confirmed that learn2mix not only accelerates convergence compared to classical training methods, but also reduces overfitting in the presence of class-imbalances. Accordingly, models trained with learn2mix achieved improved performance in constrained training regimes. Our findings underscore the potential of dynamic batch composition strategies in optimizing neural network training, paving the way for more efficient and robust machine learning models in resource-constrained environments.

Acknowledgments and Disclosure of Funding

Shyam Venkatasubramanian and Vahid Tarokh were supported in part by the Air Force Office of Scientific Research under award FA9550-21-1-0235.

References

Sébastien M. R. Arnold, Praateek Mahajan, Debajyoti Datta, Ian Bunner, and Konstantinos Saitas Zarkias. learn2learn: A library for meta-learning research, 2020.

Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. In *2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 776–780. IEEE, 2020.

Yoshua Bengio, Jean Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th Annual International Conference on Machine Learning*, pages 41–48. ACM, 2009.

Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

Mateusz Buda, Atsuto Maki, and Maciej A Mazurowski. A systematic study of the class imbalance problem in convolutional neural networks. *Neural Networks*, 106:249–259, 2018.

Nitesh V Chawla, Kevin W Bowyer, Lawrence O Hall, and W Philip Kegelmeyer. Smote: Synthetic minority over-sampling technique. In *Proceedings of the 2002 Joint Conference on IEEE International Conference on Knowledge Discovery and Data Mining and IEEE European Conference on Machine Learning*, pages 878–884. IEEE, 2002.

Paulo Cortez, António Cerdeira, Fernando Almeida, Telmo Matos, and José Reis. Modeling wine preferences by data mining from physicochemical properties. *Decision support systems*, 47(4): 547–553, 2009.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

Li Deng. The mnist database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142, 2012.

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.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR, 06–11 Aug 2017.

Carlos Florensa, Yoshua Bengio, and Aaron Courville. Automatic goal generation for reinforcement learning agents. In *International Conference on Learning Representations*, 2017.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Ari Kyrola, Joshua Tulloch, Yangqing Jia, and Kaiming He. Accurate, large minibatch sgd: Training imagenet in 1 hour. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1206–1214, 2017.

Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

Alex Graves, Marc G Bellemare, Jacob Menick, Remi Munos, and Koray Kavukcuoglu. Automated curriculum learning for neural networks. In *international conference on machine learning*, pages 1311–1320. Pmlr, 2017.

Guy Hacohen and Daphna Weinshall. On the power of curriculum learning in training deep networks. In *International conference on machine learning*, pages 2535–2544. PMLR, 2019.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, 2019.

Jeremy Howard. Imagenette. `https://github.com/fastai/imagenette`, 2020.

Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456. PMLR, 2015.

Mathias Johansson and Emma Lindberg. Importance sampling in deep learning: A broad investigation on importance sampling performance, 2022.

Angelos Katharopoulos and François Fleuret. Not all samples are created equal: Deep learning with importance sampling. In *International conference on machine learning*, pages 2525–2534. PMLR, 2018.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2980–2988, 2017.

Xu-Ying Liu, Jianxin Wu, and Zhi-Hua Zhou. Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39(2): 539–550, 2008.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 142–150, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.

Minlong Peng, Qi Zhang, Xiaoyu Xing, Tao Gui, Xuanjing Huang, Yu-Gang Jiang, Keyu Ding, and Zhigang Chen. Trainable undersampling for class-imbalance learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 4707–4714, 2019.

Lutz Prechelt. Early stopping - but when? In *Neural Networks: Tricks of the trade*, pages 55–69. Springer, 1998.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8), 2019.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning. In *International conference on machine learning*, pages 4334–4343. PMLR, 2018.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, 1986.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of Big Data*, 6(1):60, 2019.

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

Leslie N. Smith. Cyclical learning rates for training neural networks. In *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 464–472. IEEE, 2017.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Muhammad Atif Tahir, Josef Kittler, and Fei Yan. Inverse random under sampling for class imbalance problem and its application to multi-label classification. *Pattern Recognition*, 45(10):3738–3750, 2012.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

Vladimir Vapnik and Léon Bottou. Local algorithms for pattern recognition and dependencies estimation. *Neural Computation*, 5(6):893–909, 1993.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008, 2017.

Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):4555–4576, 2021.

Xueyuan Wang and M Cenk Gursoy. Resilient path planning for uavs in data collection under adversarial attacks. *IEEE Transactions on Information Forensics and Security*, 18, 2023.

Yiru Wang, Weihao Gan, Jie Yang, Wei Wu, and Junjie Yan. Dynamic curriculum learning for imbalanced data classification. In *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5016–5025, 2019. doi: 10.1109/ICCV.2019.00512.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Appendix

A Proofs of the Theoretical Results

In this section, we present the proofs of the theoretical results outlined in the main text.

Proposition 2.3. *Let $\mathcal{L}(\theta^t), \mathcal{L}(\theta^*) \in \mathbb{R}^k$ denote the class-wise loss vectors for the model parameters at time t and the optimal model parameters. Suppose each class-wise loss $\mathcal{L}_i(\theta) \in \mathbb{R}$ is strongly convex in θ, with strong convexity parameter $\mu_i \in \mathbb{R}_{>0}$, $\forall i \in \{1, \ldots, k\}$, and each class-wise loss gradient $\nabla_\theta \mathcal{L}_i(\theta) \in \mathbb{R}^m$ is Lipschitz continuous in θ, having Lipschitz constant $L_i \in \mathbb{R}_{\geq 0}$, $\forall i \in \{1, \ldots, k\}$. Let $\mu^* = \min_{i \in \{1,\ldots,k\}} \mu_i$, $L^* = \max_{i \in \{1,\ldots,k\}} L_i$. Then, if the model parameters at time $t+1$ are obtained via the gradient of the loss for learn2mix training, where:*

$$\theta^{t+1} = \theta^t - \eta \nabla_\theta \mathcal{L}(\theta^t, \alpha^t), \qquad \textit{with:} \quad \eta \in \mathbb{R}_{>0}, \tag{14}$$

It follows that for learning rate, $\eta \in (0, 2/L^*)$, and for mixing rate, $\gamma \in (0, 1)$:

$$\lim_{t \to \infty} \theta^t = \theta^*, \qquad \textit{and:} \quad \lim_{t \to \infty} \alpha^t = \alpha^* = \frac{\mathcal{L}(\theta^*)}{\mathbb{1}_k^T \mathcal{L}(\theta^*)}. \tag{15}$$

Proof. We begin by recalling that $\mathcal{L}_i(\theta)$ is strongly convex in θ with strong convexity parameter μ_i, $\forall i \in \{1, \ldots, k\}$. Accordingly, $\forall \alpha \in [0, 1]^k$, with $\sum_{i=1}^k \alpha_i = 1$, the loss function $\mathcal{L}(\theta, \alpha)$ is strongly convex in θ with parameter, $\mu' \in \mathbb{R}_{>0}$, which is lower bounded by $\mu^* \in \mathbb{R}_{>0}$, as per Eq. (16).

$$\mu' \geq \mu^* > 0, \qquad \text{where:} \quad \mu^* = \min_{i \in \{1,\ldots,k\}} \mu_i, \qquad \text{and:} \quad \mu' = \sum\nolimits_{i=1}^k \alpha_i \mu_i. \tag{16}$$

We note that this lower bound on the strong convexity parameter, $\mu' \geq \mu^*$, holds independently of α. Now, recall that $\nabla_\theta \mathcal{L}_i(\theta)$, is Lipschitz continuous in θ with Lipschitz constant L_i, $\forall i \in \{1, \ldots, k\}$. Accordingly, $\forall \alpha \in [0, 1]^k$, where $\sum_{i=1}^k \alpha_i = 1$, the loss gradient $\nabla_\theta \mathcal{L}(\theta, \alpha)$ is Lipschitz continuous in θ with Lipschitz constant, $L' \in \mathbb{R}_{\geq 0}$, which is upper bounded by $L^* \in \mathbb{R}_{\geq 0}$, as per Eq. (17).

$$L^* \geq L' \geq 0, \qquad \text{where:} \quad L^* = \max_{i \in \{1,\ldots,k\}} L_i, \qquad \text{and:} \quad L' = \sum\nolimits_{i=1}^k \alpha_i L_i. \tag{17}$$

We affirm that this upper bound on the Lipschitz constant, $L' \leq L^*$, holds independently of α. Now, suppose that $\alpha = \alpha^t$, where $\mathcal{L}(\theta, \alpha^t)$ is strongly convex in θ with parameter $\mu' \geq \mu^*$ and $\nabla_\theta \mathcal{L}(\theta, \alpha^t)$ is Lipschitz continuous in θ with constant $L' \leq L^*$. Let $\rho = \max\{|1 - \eta\mu^*|, |1 - \eta L^*|\}$. By the gradient descent convergence theorem, for learning rate, $\eta \in (0, 2/L^*)$, it follows that:

$$\lim_{t \to \infty} \|\theta^t - \theta^*\| \leq \lim_{t \to \infty} \rho^t \|\theta^0 - \theta^*\| = \|\theta^0 - \theta^*\| \lim_{t \to \infty} \rho^t = 0. \tag{18}$$

Therefore, $\lim_{t \to \infty} \theta^t = \theta^*$. Let $\beta^{t-1} = \mathcal{L}(\theta^{t-1}) / \left[\mathbb{1}_k^T \mathcal{L}(\theta^{t-1})\right]$, wherein $\beta^{t-1} \in [0, 1]^k$. Unrolling the recurrence relation from Eq. (5) and expressing it in terms of β^{t-1}, we obtain:

$$\alpha^t = (1 - \gamma)^t \alpha^0 + \gamma \sum_{l=0}^{t-1} (1 - \gamma)^{t-1-l} \beta^l. \tag{19}$$

Taking the limit and re-indexing the summation using $n = t - 1 - l$ and $l = t - 1 - n$, we obtain:

$$\lim_{t \to \infty} \alpha^t = \lim_{t \to \infty} \left[(1 - \gamma)^t \alpha^0\right] + \lim_{t \to \infty} \left[\gamma \sum_{n=0}^{t-1} (1 - \gamma)^n \beta^{t-1-n}\right] \tag{20}$$

$$= \mathbf{0}_k + \gamma \lim_{t \to \infty} \left[\sum_{n=0}^{t-1} (1 - \gamma)^n \beta^{t-1-n}\right]. \tag{21}$$

We proceed with the steps to invoke the dominated convergence theorem. We note that for fixed n:

$$\lim_{t \to \infty} \left[(1 - \gamma)^n \beta^{t-1-n}\right] = (1 - \gamma)^n \lim_{t \to \infty} \left[\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T \mathcal{L}(\theta^{t-1})}\right] = (1 - \gamma)^n \frac{\mathcal{L}(\theta^*)}{\mathbb{1}_k^T \mathcal{L}(\theta^*)}. \tag{22}$$

Now, consider $g(n) = (1-\gamma)^n$. For this choice of $g(n)$, we have that:

$$\|(1-\gamma)^n \beta^{t-1-n}\| \leq (1-\gamma)^n \|\beta^{t-1-n}\| \leq g(n),\ \forall t, n \in \mathbb{N} \tag{23}$$

$$\sum_{n=0}^{\infty} g(n) = \sum_{n=0}^{\infty} (1-\gamma)^n = \frac{1}{1-(1-\gamma)} = \frac{1}{\gamma} < \infty. \tag{24}$$

We now invoke the dominated convergence theorem. Recalling Eq. (21), we observe that:

$$\lim_{t\to\infty} \alpha^t = \gamma \lim_{t\to\infty} \left[\sum_{n=0}^{t-1} (1-\gamma)^n \beta^{t-1-n} \right] \tag{25}$$

$$= \gamma \sum_{n=0}^{\infty} (1-\gamma)^n \lim_{t\to\infty} \beta^{t-1-n} = \gamma \sum_{n=0}^{\infty} (1-\gamma)^n \frac{\mathcal{L}(\theta^*)}{\mathbb{1}_k^T \mathcal{L}(\theta^*)} \tag{26}$$

$$= (\gamma)\left(\frac{1}{\gamma}\right) \frac{\mathcal{L}(\theta^*)}{\mathbb{1}_k^T \mathcal{L}(\theta^*)} = \frac{\mathcal{L}(\theta^*)}{\mathbb{1}_k^T \mathcal{L}(\theta^*)} = \alpha^*. \tag{27}$$

Therefore, $\lim_{t\to\infty} \alpha^t = \alpha^* = \mathcal{L}(\theta^*)/\left[\mathbb{1}_k^T \mathcal{L}(\theta^*)\right]$. Cumulatively, for $\eta \in (0, 2/L^*)$ and $\gamma \in (0,1)$, under learn2mix training, $\lim_{t\to\infty} \theta^t = \theta^*$, and $\lim_{t\to\infty} \alpha^t = \alpha^* = \mathcal{L}(\theta^*)/\left[\mathbb{1}_k^T \mathcal{L}(\theta^*)\right]$. □

Corollary 2.4. *Let $\mathcal{L}(\theta^t) \in \mathbb{R}^k$ denote the class-wise loss vector at time t. Suppose each class-wise loss, $\mathcal{L}_i(\theta) \in \mathbb{R}$, is strongly convex in θ, with strong convexity parameter $\mu_i \in \mathbb{R}_{>0}$, $\forall i \in \{1,\ldots,k\}$, and suppose each class-wise loss gradient $\nabla_\theta \mathcal{L}_i(\theta) \in \mathbb{R}^m$ is Lipschitz continuous in θ with Lipschitz constant $L_i \in \mathbb{R}_{\geq 0}$, $\forall i \in \{1,\ldots,k\}$. Let $\mu^* = \min_{i\in\{1,\ldots,k\}} \mu_i$, $L^* = \max_{i\in\{1,\ldots,k\}} L_i$. Then, the following holds, $\forall \alpha \in [0,1]^k$, with $\sum_{i=1}^k \alpha_i = 1$:*

$$\frac{\mu^*}{2}\|\theta^t - \theta^*\| \leq \|\nabla_\theta \mathcal{L}(\theta^t, \alpha)\| \leq L^* \|\theta^t - \theta^*\|, \tag{28}$$

$$\textit{Wherein: } \|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\| + \|\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\| \leq 2L^* \|\theta^t - \theta^*\|. \tag{29}$$

Proof. We begin by recalling that $\mathcal{L}_i(\theta)$ is strongly convex in θ with strong convexity parameter μ_i, $\forall i \in \{1,\ldots,k\}$. Accordingly, $\forall \alpha \in [0,1]^k$, with $\sum_{i=1}^k \alpha_i = 1$, the loss function $\mathcal{L}(\theta, \alpha)$ is strongly convex in θ with parameter, $\mu' \in \mathbb{R}_{>0}$, which is lower bounded by $\mu^* \in \mathbb{R}_{>0}$, as per Eq. (30).

$$\mu' \geq \mu^* > 0, \qquad \text{where:} \quad \mu^* = \min_{i\in\{1,\ldots,k\}} \mu_i, \qquad \text{and:} \quad \mu' = \sum\nolimits_{i=1}^k \alpha_i \mu_i. \tag{30}$$

Now, recall that $\nabla_\theta \mathcal{L}_i(\theta)$, is Lipschitz continuous in θ with Lipschitz constant L_i, $\forall i \in \{1,\ldots,k\}$. Accordingly, $\forall \alpha \in [0,1]^k$, where $\sum_{i=1}^k \alpha_i = 1$, the loss gradient $\nabla_\theta \mathcal{L}(\theta, \alpha)$ is Lipschitz continuous in θ with Lipschitz constant, $L' \in \mathbb{R}_{\geq 0}$, which is upper bounded by $L^* \in \mathbb{R}_{\geq 0}$, as per Eq. (31).

$$L^* \geq L' \geq 0, \qquad \text{where:} \quad L^* = \max_{i\in\{1,\ldots,k\}} L_i, \qquad \text{and:} \quad L' = \sum\nolimits_{i=1}^k \alpha_i L_i. \tag{31}$$

Note that $\nabla_\theta \mathcal{L}(\theta^*, \alpha) = \mathbf{0}_m$. Since $\mathcal{L}(\theta, \alpha)$ is strongly convex in θ, the following inequalities hold:

$$\mathcal{L}(\theta^t, \alpha) - \mathcal{L}(\theta^*, \alpha) \geq \nabla_\theta \mathcal{L}(\theta^*, \alpha)^T (\theta^t - \theta^*) + \frac{\mu'}{2}\|\theta^t - \theta^*\|^2 = \frac{\mu'}{2}\|\theta^t - \theta^*\|^2, \tag{32}$$

$$\mathcal{L}(\theta^t, \alpha) - \mathcal{L}(\theta^*, \alpha) \leq \nabla_\theta \mathcal{L}(\theta^t, \alpha)^T (\theta^t - \theta^*) \leq \|\nabla_\theta \mathcal{L}(\theta^t, \alpha)\| \|\theta^t - \theta^*\|. \tag{33}$$

Combining Eq. (32) and Eq. (33), and recalling Eq. (30), we obtain the following inequality:

$$\|\nabla_\theta \mathcal{L}(\theta^t, \alpha)\| \geq \frac{\mathcal{L}(\theta^t, \alpha) - \mathcal{L}(\theta^*, \alpha)}{\|\theta^t - \theta^*\|} \geq \frac{\mu^*}{2}\|\theta^t - \theta^*\|. \tag{34}$$

Furthermore, since $\nabla_\theta \mathcal{L}(\theta, \alpha)$ is Lipschitz continuous in θ and recalling Eq. (31), it follows that:

$$\|\nabla_\theta \mathcal{L}(\theta^t, \alpha) - \nabla_\theta \mathcal{L}(\theta^*, \alpha)\| \leq L' \|\theta^t - \theta^*\| \implies \|\nabla_\theta \mathcal{L}(\theta^t, \alpha)\| \leq L^* \|\theta^t - \theta^*\|. \tag{35}$$

Altogether, combining Eq. (34) and Eq. (35), we arrive at the final inequality:

$$\frac{\mu^*}{2}\|\theta^t - \theta^*\| \leq \|\nabla_\theta \mathcal{L}(\theta^t, \alpha)\| \leq L^* \|\theta^t - \theta^*\|. \tag{36}$$

Furthermore, since Eq. (35) holds $\forall \alpha \in [0,1]^k$ where $\sum_{i=1}^k \alpha_i = 1$, it follows that:

$$\|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\| + \|\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\| \leq 2L^* \|\theta^t - \theta^*\|. \tag{37}$$

□

Proposition 2.5. *Let $\mathcal{L}(\theta^t), \mathcal{L}(\theta^*) \in \mathbb{R}^k$ denote the respective class-wise loss vectors for the model parameters at time t and for the optimal model parameters. Suppose each class-wise loss, $\mathcal{L}_i(\theta) \in \mathbb{R}$ is strongly convex in θ with strong convexity parameter $\mu_i \in \mathbb{R}_{>0}$, $\forall i \in \{1, \ldots, k\}$, and each class-wise loss gradient $\nabla_\theta \mathcal{L}_i(\theta) \in \mathbb{R}^m$ is Lipschitz continuous in θ, having Lipschitz constant $L_i \in \mathbb{R}_{\geq 0}$, $\forall i \in \{1, \ldots, k\}$. Moreover, suppose that the loss gradient $\nabla_\theta \mathcal{L}(\theta, \alpha) \in \mathbb{R}^m$ is Lipschitz continuous in α, having Lipschitz constant $L_\alpha \in \mathbb{R}_{\geq 0}$, and let $\mu^* = \min_{i \in \{1,\ldots,k\}} \mu_i$, $L^* = \max_{i \in \{1,\ldots,k\}} L_i$. Then, if and only if the following holds:*

$$\Big[\Big(\frac{\mu^*}{2} - L^*\Big)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\Big]\Big[\|\theta^t - \theta^*\| - (\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\Big] > 0, \tag{38}$$

It follows that for every learning rate, $\eta > 0$, and for every mixing rate, $\gamma \in (0, \beta]$:

$$\big\|\big(\theta^t - \eta\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\big) - \theta^*\big\| \leq \big\|\big(\theta^t - \eta\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\big) - \theta^*\big\|, \tag{39}$$

$$\textit{Where: } \beta = \frac{\big(\frac{\mu^*}{2} - L^*\big)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))}{\eta L_\alpha L^*\Big\|\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\Big\|\Big[\|\theta^t - \theta^*\| - (\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\Big]} \tag{40}$$

Proof. We note that for all subsequent derivations, $\mathcal{F}(\theta^t, \theta^*, \eta, \alpha^t) = \|(\theta^t - \eta\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)) - \theta^*\|$, and $\mathcal{G}(\theta^t, \theta^*, \eta, \tilde{\alpha}) = \|(\theta^t - \eta\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})) - \theta^*\|$, where $\alpha^{t-1} = \tilde{\alpha}$. We begin by observing that:

$$\big[\mathcal{F}(\theta^t, \theta^*, \eta, \alpha^t)\big]^2 = \|\theta^t - \theta^*\|^2 - 2\eta(\theta^t - \theta^*)^T\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) + \eta^2\|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\|^2, \tag{41}$$

$$\big[\mathcal{F}(\theta^t, \theta^*, \eta, \tilde{\alpha})\big]^2 = \|\theta^t - \theta^*\|^2 - 2\eta(\theta^t - \theta^*)^T\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha}) + \eta^2\|\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\|^2. \tag{42}$$

Accordingly, the difference between $\big[\mathcal{F}(\theta^t, \theta^*, \eta, \alpha^t)\big]^2$ and $\big[\mathcal{G}(\theta^t, \theta^*, \eta, \tilde{\alpha})\big]^2$ is given by:

$$\begin{aligned}\big[\mathcal{F}(\theta^t, \theta^*, \eta, \alpha^t)\big]^2 - \big[\mathcal{G}(\theta^t, \theta^*, \eta, \tilde{\alpha})\big]^2 = &-2\eta\big[(\theta^t - \theta^*)^T(\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) - \nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha}))\big] \\ &+ \eta^2\big[\|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\|^2 - \|\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\|^2\big].\end{aligned} \tag{43}$$

Consequently, suppose that $\mathcal{H}(\theta^t, \theta^*, \eta, \tilde{\alpha}, \alpha^t) = 2\eta\big[(\theta^t - \theta^*)^T(\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) - \nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha}))\big]$, and let $\mathcal{J}(\theta^t, \eta, \tilde{\alpha}, \alpha^t) = \eta^2\big[\|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\|^2 - \|\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\|^2\big]$. Suppose the loss gradient, $\nabla_\theta \mathcal{L}(\theta, \alpha)$, is Lipschitz continuous in α with Lipschitz constant, L_α. We now upper bound $\mathcal{J}(\theta^t, \eta, \tilde{\alpha}, \alpha^t)$:

$$\begin{aligned}\mathcal{J}(\theta^t, \eta, \alpha, \alpha^t) &= \eta^2\big[\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) - \nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\big]^T\big[\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) + \nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\big] \\ &\leq \|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) - \nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\|\|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) + \nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\| && (44) \\ &\leq 2\eta^2 L_\alpha\|\alpha^t - \tilde{\alpha}\|\Big[\|\nabla_\theta \mathcal{L}(\theta^t, \alpha^t)\| + \|\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\|\Big] && (45) \\ &\leq 2\eta^2 L_\alpha L^*\|\alpha^t - \tilde{\alpha}\|\|\theta^t - \theta^*\| && (46) \\ &= 2\eta^2 L_\alpha L^*\Big\|\tilde{\alpha} + \gamma\Big(\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\Big) - \tilde{\alpha}\Big\|\|\theta^t - \theta^*\| && (47) \\ &= 2\eta^2 L_\alpha L^*\gamma\Big\|\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\Big\|\|\theta^t - \theta^*\|. && (48)\end{aligned}$$

We note that this upper bound follows from the Cauchy-Schwarz inequality and Corollary 2.4. We now proceed by lower bounding $\mathcal{H}(\theta^t, \theta^*, \eta, \tilde{\alpha}, \alpha^t)$:

$$\begin{aligned}\mathcal{H}(\theta^t, \theta^*, \eta, \tilde{\alpha}, \alpha^t) &= 2\eta\Big[(\theta^t - \theta^*)^T\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) - (\theta^t - \theta^*)^T\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\Big] && (49) \\ &\geq 2\eta\Big[(\theta^t - \theta^*)^T\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) - \|\theta^t - \theta^*\|\|\nabla_\theta \mathcal{L}(\theta^t, \tilde{\alpha})\|\Big] && (50) \\ &\geq 2\eta\Big[(\theta^t - \theta^*)^T\nabla_\theta \mathcal{L}(\theta^t, \alpha^t) - L^*\|\theta^t - \theta^*\|^2\Big] && (51) \\ &= 2\eta\left[\frac{\mu^*}{2}\|\theta^t - \theta^*\|^2 + \mathcal{L}(\theta^t, \alpha^t) - \mathcal{L}(\theta^*, \alpha^t) - L^*\|\theta^t - \theta^*\|^2\right] && (52) \\ &= 2\eta\Bigg[\Big(\frac{\mu^*}{2} - L^*\Big)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*)) \\ &\qquad + \gamma\Big(\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\Big)^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\Bigg]. && (53)\end{aligned}$$

As in above, we note that this lower bound also follows from the Cauchy-Schwarz inequality and Corollary 2.4, and further invokes the strong convexity of $\mathcal{L}(\theta, \alpha)$ in θ. Combining Eq. (48) and Eq. (53), we obtain the following upper bound on $[\mathcal{F}(\theta^t, \theta^*, \eta, \alpha^t)]^2 - [\mathcal{G}(\theta^t, \theta^*, \eta, \tilde{\alpha})]^2$:

$$\left[\mathcal{F}(\theta^t, \theta^*, \eta, \alpha^t)\right]^2 - \left[\mathcal{G}(\theta^t, \theta^*, \eta, \tilde{\alpha})\right]^2 \leq \mathcal{K}(\theta^t, \theta^*, \eta, \gamma, \tilde{\alpha}, \alpha^t), \tag{54}$$

$$\begin{aligned}\text{Where: } \mathcal{K}(\theta^t, \theta^*, \eta, \gamma, \tilde{\alpha}, \alpha^t) = -2\eta\Bigg[&\left(\frac{\mu^*}{2} - L^*\right)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*)) \\ &+ \gamma\left(\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\right)^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\Bigg] \\ &+ 2\eta^2 L_\alpha L^* \gamma\left\|\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\right\|\|\theta^t - \theta^*\|.\end{aligned} \tag{55}$$

Now, consider the following chain of inequalities deriving from Eq. (54):

$$\begin{aligned}\mathcal{K}(\theta^t, \theta^*, \eta, \gamma, \tilde{\alpha}, \alpha^t) \leq 0 \implies &\left[\mathcal{F}(\theta^t, \theta^*, \eta, \alpha^t)\right]^2 - \left[\mathcal{G}(\theta^t, \theta^*, \eta, \tilde{\alpha})\right]^2 \leq 0 \\ \implies &\left[\mathcal{F}(\theta^t, \theta^*, \eta, \alpha^t)\right] \leq \left[\mathcal{G}(\theta^t, \theta^*, \eta, \tilde{\alpha})\right].\end{aligned} \tag{56}$$

Accordingly, we aim to find a condition on the mixing rate, γ, under which the chain of inequalities is satisfied. We proceed by letting $\mathcal{K}(\theta^t, \theta^*, \eta, \gamma, \tilde{\alpha}, \alpha^t) \leq 0$, and rearrange the terms:

$$\begin{aligned}\left(\frac{\mu^*}{2} - L^*\right)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*)) \geq \gamma\Bigg[&\eta L_\alpha L^*\left\|\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\right\|\|\theta^t - \theta^*\| \\ &- \left(\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\right)^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\Bigg].\end{aligned} \tag{57}$$

We note that this chain of inequalities is satisfied if, for every $\eta > 0$:

$$0 < \gamma \leq \frac{\left(\frac{\mu^*}{2} - L^*\right)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))}{\eta L_\alpha L^*\left\|\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\right\|\|\theta^t - \theta^*\| - \left(\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\right)^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))} \leq \beta, \tag{58}$$

$$\text{Where: } \beta = \frac{\left(\frac{\mu^*}{2} - L^*\right)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))}{\eta L_\alpha L^*\left\|\frac{\mathcal{L}(\theta^{t-1})}{\mathbb{1}_k^T\mathcal{L}(\theta^{t-1})} - \tilde{\alpha}\right\|\left[\|\theta^t - \theta^*\| - (\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\right]}. \tag{59}$$

However, $\gamma > 0$ iff the numerator and denominator from Eq. (59) have the same sign, ensuring that $\beta > 0$. Accordingly, if and only if the condition provided in Eq. (60) is satisfied:

$$\left[\left(\frac{\mu^*}{2} - L^*\right)\|\theta^t - \theta^*\|^2 + \tilde{\alpha}^T(\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\right]\left[\|\theta^t - \theta^*\| - (\mathcal{L}(\theta^t) - \mathcal{L}(\theta^*))\right] > 0, \tag{60}$$

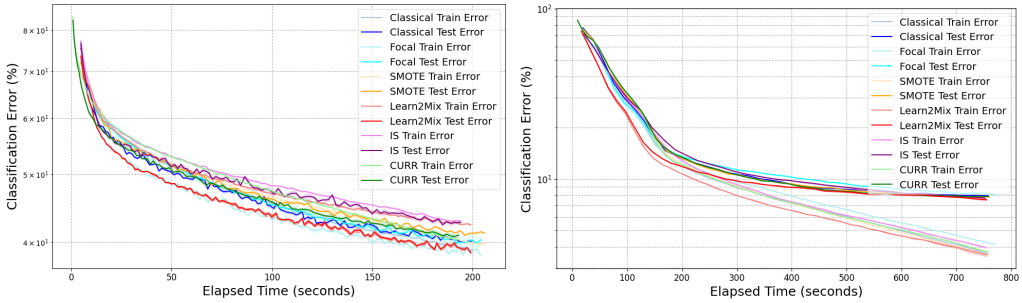
It follows that for every learning rate $\eta > 0$, and for every mixing rate $\gamma \in (0, \beta]$ satisfying Eq. (59):

$$\left\|\left(\theta^t - \eta\nabla_\theta\mathcal{L}(\theta^t, \alpha^t)\right) - \theta^*\right\| \leq \left\|\left(\theta^t - \eta\nabla_\theta\mathcal{L}(\theta^t, \tilde{\alpha})\right) - \theta^*\right\|. \tag{61}$$

□

B Additional Empirical Results

For further performance verification of learn2mix, we present several ablation studies quantifying the effects of different architectures, optimizers, batch sizes, learning rates, and mixing rates for the considered classification tasks from the main text. We further present the worst-class classification accuracy on Imagenette to further gauge the efficacy of learn2mix within imbalanced classification settings, and illustrate how the mixing parameters converge to a stable distribution on Mean Estimation. We first consider CIFAR-10 and CIFAR-100 (per Section C), and evaluate whether the gains afforded by learn2mix persist across architectures. For CIFAR-10, we recall the Large LeNet architecture, trained using the Adam optimizer and Cross Entropy Loss with learning rate $\eta =$ 7e-5 for $\underline{E} = 200$ s, and the MobileNet-V3 Small architecture, trained using the Adam optimizer and Cross Entropy Loss with learning rate $\eta =$ 1e-4 for $\underline{E} = 750$ s. For CIFAR-100, we consider again the MobileNet-V3 Small architecture, trained using the Adam optimizer and Cross Entropy Loss with learning rate



(a) CIFAR-10: Large LeNet

(b) CIFAR-10: MobileNet-V3 Small

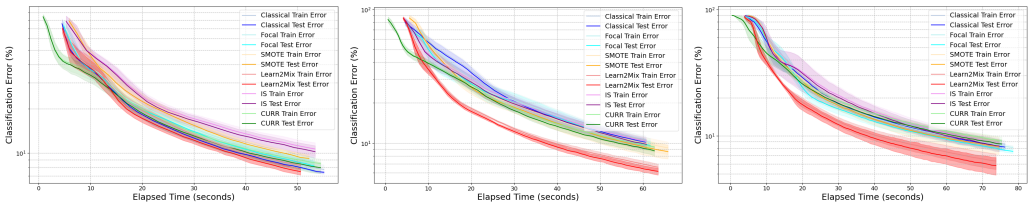
(c) CIFAR-100: Large LeNet

(d) CIFAR-100: MobileNet-V3 Small

Figure 4: Comparing model classification errors for learn2mix, classical, FCL, SMOTE, IS, and CURR training. The x-axis is the elapsed [training] time, while the y-axis is the classification error.

$\eta = 1\text{e-}4$ for $\underline{E} = 200$ s, and the Large LeNet architecture, trained using the Adam optimizer and Cross Entropy Loss with learning rate $\eta = 1\text{e-}4$ for $\underline{E} = 50$ s. The results are depicted in Figure 4. We observe that for both Large LeNet and MobileNet-V3 Small, the learn2mix-trained models converge faster than the classical, FCL, SMOTE, IS, and CURR trained models.

Next, we evaluate the robustness of learn2Mix to different optimizers and batch sizes. As we used the Adam optimizer in the main text, we now consider the RMSProp optimizer [Graves, 2013] with batch size $M \in \{250, 500, 1000\}$. We train LeNet-5 on MNIST using Cross Entropy Loss with learning rate $\eta = 1\text{e-}5$ for $\underline{E} = 45$ s, $\underline{E} = 60$ s, and $\underline{E} = 70$ s. As depicted in Figure 5, we see that learn2mix converges faster than classical, FCL, SMOTE, IS, and CURR training.



(a) MNIST: $M = 150$

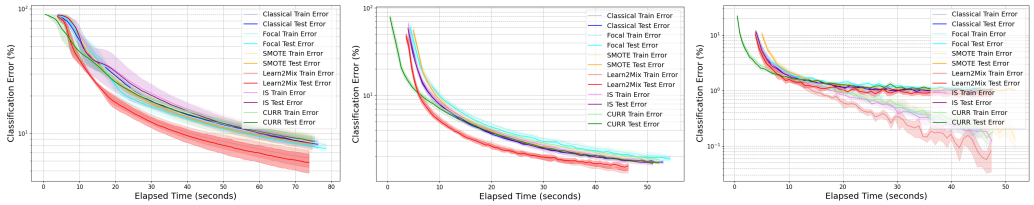
(b) MNIST: $M = 500$

(c) MNIST: $M = 1000$

Figure 5: Comparing model classification errors for learn2mix, classical, FCL, SMOTE, IS, and CURR training. The x-axis is the elapsed [training] time, while the y-axis is the classification error.

We further verify the robustness of learn2Mix to different learning rates. We train LeNet-5 on MNIST using Cross Entropy Loss with learning rate $\eta \in \{1\text{e-}5, 1\text{e-}4, 1\text{e-}3\}$ for $\underline{E} = 75$ s, $\underline{E} = 50$ s, and $\underline{E} = 45$ s. Per Figure 6, we observe that the faster convergence afforded by learn2mix is apparent for $\eta \in \{1\text{e-}5, 1\text{e-}4\}$. For $\eta = 1\text{e-}3$, we note that after convergence, the learn2mix train error continues to decreases at a faster rate than the the classical, FCL, SMOTE, IS, and CURR train errors.

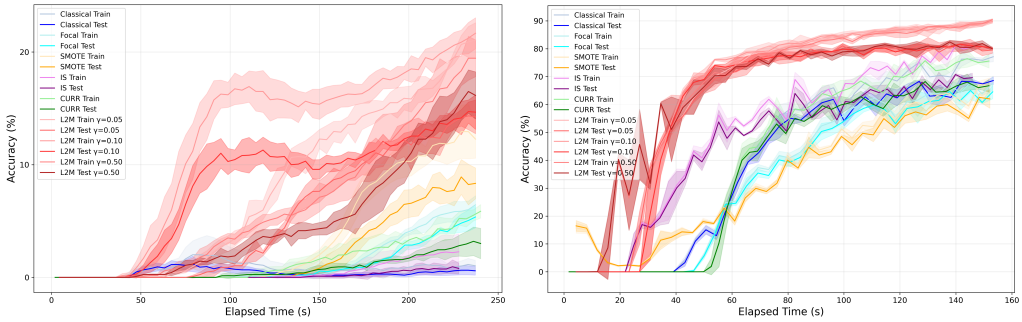
We now illustrate the worst-class classification accuracy on Imagenette and IMDB as an additional metric to gauge the efficacy of learn2mix for imbalanced classification settings. We train ResNet-18



(a) MNIST: $\eta =$ 1e-5 (b) MNIST: $\eta =$ 1e-4 (c) MNIST: $\eta =$ 1e-3

Figure 6: Comparing model classification errors for learn2mix, classical, FCL, SMOTE, IS, and CURR training. The x-axis is the elapsed [training] time, while the y-axis is the classification error.

on Imagenette via Cross Entropy Loss with learning rate $\eta =$ 1e-5 for $\underline{E} = 240$ s, and a transformer on IMDB using Cross-Entropy Loss with learning rate $\eta =$ 1e-4 for $\underline{E} = 150$ s, and record the test classification accuracy of the worst class after each training epoch, t. To demonstrate relative insensitivity to the choice of γ, we ablate the mixing rate for $\gamma \in [0.01, 0.1]$. The result is depicted in Figure 7. We see that learn2mix offers a considerable improvement in the worst-class classification accuracy metric versus classical, FCL, SMOTE, IS, and CURR training, which matches intuition; the theoretical foundation of learn2Mix is to increase the proportion of harder classes during training, which directly translates to stronger results for the most challenging classes.



(a) Imagenette (b) IMDB

Figure 7: Comparing worst-class model classification accuracies using learn2mix, classical, FCL, SMOTE, IS, and CURR training on Imagenette and IMDB. The x-axis is the elapsed [training] time, while the y-axis is the classification accuracy of the worst-class.

To illustrate how the mixing parameters converge to a stable distribution during training (as detailed in Section 2), we train a fully connected network on Mean Estimation (where the Normal, Exponential, and Chi-squared cases have similar variance but the Uniform case is substantially more variable) using Cross-Entropy Loss with learning rate $\eta =$ 5e-5 for $E = 500$ epochs. As depicted in Figure 8, learn2mix prioritizes the hardest class without overstating differences among the easier ones.

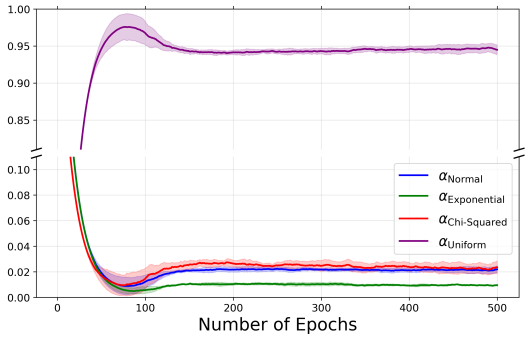


Figure 8: Evolution of learn2mix mixing parameters across training epochs on Mean Estimation.

C Dataset Descriptions

C.1 MNIST Dataset

The **MNIST** (Modified National Institute of Standards and Technology) dataset is a collection of handwritten digits commonly used to train image processing systems. For the MNIST classification result from Section 4.1, the original training dataset, J, comprises $N = 60000$ samples, wherein the fixed-proportion mixing parameters (for default numerical class ordering of digits from $1-10$) are:

$$\tilde{\alpha} = [0.0987, 0.1124, 0.0993, 0.1022, 0.0974, 0.0904, 0.0986, 0.1044, 0.0975, 0.0991]^T$$

The test dataset, K, comprises $N_{\text{test}} = 10000$ samples, with class proportions equivalent to the class proportions in the base MNIST test dataset. For MNIST reconstruction (see Section 4.3), we utilize manual class imbalancing, reducing the number of samples comprising each numerical class $6-10$ by a factor of 5. The original training dataset, J, now contains $N = 36475$ samples, wherein the fixed-proportion mixing parameters (for default numerical class ordering of digits from $1-10$) are:

$$\tilde{\alpha} = [0.1624, 0.1848, 0.1633, 0.1681, 0.1602, 0.0297, 0.0324, 0.0344, 0.0321, 0.0326]^T$$

We note that the test dataset maintains the same class proportions as in the base MNIST test dataset. The features and labels within MNIST are summarized as follows:

- Each feature (image) is of size 28×28, representing grayscale intensities from 0 to 255.
- Target Variable: The numerical class (digit) the image represents, ranging from 1 to 10.

C.2 Fashion MNIST Dataset

The **Fashion MNIST** dataset is a collection of clothing images commonly used to train image processing systems. For the Fashion MNIST classification result from Section 4.1, the original training dataset, J, consists of $N = 60000$ samples, wherein the fixed-proportion mixing parameters (for default numerical class ordering of clothing from $1-10$) are:

$$\tilde{\alpha} = [0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1, 0.1]^T = (0.1)\mathbb{1}_{10}$$

The test dataset, K, comprises $N_{\text{test}} = 10000$ samples, with class proportions equivalent to the class proportions in the base Fashion MNIST test dataset. For Fashion MNIST reconstruction (see Section 4.3), we use manual class imbalancing, reducing the number of samples within each numerical class $6-10$ by a factor of 5. The original training dataset J, now has $N = 36000$ samples. The fixed-proportion mixing parameters (for default numerical class ordering of clothing from $1-10$) are:

$$\tilde{\alpha} = [(0.1667)\mathbb{1}_5^T, (0.0333)\mathbb{1}_5^T]^T$$

We note that the test dataset maintains the same class proportions as in the base Fashion MNIST test dataset. The features and labels within Fashion MNIST are summarized as follows:

- Each feature (image) is of size 28×28, representing grayscale intensities from 0 to 255.
- Target Variable: The numerical class (clothing) the image represents, ranging from 1 to 10.

C.3 CIFAR-10 Dataset

The **CIFAR-10** dataset is a collection of color images categorized into 10 different classes, and is commonly used to train image processing systems. For the CIFAR-10 classification result in Section 4.1, the original training dataset, J, comprises $N = 50000$ samples, wherein the fixed-proportion mixing parameters (for default numerical class ordering of categories from $1-10$) are:

$$\tilde{\alpha} = (0.1)\mathbb{1}_{10}$$

The test dataset, K, comprises $N_{\text{test}} = 10000$ samples, with class proportions equivalent to the class proportions in the base CIFAR-10 test dataset. For CIFAR-10 reconstruction (see Section 4.3), we use manual class imbalancing, reducing the number of samples in numerical classes $1-4, 7-10$ by a factor of 10. The original training dataset, J, now has $N = 14000$ samples. The fixed-proportion mixing parameters (for default numerical class ordering of categories from $1-10$) are:

$$\tilde{\alpha} = [(0.0357)\mathbb{1}_4^T, (0.3571)\mathbb{1}_2^T, (0.0357)\mathbb{1}_4^T]^T$$

We note that the test dataset maintains the same class proportions found in the base CIFAR-10 test dataset. The features and labels within CIFAR-10 are summarized as follows:

- Each feature (image) is of size $32 \times 32 \times 3$, with three color channels (RGB), and size 32 x 32 pixels for each channel, represented as a grayscale intensity from 0 to 255.
- Target Variable: The numerical class (category) the image represents, ranging from 1 to 10.

C.4 Imagenette Dataset

The **Imagenette** dataset contains a subset of 10 classes from the ImageNet dataset of color images, and is commonly used to train image processing systems. The base Imagenette training dataset, I, comprises $N_I = 9469$ samples, and the base Imagenette test dataset, K, comprises $N_{\text{test}} = 3925$ samples. For the Imagenette classification result in Section 4.1, we utilize manual class imbalancing. Let $N_i \in \mathbb{N}$ be the number of samples in each class, $i \in \{1, \ldots, 10\}$, from I, where $N_I = \sum_{i=1}^{10} N_i$. We define $\epsilon_i = 1 - 0.1i$, $\forall i \in \{1, \ldots, 10\}$ as the linearly decreasing *imbalance factor*. Accordingly, the original training dataset, J, has $N = \sum_{i=1}^{10} \epsilon_i N_i = 5207$ samples. The fixed-proportion mixing parameters (for default numerical class ordering of categories from $1 - 10$) are:

$$\tilde{\alpha} = [0.1849, 0.1650, 0.1525, 0.1152, 0.1083, 0.0918, 0.0737, 0.0536, 0.0365, 0.0184]^T$$

We note that the test dataset maintains the same class proportions found in the base Imagenette test dataset. The features and labels within Imagenette are summarized as follows:

- Each feature (image) is of size $224 \times 224 \times 3$, with three color channels (RGB), and size 224 x 224 pixels for each channel, represented as a grayscale intensity from 0 to 255.
- Target Variable: The numerical class (category) the image represents, ranging from 1 to 10.

C.5 CIFAR-100 Dataset

The **CIFAR-100** dataset is a collection of color images categorized into 100 different classes, and is commonly used to train image processing systems. The base CIFAR-100 training dataset, I, has $N_I = 50000$ samples, and the base CIFAR-100 test dataset, K, has $N_{\text{test}} = 10000$ samples. For the CIFAR-100 classification result in Section 4.1, we utilize manual class imbalancing. Let $N_i \in \mathbb{N}$ be the number of samples in each class, $i \in \{1, \ldots, 100\}$, from I, whereby $N_I = \sum_{i=1}^{100} N_i$. We define $\epsilon_i = 40^{-i/100}$, $\forall i \in \{1, \ldots, 100\}$ as the logarithmically decreasing *imbalance factor*. Accordingly, the original training dataset, J, has $N = \sum_{i=1}^{100} \epsilon_i N_i = 13209$ samples. The fixed-proportion mixing parameters (for default numerical class ordering of categories from $1 - 100$) are:

$$\tilde{\alpha} = [\tilde{\alpha}_1, \tilde{\alpha}_2, \ldots, \tilde{\alpha}_{100}]^T, \quad \text{where:} \quad \tilde{\alpha}_i = (\epsilon_i N_i)/N, \ \forall i \in \{1, \ldots, 100\}$$

We note that the test dataset maintains the same class proportions found in the base CIFAR-100 test dataset. The features and labels within CIFAR-100 are summarized as follows:

- Each feature (image) is of size $32 \times 32 \times 3$, with three color channels (RGB), and size 32 x 32 pixels for each channel, represented as a grayscale intensity from 0 to 255.
- Target Variable: The numerical class (category) the image denotes, ranging from 1 to 100.

C.6 IMDB Dataset

The **IMDB** dataset is a collection of movie reviews, categorized as positive or negative in sentiment. We split the IMDB dataset such that the base IMDB training dataset, I, has $N_I = 40000$ samples, and the base IMDB test dataset, K, consists of $N_{\text{test}} = 10000$ samples. For the IMDB classification result in Section 4.1, we leverage manual class imbalancing, wherein numerical class 1 retains 30% of its samples. Accordingly, the original training dataset, J, has $N = 26000$ samples. The fixed-proportion mixing parameters (for default numerical class ordering of sentiment from $1, 2$) are:

$$\tilde{\alpha} = [0.2307, 0.7693]^T$$

We note that the test dataset maintains the same class proportions as in the base IMDB test dataset. The features and labels within the IMDB dataset are summarized as follows:

- Each feature (review) is tokenized and encoded as a sequence of word indices with a max length of 500 tokens. Sequences are padded or truncated to ensure uniform length.
- Target Variable: The numerical class (sentiment) the review represents, either 1 or 2.

C.7 Mean Estimation Dataset

The **Mean Estimation** dataset is a synthetic benchmark designed for regression tasks, wherein each example, (x_j, y_j), comprises a 10-dimensional feature vector, x_j, of samples from one of four statistical distributions, and the mean, y_j, of this distribution. We create an imbalanced original training dataset, J, with $N = 3000$ samples, where J_1 has 1000 examples drawn from a normal distribution with $\sigma = 1$, J_2 has 1000 examples drawn from an exponential distribution, J_3 has 800 examples drawn from a chi-squared distribution, and J_4 has 200 samples drawn from a uniform distribution. The fixed-proportion mixing parameters (for numerical ordering of distributions from $1 - 4$) are:

$$\tilde{\alpha} = [0.333, 0.333, 0.267, 0.067]^T$$

The test dataset, K, is created as a balanced dataset that has 1000 examples from each distribution, wherein $N_{\text{test}} = 4000$. The Mean Estimation dataset features and labels are summarized as follows:

- Each feature (vector of samples) is generated from one of four statistical distributions (normal, exponential, chi-squared, uniform). The feature vectors are created by sampling from these distributions with means uniformly drawn from the interval $[0, 1]$ for normal, exponential, and chi-squared distributions, and from $[20, 50]$ for the uniform distribution.
- Target Variable: The mean parameter used to generate the vector of samples, representing the underlying expected value of the chosen distribution.

C.8 Wine Quality Dataset

The **Wine Quality** dataset consists of physicochemical tests on white and red wine samples, and the corresponding quality rating. We treat the wine type (white $= 1$, red $= 2$) as a categorical variable, wherein $k = 2$. We split the Wine Quality dataset such that the base Wine Quality training dataset, J, has $N = 3248$ samples, and the base Wine Quality test dataset, K, has $N_{\text{test}} = 3249$ samples. For the Wine Quality regression result in Section 4.2, we utilize manual class imbalancing, reducing the number of samples in numerical class 1 by a factor of 10. The original training dataset, J, now has $N = 1043$ samples, where the fixed-proportion mixing parameters (for numerical class ordering of wine type from $1, 2$) are:

$$\tilde{\alpha} = [0.234, 0.766]^T$$

We note that the test dataset maintains the same class proportions as in the base Wine Quality test dataset. The features and labels within the Wine Quality dataset are summarized as follows:

- Each feature (physicochemical tests) contains a set of test results, and is of size 11×1.
- Target Variable: The wine quality rating given to the set of physicochemical tests.

C.9 California Housing Dataset

The **California Housing** dataset contains housing data from California and their associated prices. As the ocean proximity variable is categorical (<1H OCEAN $= 1$, INLAND $= 2$, NEAR BAY $= 3$, NEAR OCEAN $= 4$), we denote $k = 4$. We split the California Housing dataset such that the base California Housing training dataset, J, has $N = 10214$ samples, and the base California Housing test dataset, K, has $N_{\text{test}} = 10214$ samples. For the California Housing regression result in Section 4.2, we use manual class imbalancing, reducing the number of samples in numerical classes $1, 2, 4$ by a factor of 20. The original training dataset, J, now has $N = 3641$ samples. The fixed-proportion mixing parameters (for numerical class ordering of ocean proximity from $1 - 4$) are:

$$\tilde{\alpha} = [0.0615, 0.9055, 0.0154, 0.0176]^T$$

We note that the test dataset maintains the same class proportions as in the base California Housing test dataset. The features and labels in the California Housing dataset are summarized as follows:

- Each feature (housing data) contains various housing attributes, and is of size 8×1.
- Target Variable: The housing price associated with the housing data.

D Experiment Details

D.1 Neural Network Architectures

We provide comprehensive descriptions for six different neural network architectures designed for various tasks: classification, regression, and image reconstruction. Each of these architectures were employed to generate the respective empirical results pertaining to the aforementioned tasks.

D.1.1 Fully Connected Networks

We leverage fully connected networks in our analysis for regression on Mean Estimation, California Housing, and Wine Quality. The network consists of the following layers, wherein $d = 10$ for Mean Estimation, $d = 11$ for Wine Quality, and $d = 8$ for California Housing:

- **Fully Connected Layer** (`fc1`): Transforms the input features from a d-dimensional space to a 64-dimensional space.
- **ReLU Activation** (`relu`): Applies the ReLU activation function to the output of `fc1`.
- **Fully Connected Layer** (`fc2`): Maps the 64-dimensional representation from `relu` to a 1-dimensional output.

D.1.2 Convolutional Neural Networks

We utilize the LeNet-5 convolutional neural network architecture in our analysis for image classification on MNIST and Fashion MNIST. The network consists of the following layers:

- **Convolutional Layer** (`conv1`): Applies a 2D convolution with 1 input channel, 6 output channels, and a kernel size of 5.
- **ReLU Activation** (`relu1`): Applies the ReLU activation function to the output of `conv1`.
- **Max Pooling Layer** (`pool1`): Performs 2x2 max pooling on the output of `relu1`.
- **Convolutional Layer** (`conv2`): Applies a 2D convolution with 6 input channels, 16 output channels, and a kernel size of 5.
- **ReLU Activation** (`relu2`): Applies the ReLU activation function to the output of `conv2`.
- **Max Pooling Layer** (`pool2`): Performs 2x2 max pooling on the output of `relu2`.
- **Flatten Layer**: Reshapes the pooled feature maps into a 1D vector.
- **Fully Connected Layer** (`fc1`): Maps the flattened vector to a 120-dimensional space.
- **ReLU Activation** (`relu3`): Applies the ReLU activation function to the output of `fc1`.
- **Fully Connected Layer** (`fc2`): Maps the 120-dimensional input to a 84-dimensional space
- **ReLU Activation** (`relu4`): Applies the ReLU activation function to the output of `fc2`.
- **Fully Connected Layer** (`fc3`): Produces a 10-dimensional output for classification.

For image classification on CIFAR-10 and CIFAR-100, we employ an adapted, larger version of the LeNet-5 model, which we call 'Large LeNet'. The network consists of the following layers, wherein $k = 10$ for CIFAR-10 and $k = 100$ for CIFAR-100.

- **Convolutional Layer** (`conv1`): Applies 2D convolution with 3 input channels, 16 output channels, and a kernel size of 3.
- **ReLU Activation** (`relu1`): Applies the ReLU activation function to the output of `conv1`.
- **Max Pooling Layer** (`pool1`): Performs 2x2 max pooling on the output of `relu1`.
- **Convolutional Layer** (`conv2`): Applies 2D convolution with 16 input channels, 32 output channels, and a kernel size of 3.
- **ReLU Activation** (`relu2`): Applies the ReLU activation function to the output of `conv2`.
- **Max Pooling Layer** (`pool2`): Performs 2x2 max pooling on the output of `relu2`.
- **Convolutional Layer** (`conv3`): Applies 2D convolution with 32 input channels, 64 output channels, and a kernel size of 3.

- **ReLU Activation** (`relu3`): Applies the ReLU activation function to the output of `conv3`.
- **Max Pooling Layer** (`pool3`): Performs 2x2 max pooling on the output of `relu3`.
- **Flatten Layer**: Reshapes the pooled feature maps into a 1D vector of size $4 \times 4 \times 64$.
- **Fully Connected Layer** (`fc1`): Maps the flattened vector to a 500-dimensional space.
- **ReLU Activation** (`relu4`): Applies the ReLU activation function to the output of `fc1`.
- **Dropout Layer** (`dropout1`): Applies dropout with $p = 0.5$ to the output of `relu4`.
- **Fully Connected Layer** (`fc2`): Produces a k-dimensional output for classification.

D.2 Mobile Neural Networks

For image classification on CIFAR-10 and CIFAR-100, we also employ the MobileNet-V3 Small architecture. The network consists of the following layers, where $k = 10$ for CIFAR-10 and $k = 100$ for CIFAR-100.

- **Convolutional Stem** (`features0`): 3 input channels, 16 output channels, kernel size 3, stride 2, padding 1, followed by BatchNorm and Hard-Swish activation.
- **Inverted Residual Block 1** (`features1`): expansion factor 1, 16 to 16 channels, kernel size 3, stride 2, SE disabled, activation ReLU.
- **Inverted Residual Block 2** (`features2`): expansion factor 4.5, 16 to 24 channels, kernel size 3, stride 2, SE disabled, activation ReLU.
- **Inverted Residual Block 3** (`features3`): expansion factor 3.67, 24 to 24 channels, kernel size 3, stride 1, SE disabled, activation ReLU.
- **Inverted Residual Block 4** (`features4`): expansion factor 4, 24 to 40 channels, kernel size 5, stride 2, SE enabled, activation Hard-Swish.
- **Inverted Residual Block 5** (`features5`): expansion factor 6, 40 to 40 channels, kernel size 5, stride 1, SE enabled, activation Hard-Swish.
- **Inverted Residual Block 6** (`features6`): expansion factor 6, 40 to 40 channels, kernel size 5, stride 1, SE enabled, activation Hard-Swish.
- **Inverted Residual Block 7** (`features7`): expansion factor 3, 40 to 48 channels, kernel size 5, stride 1, SE enabled, activation Hard-Swish.
- **Inverted Residual Block 8** (`features8`): expansion factor 3, 48 to 48 channels, kernel size 5, stride 1, SE enabled, activation Hard-Swish.
- **Inverted Residual Block 9** (`features9`): expansion factor 6, 48 to 96 channels, kernel size 5, stride 2, SE enabled, activation Hard-Swish.
- **Inverted Residual Block 10** (`features10`): expansion factor 6, 96 to 96 channels, kernel size 5, stride 1, SE enabled, activation Hard-Swish.
- **Inverted Residual Block 11** (`features11`): expansion factor 6, 96 to 96 channels, kernel size 5, stride 1, SE enabled, activation Hard-Swish.
- **Convolutional Head** (`features12`): 1×1 Conv2d from 96 to 576 channels, followed by BatchNorm and Hard-Swish.
- **Adaptive Average Pooling** (`features13`): global average pool to 1×1.
- **Conv Head** (`features14`): 1×1 Conv2d from 576 to 1024 channels, followed by Hard-Swish.
- **Flatten Layer**: reshapes the 1024×1×1 tensor to a 1024-dimensional vector.
- **Fully Connected Layer** (`classifier0`): linear 1024 to 1024, followed by Hard-Swish.
- **Dropout Layer** (`classifier2`): dropout with $p = 0.2$.
- **Fully Connected Layer** (`classifier3`): linear 1024 to k for classification.

D.2.1 Residual Neural Networks

For image classification on Imagenette, we employ the ResNet-18 residual neural network architecture, which consists of the following layers:

- **Convolutional Layer** (`conv1`): Applies a 7x7 convolution with 3 input channels, 64 output channels, and a stride of 2.
- **Batch Normalization** (`bn1`): Normalizes the output of `conv1`.
- **ReLU Activation** (`relu`): Applies the ReLU activation function to the output of `bn1`.
- **Max Pooling Layer** (`maxpool`): Performs 3x3 max pooling with a stride of 2 on the output of `relu`.
- **Residual Layer 1** (`layer1`): Contains two residual blocks, each with 64 channels.
- **Residual Layer 2** (`layer2`): Contains two residual blocks, each with 128 channels.
- **Residual Layer 3** (`layer3`): Contains two residual blocks, each with 256 channels.
- **Residual Layer 4** (`layer4`): Contains two residual blocks, each with 512 channels.
- **Average Pooling** (`avgpool`): Applies adaptive average pooling to reduce the spatial dimensions to 1x1.
- **Fully Connected Layer** (`fc`): Produces a 10-dimensional output for classification.

D.2.2 Transformer Models

For sentiment classification on IMDB Sentiment Analysis, we leverage a transformer architecture, which consists of the following layers:

- **Embedding Layer** (`embedding`): Maps input tokens to 64-dimensional embeddings.
- **Positional Encoding** (`pos_encoder`): Adds positional information to the embeddings with a maximum sequence length of 500.
- **Transformer Encoder** (`transformer_encoder`): Applies a transformer encoder with 1 layer, 4 attention heads, and a hidden dimension of 128.
- **Pooling Layer** (`pool`): Averages the transformer outputs across the sequence length.
- **Dropout Layer** (`dropout`): Applies dropout with probability 0.1 to the pooled output.
- **Fully Connected Layer** (`fc1`): Maps the 64-dimensional pooled vector to 32-dimensional space.
- **ReLU Activation** (`relu1`): Applies the ReLU activation function to the output of `fc1`.
- **Fully Connected Layer** (`fc2`): Maps the 32-dimensional input to 2 output classes.

D.2.3 Autoencoder Models

For image reconstruction on MNIST, Fashion MNIST, and CIFAR-10, we employ an autoencoder. This network consists of the following layers, where $d = 784$ for MNIST and Fashion MNIST, and $d = 3072$ for CIFAR-10:

- **Fully Connected Layer** (`fc1`): Transforms the input features from a d-dimensional space to a 128-dimensional space.
- **ReLU Activation** (`relu1`): Applies the ReLU activation function to the output of `fc1`.
- **Fully Connected Layer** (`fc2`): Reduces the 128-dimensional representation to a 32-dimensional encoded vector.
- **Fully Connected Layer** (`fc3`): Expands the 32-dimensional encoded vector back to a 128-dimensional space.
- **ReLU Activation** (`relu1`): Applies the ReLU activation function to the output of `fc3`.
- **Fully Connected Layer** (`fc4`): Maps the 128-dimensional representation back to the original d-dimensional space.
- **Sigmoid Activation** (`sigmoid1`): Applies the Sigmoid activation function to ensure the output values are between 0 and 1.

D.3 Focal Training

For the classification tasks outlined in Section 4.1, we compare learn2mix and classical training with focal loss-based neural network training (focal training). Let $\tilde{\alpha} \in [0,1]^k$ denote the vector of fixed-proportion mixing parameters, let $\mathcal{L}(\theta^t) \in \mathbb{R}^k$ denote the vector of class-wise cross entropy losses at time t, and let $\omega \in \mathbb{R}^k$ denote the vector of class-wise weighting factors, where $\forall i \in \{1,\ldots,k\}$:

$$\omega_i = \frac{[1/(\tilde{\alpha}_i N)]}{\sum_{i'=1}^{k}[1/(\tilde{\alpha}_{i'} N)]} \times k. \tag{62}$$

The vector of predicted class-wise probabilities, $p \in [0,1]^k$, is given by $p = \exp\left(-\mathcal{L}(\theta^t)\right)$, and we let $\Gamma \in \mathbb{R}_{\geq 0}$ be the focusing parameter. The focal loss at time t, $\mathcal{L}_{\text{FCL}}(\theta^t, \omega) \in \mathbb{R}_{\geq 0}$, is given by:

$$\mathcal{L}_{\text{FCL}}(\theta^t, \tilde{\alpha}) = \frac{1}{k}\sum_{i=1}^{k}(-\omega_i)(1-p_i)^{\Gamma}\log(p_i). \tag{63}$$

Per the recommendations in [Lin et al., 2017], we choose $\Gamma = 2$ in compiling the empirical results.

D.4 SMOTE Training

For the classification tasks outlined in Section 4.1, we also compare learn2mix and classical training with neural networks trained on SMOTE-oversampled datasets (SMOTE training). Let J denote the original training dataset, where the number of samples in each class, $i \in \{1,\ldots,k\}$ is given by $\tilde{\alpha}_i N$. After applying SMOTE oversampling, we obtain a new training dataset, J^{SMOTE}, with uniform class proportions, $\tilde{\alpha}_i^{\text{SMOTE}} = \frac{1}{k}$, $\forall i \in \{1,\ldots,k\}$. The total number of samples in J^{SMOTE}, is given by:

$$N^{\text{SMOTE}} = \left(\max_{i\in\{1,\ldots,k\}} \tilde{\alpha}_i N\right) \times k. \tag{64}$$

In the original training dataset, J, we use a batch size of M, resulting in $P = N/M$ total batches. For consistency with learn2mix and classical training (see Section 4.1), we perform SMOTE training on P batches of size M from the SMOTE oversampled training dataset, J^{SMOTE}, during each epoch.

D.5 IS Training

For the classification tasks outlined in Section 4.1, we compare learn2mix and classical training with importance sampling–based neural network training (IS training) adapted from [Katharopoulos and Fleuret, 2018] and [Johansson and Lindberg, 2022]. Let J denote the original training dataset, and let $\mathcal{L}_{\text{ind}}^{M}(\theta^t) \in \mathbb{R}^M$ denote the vector of individual cross-entropy losses at time t on a batch of size M drawn uniformly from J. We normalize these losses to sampling probabilities, $p_j \in [0,1]$, sample without replacement a subset of size $b = M/2$ according to $\{p_j\}$, and update the model by taking a gradient step on the average loss over that subset, where:

$$p_j = \frac{\mathcal{L}_{\text{ind},j}^{M}(\theta^t)}{\sum_{j'=1}^{M}\mathcal{L}_{\text{ind},j'}^{M}(\theta^t)}, \quad \text{and:} \quad \mathcal{L}_{\text{IS}}(\theta^t) = \frac{1}{b}\sum_{r=1}^{b}\mathcal{L}_{\text{ind},\,i_r}^{M}(\theta^t). \tag{65}$$

In the original training dataset J, we use a batch size of M, resulting in $P = N/M$ total batches, and perform IS training on P batches of size M during each epoch.

D.6 CURR Training

For the classification tasks outlined in Section 4.1, we compare learn2mix and classical training with curriculum learning–based neural network training (CURR) following the self-taught scoring and fixed exponential pacing scheme of [Hacohen and Weinshall, 2019]. Let J be the original training dataset, and denote by $\tilde{s}_j = 1 - \hat{p}_j$ the self-taught score of sample j, where $\hat{p}_j$ is the network's confidence in the correct label after preliminary convergence training on uniform mini-batches (this warm-up stage is used only to compute $\{\tilde{s}_j\}$ and is not included in our reported CURR timings, nor is any analogous stage required for learn2mix). We sort the samples by increasing $\tilde{s}_j$ (easiest first) to obtain sorted indices $\{i_1,\ldots,i_N\}$. At epoch t, let the curriculum fraction be:

$$\text{frac}(t) = \min\big(\text{starting\_percent} \times \text{inc}^{\lfloor t/\text{step\_length}\rfloor},\, 1.0\big), \tag{66}$$

with starting_percent $= 0.5$, inc $= 1.2$, and step_length $= 10$. We form a curriculum subset of size $\lfloor \mathrm{frac}(t)\, N \rfloor$ by taking the first $\lfloor \mathrm{frac}(t)\, N \rfloor$ sorted indices, and train on mini-batches of size M. The curriculum loss at time t is then:

$$\mathcal{L}_{\mathrm{CURR}}(\theta^t) = \frac{1}{\lfloor \mathrm{frac}(t)\, N \rfloor} \sum_{r=1}^{\lfloor \mathrm{frac}(t)\, N \rfloor} \mathcal{L}^1_{\mathrm{ind},\, i_r}(\theta^t), \tag{67}$$

where $\mathcal{L}^1_{\mathrm{ind},\, j}(\theta^t) \in \mathbb{R}$ is the individual cross-entropy loss on sample j, and each epoch processes $\lfloor \mathrm{frac}(t)\, N \rfloor / M$ batches of size M.

D.7 Neural Network Training Hyperparameters

The relevant hyperparameters used to train the neural networks outlined in Section D.1 are given in Table 3. All results presented in the main text were produced using these hyperparameter choices.

Table 3: Neural network training hyperparameters (grouped by task).

Dataset	**Task**	**Optimizer**	**Learning Rate (η)**	**Mixing Rate (γ)** (Learn2Mix)	**Batch Size (M)**
MNIST	Classification	Adam	0.0001	0.1	1000
Fashion MNIST	Classification	Adam	0.0001	0.5	1000
CIFAR-10	Classification	Adam	7.0e-5	0.1	1000
Imagenette	Classification	Adam	1.0e-6	0.1	100
CIFAR-100	Classification	Adam	0.0001	0.5	5000
IMDB	Classification	Adam	0.0001	0.1	500
Mean Estimation	Regression	Adam	5.0e-5	0.01	500
Wine Quality	Regression	Adam	0.0001	0.05	100
California Housing	Regression	Adam	5.0e-5	0.01	1000
MNIST	Reconstruction	Adam	0.0005	0.1	1000
Fashion MNIST	Reconstruction	Adam	1.0e-5	0.1	1000
CIFAR-10	Reconstruction	Adam	1.0e-5	0.1	1000

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The paper presents a new framework for accelerating neural network convergence. We provide comprehensive empirical results and theoretical guarantees to validate this claim.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The performance gains (large or limited) afforded by learn2mix are explicitly quantified in the empirical results section, and all methods, alongside ablation studies, are thoroughly discussed in the appendix.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: We provide theorems (with all relevant terms defined) in the main text, alongside comprehensive proofs in the appendix to verify the proposed theorems.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: Alongside the loss function and optimizer details presented in the main text, all neural network architectures and training hyperparameters are discussed and tabulated in the appendix.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: The complete code for reproducing all the empirical results for learn2mix are provided in the supplementary materials. All neural network architectures and training hyperparameters are also provided in the appendix.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: The appendix details all neural network architectures utilized to generate the results presented in the main text and in the ablation studies. All hyperparameter choices are either explicitly specified in the main text/appendix, or tabulated in the appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: All empirical results in the main text and the appendix include confidence intervals in the figures and tables to explicitly declare the statistical significance of all experiments.

   Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: The compute resources (GPUs) used to produce the empirical results in the main text are specified in the empirical results section.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The presented research conforms in all aspects with the NeurIPS Code of Ethics, and the authors have reviewed the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Accelerating neural network convergence in resource constrained regimes is an important capability to ensure fast and efficient neural network training — the adoption of learn2mix can save compute cost and accelerate training. We find no negative societal impacts of our work.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [Yes]

    Justification: All the code is provided in the appendix and the final version will be maintained in a GitHub repository by the authors. The authors contact information will also be provided in the final version to prevent misuse.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All relevant code and models used in the paper have been properly cited.

    Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The appendix contains comprehensive descriptions of all considered neural network architectures and modified datasets used to generate the empirical results.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: The paper does not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

Justification: The paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [NA]

Justification: The core method development in this research does not involve LLMs as any important, original, or non-standard components.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.