MixTraining: A Better Trade-Off Between Compute and Performance

Anonymous authors
Paper under double-blind review

Abstract

Integrating self-supervised learning (SSL) prior to supervised learning (SL) is a prevalent strategy for enhancing model performance, especially in scenarios with limited labeled data. Nonetheless, this approach inherently introduces a trade-off between computational efficiency and performance gains. Although SSL significantly improves representation learning, it necessitates an additional and often computationally expensive training phase, posing substantial overhead in resource-constrained environments. To mitigate these limitations, we propose MixTraining, a novel training framework designed to interleave multiple epochs of SSL and SL within a unified *mixtraining phase*. This phase enables a seamless transition between self-supervised and supervised objectives, facilitating enhanced synergy and improved overall accuracy. Additionally, MixTraining consolidates shared computational steps, thereby reducing redundant computations and lowering overall training latency. Comprehensive experimental evaluations demonstrate that MixTraining provides a superior trade-off between computational efficiency and model performance compared to conventional training pipelines. Specifically, on the TinyImageNet dataset using the ViT-Tiny model, MixTraining achieves an absolute accuracy improvement of 8.81% (a relative gain of 18.89%) while concurrently accelerating training by 1.29×.

1 Introduction

Self-supervised learning (SSL) has emerged as a powerful paradigm for learning general representations from unlabeled data, significantly improving performance on downstream tasks (Brown et al., 2020; He et al., 2022). The standard approach combines SSL with supervised learning (SL) into a two-phase pipeline: first training a model on unlabeled data to learn general representations, then adapting it to specific tasks using labeled data. This SSL+SL pipeline has become the de facto standard across computer vision (Chen et al., 2020; He et al., 2020, 2022; Wang et al., 2023b), natural language processing (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020; Ouyang et al., 2022; Min et al., 2023), and speech recognition (Hsu et al., 2021; Chen et al., 2022), particularly excelling in data-scarce scenarios where labeled examples are limited.

However, this two-phase pipeline introduces a fundamental trade-off between computational efficiency and model performance. While the first phase of SSL improves downstream accuracy, it requires substantial additional computation that can be prohibitive in resource-constrained environments. In fact, the SSL phase often demands hundreds or thousands of GPU-days before any task-specific training begins (Kaplan et al., 2020; Dosovitskiy et al., 2020; Chung et al., 2024). This computational overhead creates a significant barrier for practitioners, particularly those who must train models from scratch due to data privacy requirements, domain-specific constraints, or the absence of suitable pretrained models.

The abrupt transition between SSL and SL phases in the standard pipeline also presents optimization challenges. The model must suddenly shift from optimizing self-supervised objectives (e.g., reconstruction loss) to supervised objectives (e.g., classification loss), potentially causing training instability and suboptimal adaptation (Peters et al., 2019; Mosbach et al., 2020; Kumar et al., 2022). This discontinuous transition may prevent the model from fully leveraging the synergies between self-supervised and supervised learning signals.

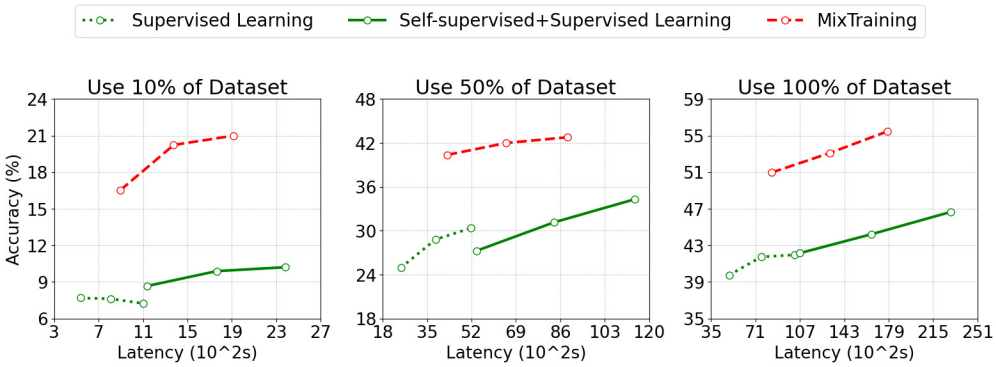

Figure 1: MixTraining demonstrates significant accuracy and computation gains over standard self-supervised learning and supervised learning pipeline across various data limitations levels (10%, 50%, and 100%). Experiments are conducted on the TinyImageNet dataset with the ViT-Tiny model; Each set of three points on the same line represents results obtained with different training durations (50, 75, and 100 epochs).

To address these limitations, we propose MixTraining, a novel training framework that introduces a dedicated mixtraining phase between the SSL and SL phases. Our key insight is that the rigid separation between SSL and SL phases in conventional pipelines is neither necessary nor optimal. As illustrated in Fig. 2, the mixtraining phase jointly optimizes both self-supervised and supervised objectives, creating a smooth transition that better preserves learned representations while adapting to downstream tasks. Our design also consolidate what would be separate forward and backward passes in the standard pipeline into unified operations (see Fig. 3 for details), substantially reducing training time while improving model performance.

As shown in Fig. 1, MixTraining consistently achieves superior accuracy and lower training latency across varying data availability levels and training durations. For example, on the full TinyImageNet dataset (Le & Yang, 2015) with reconstruction-based SSL (Hinton & Zemel, 1993; He et al., 2022), MixTraining delivers an 18.89% relative improvement in accuracy (8.81% absolute) and achieves 1.29× speedup over the standard SSL+SL pipeline. The advantage becomes even more pronounced under severe data constraints: at 10% data availability, MixTraining improves accuracy by 105.58% (relative) and reduces training latency by 1.24×.

Contributions. We introduce the MixTraining framework, which enhances both computational efficiency and model performance over the conventional SSL+SL pipeline. Our core contributions are:

- **Superior accuracy through smooth objective transition.** MixTraining introduces a mixtraining phase that creates a gradual transition between self-supervised and supervised objectives, avoiding the abrupt shift that can destabilize training in standard pipelines. This smooth transition leads to consistently higher accuracy across all evaluated settings. On TinyImageNet with ViT-Tiny, this design achieves an 18.89% relative accuracy gain over the standard SSL+SL baseline.
- **Computational efficiency via unified forward/backward passes.** By merging SSL and SL epochs into a joint optimization phase, MixTraining consolidates separate forward and backward passes over the backbone into single operations. Since backbone computations typically dominate training cost (the backbone is usually much larger than task-specific heads), this system-level optimization yields substantial speedups (e.g., 1.29× for ViT-T on TinyImageNet) while simultaneously improving accuracy.
- **Robust generalization across diverse settings.** MixTraining demonstrates consistent Pareto improvements over SSL+SL baselines across different experimental configurations, including multiple architectures (ViT, ResNet), datasets (CIFAR-10, CIFAR-100, TinyImageNet), data limitation levels (from 10% to 100%), and training epochs. The framework's modular design enables integration with existing SSL+SL pipelines with minimal modifications.

Scope and Applicability. It is important to clarify that MixTraining does not aim to replace the existing pretraining-finetuning paradigm when high-quality pretrained models are readily available. Instead,

our framework targets scenarios where practitioners must train models from scratch due to data privacy constraints, domain-specific requirements where generic pretrained models prove inadequate, or the absence of suitable pretrained models. In these settings, MixTraining provides a more efficient alternative to the standard SSL+SL pipeline, achieving superior performance while reducing training latency.

Organization. The remainder of this paper is structured as follows. Section 2 reviews related work. Section 3 introduces the MixTraining framework, detailing its design intuition and operational procedure. Section 4 presents comprehensive empirical evaluations across multiple datasets and architectures. Section 5 concludes with a discussion of limitations and future directions.

2 Related Work

Learning Pipelines and the Compute-Performance Trade-off. Early deep learning models primarily relied on supervised learning (SL) with large labeled datasets (Krizhevsky et al., 2012; Simonyan & Zisserman, 2014; He et al., 2016). More recently, self-supervised learning (SSL) has emerged as a powerful paradigm for learning transferable representations without labels, giving rise to a training pipeline that first leverages SSL for representation learning and then adapts the model to downstream tasks via supervised learning. This SSL+SL pipeline has been widely adopted across domains, including computer vision (Chen et al., 2020; He et al., 2020, 2022; Wang et al., 2023b), natural language processing (Devlin et al., 2019; Radford et al., 2019; Brown et al., 2020; Ouyang et al., 2022; Min et al., 2023), and speech recognition (Hsu et al., 2021; Chen et al., 2022). While SSL+SL consistently improves accuracy over SL alone, especially in low-label regimes, it incurs a substantial increase in compute due to the additional SSL phase (Kaplan et al., 2020; Chung et al., 2024), leading to a compute-performance trade-off between the two approaches. Recent work has begun to examine aspects of this trade-off; for example, Liu et al. (2024) explore strategies to reduce the compute cost of the SSL stage. In this work, we directly study on the compute-performance trade-off between SSL+SL and SL, and propose a new MixTraining pipeline that adds a dedicated mixtraining phase between SSL and SL, yielding Pareto improvements in both accuracy and compute efficiency over the standard SSL+SL pipeline.

Modifications to Learning Objectives. A broad class of methods modifies learning objectives to improve generalization, robustness, or stability. One important direction explores the synergy between different objectives, such as combining self-supervised learning and supervised learning objectives. Prior studies in this space have focused on domain adaptation (Pan et al., 2020; Berthelot et al., 2021), mitigating catastrophic forgetting (He et al., 2021; Mehta et al., 2023), and improving data efficiency (Zhai et al., 2019; Khosla et al., 2020; Yao et al., 2022; Zhang et al., 2024). Another related line of work mixes up data from different sources or domains to bridge distribution gaps or regularize decision boundaries (Zhang et al., 2017; Yun et al., 2019; Verma et al., 2019; Liu et al., 2022; Zou et al., 2023), often by constructing intermediate representations that incorporate information from multiple domains or tasks. Our work is inspired by these lines of research but differs in both purpose and design: we introduce a dedicated mixtraining phase between SSL and SL to smooth the transition between two learning different objectives, and we design this phase for compute efficiency by merging forward and backward passes over the shared backbone—an aspect not addressed in prior work.

Efficient Training of Deep Neural Networks. A substantial body of research has focused on reducing the computational and data costs of training deep neural networks while maintaining competitive performance. Model compression techniques (Han et al., 2015a,b; Jacob et al., 2018; Frankle & Carbin, 2018; Blalock et al., 2020; Li et al., 2023) reduce network size and complexity through pruning, weight sharing, quantization, or other architectural simplifications, thereby lowering both memory footprint and FLOPs for more efficient training and deployment. Parameter-efficient finetuning methods (Hu et al., 2021; Ding et al., 2023; Han et al., 2024) freeze most of the backbone and introduce small trainable modules or low-rank adaptations, greatly reducing optimization cost while preserving downstream performance. Data-efficient training strategies (Bengar et al., 2021; Mindermann et al., 2022; Wang et al., 2023a; Bhatt et al., 2024) accelerate convergence by prioritizing informative examples, filtering redundant or noisy data, or leveraging pretrained models for improved initialization. These approaches target model, parameter, or data-level efficiency, and are complementary to our pipeline-level strategy: MixTraining could be integrated with these techniques to further improve its efficiency.

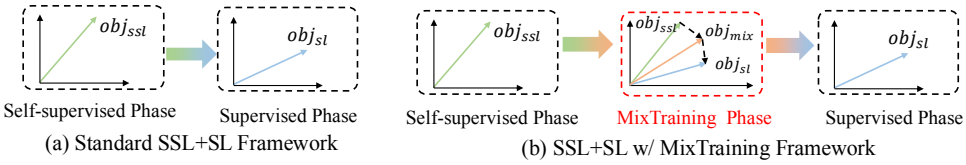

Figure 2: Comparison of MixTraining with the standard SSL+SL framework. (a) The standard SSL+SL framework features an abrupt transition from self-supervised objective ($\mathsf{obj}_{\mathsf{ssl}}$) to supervised objective ($\mathsf{obj}_{\mathsf{sl}}$). (b) Our MixTraining framework creates an dedicated mixtraining phase in the middle. The mixtraining phase optimizes towards a mixed objective ($\mathsf{obj}_{\mathsf{mix}}$), which enables a smooth transition from the self-supervised objective to the supervised objective.

3 Methodology

We briefly introduce the standard self-supervised learning and supervised learning (SSL+SL) pipeline in Section 3.1. We introduce our MixTraining framework in Section 3.2, which consists of its design intuition (Section 3.2.1) and operational procedure (Section 3.2.2).

3.1 Background: The Standard Self-supervised Learning and Supervised Learning Pipeline

The standard self-supervised learning and supervised learning pipeline consists of two separate phases: a *self-supervised learning (SSL)* phase and a *supervised learning (SL)* phase. In the self-supervised learning phase, a backbone model with a self-supervised learning head is trained to help the model learn general feature representations. Specifically, this process usually relies on learning from unlabeled data with reconstruction tasks (Hinton & Zemel, 1993; Hinton & Salakhutdinov, 2006; Michelucci, 2022) or predicting manually masked tokens (Devlin et al., 2019; He et al., 2022). The backbone model is further refined in the supervised learning phase, together with a supervised learning head. This step adapts the model to downstream tasks, usually achieving better performances than directly training the downstream tasks. Fig. 2(a) shows the standard SSL+SL framework, which has now become the go-to approach for improving the performance of AI models (Wang et al., 2023b; Min et al., 2023).

3.2 A New Framework: MixTraining

3.2.1 Design Intuition behind MixTraining

While the standard SSL+SL framework has achieved remarkable success, its self-supervised learning and supervised learning phases are completely separated, leaving limited opportunity for interaction or further optimization. To enable closer interactions between these two phases, we propose a novel MixTraining framework—as shown in Fig. 2—which effectively merges several self-supervised learning and supervised learning epochs into an additional *mixtraining phase*, featuring a smooth transition between learning objectives.

MixTraining introduces a new mixtraining phase that allows joint updates of self-supervised and supervised objectives, which is in contrast with the standard SSL+SL framework where one *first* updates the self-supervised objective and *then* updates the supervised objective. At a high level, the benefits of this phase are rooted in the dedicated joint optimization objective. The joint objective can be easily understood as a weighted average of the self-supervised objective and the supervised objective to balance these two objectives. We next explain the design intuition behind the mixtraining phase to achieve both *computation gains* and *accuracy gains*.

Accuracy Gains. Since the self-supervised learning phase aims at learning general representations and the following supervised learning phase aims at learning task-specific information, intuitively, these two phases optimize the model in different directions. The standard SSL+SL framework features an abrupt change in optimization directions during the transition from self-supervised learning and supervised learning (Fig. 2(a)), which may cause instability in model performance (Mosbach et al., 2020). Indeed, there are also studies

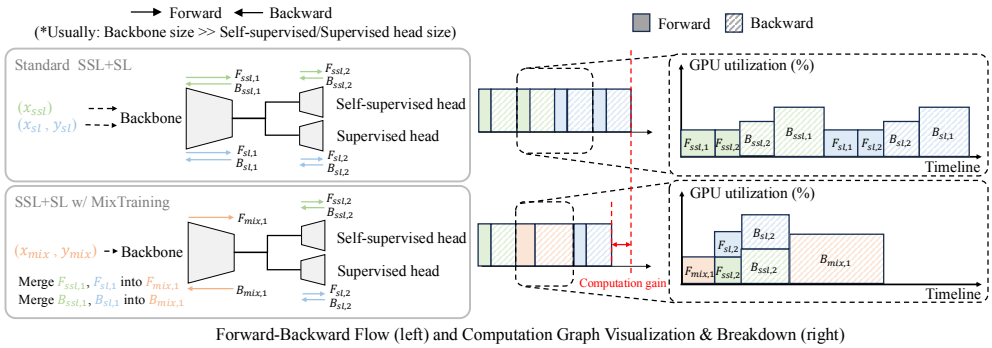

Figure 3: Comparison of MixTraining with the standard SSL+SL framework. MixTraining achieves computation gains over the standard SSL+SL framework. *Top: Standard SSL+SL.* Data first goes through a self-supervised learning pass ($F_{\mathsf{ssl},1} \to F_{\mathsf{ssl},2} \to B_{\mathsf{ssl},2} \to B_{\mathsf{ssl},1}$) and then goes through a supervised learning pass ($F_{\mathsf{sl},1} \to F_{\mathsf{sl},2} \to B_{\mathsf{sl},2} \to B_{\mathsf{sl},1}$). *Bottom: MixTraining.* We merge two forward passes ($F_{\mathsf{ssl},1}$ and $F_{\mathsf{sl},1}$) over the backbone model together into a single pass $F_{\mathsf{mix},1}$, and use its result for both self-supervised head and supervised head; the backward passes ($B_{\mathsf{ssl},1}$ and $B_{\mathsf{sl},1}$) are merged into $B_{\mathsf{mix},1}$ (bottom left). Our modifications reduce computation requirements and allow better parallelization (bottom right).

show that, in certain settings, supervised learning in the second phase can lead to worse model performance (Peters et al., 2019; Kumar et al., 2022). In our MixTraining framework, the mixtraining phase creates a middle ground, i.e., a weighted combination of two objectives, allowing a rather smooth transition from the self-supervised learning objectives to the supervised learning objective, as illustrated in Fig. 2(b). We hypothesize that such a smooth transition avoids instability in phase transition, thus allowing the model to better adapt to the target task and achieve higher accuracy. We validate the smooth-transition intuition by measuring the cosine similarity of the last-backbone-layer gradients induced by different learning objectives. The average cosine similarity between SSL and SL objectives is only 0.018, indicating that these two objectives are nearly orthogonal. In contrast, the cosine similarity between SSL and MixTraining objectives is 0.791, and between SL and MixTraining objectives is 0.598, suggesting that the MixTraining objective indeed provides a smooth transition from SSL to SL. Our empirical results in Section 4 further support the hypothesis that smooth transitions lead to higher accuracy.

Computation Gains. In the standard SSL+SL framework, we first run self-supervised learning passes with data $(x_{\mathsf{ssl}}, y_{\mathsf{ssl}})$, and then run supervised learning passes with data $(x_{\mathsf{sl}}, y_{\mathsf{sl}})$. This process involves forward/backward passes of both $(x_{\mathsf{ssl}}, y_{\mathsf{ssl}})$ and $(x_{\mathsf{sl}}, y_{\mathsf{sl}})$, and strictly follows a *sequential order* to compute each sub-processes (top part of Fig. 3). In contrast, MixTraining aims to jointly optimize self-supervised and supervised objectives. Specifically, it "merges" $(x_{\mathsf{ssl}}, y_{\mathsf{ssl}})$ and $(x_{\mathsf{sl}}, y_{\mathsf{sl}})$ into a *mixed data* $(x_{\mathsf{mix}}, y_{\mathsf{mix}})$ and thus merging the separate forward passes over the backbone model into a single pass, and use its result for both self-supervised head and supervised head; from the computation aspect, we also merge the backward passes of self-supervised learning and supervised learning tasks over the backbone model into a single pass (bottom left part of Fig. 3). Since the size of the backbone model is usually much larger than the size of self-supervised and supervised heads (He et al., 2022; Du et al., 2021; Yang et al., 2023), the merge of forward/backward passes over the backbone model allows us to reduce computation compared to the synchronous setting substantially. Additionally, the merge of forward/backward passes over the backbone model allows better parallelization of forward/backward passes over the self-supervised and supervised heads (right part of Fig. 3), which further speeds up the computation. Empirically, we find the runtime of a single merged mixtraining pass (0.317s) is lower than the combined runtime of an SL pass and an SSL pass (0.543s = 0.248s + 0.295s), validating our intuition.

3.2.2 The MixTraining Procedure

In this section, we introduce the operational procedure of our MixTraining framework in detail. Besides the number of self-supervised learning epoch e_{ssl} and the number of supervised learning epoch e_{sl}, MixTraining takes as input a hyperparameter MIX-RATIO $\rho \in [0, 1]$ to determine the number of self-supervised

Algorithm 1 The MixTraining Framework

Input: Self-supervised learning epoch e_{ssl}, Supervised learning epoch e_{sl}, MIX-RATIO $\rho \in [0, 1]$.

Initialize model parameters θ.
Calculate mixtraining epoch e_{mix} via Eq. (1).
▷ Vanilla Self-supervised Learning Phase
for $e = 1$ to $e_{\mathsf{ssl}} - e_{\mathsf{mix}}$ **do**
    Conduct vanilla self-supervised learning phase *w.r.t.* self-supervised loss $\ell_{\mathsf{ssl}}(x; \theta)$ to optimize θ.
▷ MixTraining Phase
for $e = 1$ to e_{mix} **do**
    Optimize the model parameter θ with respect to the joint optimization objective as in Eq. (3).
▷ Vanilla Supervised Learning Phase
for $e = 1$ to $e_{\mathsf{sl}} - e_{\mathsf{mix}}$ **do**
    Conduct vanilla supervised learning phase *w.r.t.* supervised loss $\ell_{\mathsf{sl}}(f(x), y; \theta)$ to optimize θ.

learning/supervised learning epochs e_{mix} to be merged into the mixtraining phase. We set

$$e_{\mathsf{mix}} = \lfloor \rho \min(e_{\mathsf{ssl}}, e_{\mathsf{sl}}) \rfloor. \tag{1}$$

Our MixTraining framework then operates by running (i) the vanilla self-supervised learning phase for $e_{\mathsf{ssl}} - e_{\mathsf{mix}}$ epochs, (ii) the mixtraining phase for e_{mix} epochs, and (iii) the vanilla supervised learning phase for $e_{\mathsf{sl}} - e_{\mathsf{mix}}$ epochs, as shown in Algorithm 1.

Since the self-supervised learning and supervised learning phases are standard, in the following, we mainly discuss the mixtraining phase. In the mixtraining phase, we (i) design a mixing function g to generate *a mixed dataset*, and (ii) design *a joint optimization objective* as supervision signal. We next highlight the design choice for these two parts.

The Mixed Dataset. The goal of creating a mixed dataset $\mathcal{D}_{\mathsf{mix}} = g(\mathcal{D}_{\mathsf{ssl}}, \mathcal{D}_{\mathsf{sl}})$ is to extract information stored in self-supervised learning dataset $\mathcal{D}_{\mathsf{ssl}} = \{x_i\}_i$ and supervised learning dataset $\mathcal{D}_{\mathsf{sl}} = \{(x_i, y_i)\}_i$, featuring a smooth transition between two learning objectives Fig. 2 (b). In the simple case where self-supervised learning and supervised learning use the same input (i.e., x_i), we can simply set $g(\mathcal{D}_{\mathsf{ssl}}, \mathcal{D}_{\mathsf{sl}}) = \mathcal{D}_{\mathsf{sl}}$. We remark that the importance of this simple case is usually overlooked: conducting self-supervised learning and supervised learning on the same ImageNet dataset allows one to boost the top-1 classification accuracy from 82.5% to 84.9%, without using extra data (He et al., 2022).

We next discuss the general case where the self-supervised learning dataset is not the same as the supervised learning dataset, i.e., $\mathcal{D}_{\mathsf{ssl}} \neq \mathcal{D}_{\mathsf{sl}}$. Inspired by the mixup method in machine learning to improve the generalization and robustness to adversarial examples (Zhang et al., 2017), we consider a *randomized* mixing function g, which randomly mixes up data points from both datasets. Specifically, we set

$$\mathcal{D}_{\mathsf{mix}} = g(\mathcal{D}_{\mathsf{ssl}}, \mathcal{D}_{\mathsf{sl}}) = \{(x_{\mathsf{mix}}, y_{\mathsf{sl}}) : x_{\mathsf{mix}} = \lambda\, x_{\mathsf{sl}} + (1 - \lambda)\, x_{\mathsf{ssl}}, (x_{\mathsf{sl}}, y_{\mathsf{sl}}) \in \mathcal{D}_{\mathsf{sl}}, x_{\mathsf{ssl}} \in \mathcal{D}_{\mathsf{ssl}}\}, \tag{2}$$

where for each $(x_{\mathsf{sl}}, y_{\mathsf{sl}}) \in \mathcal{D}_{\mathsf{sl}}$, we randomly draw a self-supervised data point x_{ssl} from $\mathcal{D}_{\mathsf{ssl}}$ and generate a mixup feature $\lambda\, x_{\mathsf{sl}} + (1 - \lambda)\, x_{\mathsf{ssl}}$, where λ is a hyperparameter of user's choice (we set $\lambda = 0.5$ in our experiments to balance the contribution from both datasets). We adopt the supervised label y_{sl} to provide supervised signal since the self-supervised part typically doesn't require labels.

The Joint Optimization Objective. Let θ denote the model parameters. Let $\ell_{\mathsf{ssl}}(x; \theta)$ denote the self-supervised loss, e.g., MSE reconstruction loss for masked autoencoders (He et al., 2022), and let $\ell_{\mathsf{sl}}(f(x), y; \theta)$ denote the supervised loss, e.g., cross-entropy for image classification (Krizhevsky et al., 2012). Let $\mathcal{D}_{\mathsf{ssl}}$ represent the SSL dataset, which contains examples $\{x_i\}_i$, and $\mathcal{D}_{\mathsf{sl}}$ represent the SL dataset, which also contains examples $\{(x_i, y_i)\}_i$. These datasets serve as the sources for self-supervised and supervised learning, respectively. The classical SSL+SL framework first optimizes $\min_\theta \mathbb{E}_{x,y\sim\mathcal{D}_{\mathsf{ssl}}}[\ell_{\mathsf{ssl}}(x; \theta)]$ and then optimizes $\min_\theta \mathbb{E}_{x,y\sim\mathcal{D}_{\mathsf{sl}}}[\ell_{\mathsf{sl}}(f(x), y; \theta)]$.

To integrate self-supervised and supervised learning objectives, MixTraining considers a weighted combination of these two objectives and optimizes the following goal:

$$\min_{\theta} \mathbb{E}_{(x,y)\sim\mathcal{D}_{\mathsf{mix}}}[\alpha\, \ell_{\mathsf{ssl}}(x;\theta) + (1-\alpha)\, \ell_{\mathsf{sl}}(f(x), y;\theta)], \tag{3}$$

where $\mathcal{D}_{\mathsf{mix}}$ is the mixed dataset and $\alpha \in (0,1)$ is a hyperparameter loss-ratio designed to balance the focus between learning general representations (by optimizing self-supervised loss $\ell_{\mathsf{ssl}}(x;\theta)$) and achieving specific target (by optimizing supervised loss $\ell_{\mathsf{sl}}(f(x), y;\theta)$). Eq. (3) can be naturally extended to include a continuously annealed loss weight α that interpolates between the SSL and SL objectives. For simplicity, we retain the current form of Eq. (3); our ablation study in Section 4.4.4 shows that both linear and cosine annealing yield comparable performance.

4 Experiments

4.1 Setups

Datasets. We conduct experiments on standard computer vision datasets, including CIFAR-10 (Krizhevsky et al., 2009), CIFAR-100 (Krizhevsky et al., 2009), and TinyImageNet (Le & Yang, 2015).

Models. Our model consists of three components: a shared backbone model, a classification head for SL, and a reconstruction head for SSL. We use the standard ViT-Tiny (ViT-T) (Dosovitskiy et al., 2020; Wu et al., 2022) or standard ResNet-18 (He et al., 2016) as the backbone and the classification head. For the reconstruction head, we use a masked autoencoder (MAE) decoder (He et al., 2022) of depth 2 for ViT-T and a two-layer transposed convolution (Zeiler et al., 2010) for ResNet-18. MAE is used as the reconstruction task for the ViT-T model, and the standard autoencoder (AE) (Hinton & Zemel, 1993) is used for the ResNet-18 model.

Baselines. We evaluate the performance of our algorithm (Algorithm 1) against the following baselines:

- *Supervised learning (SL).* Conduct standard supervised learning on the backbone and the classification head with cross-entropy loss for e_{sl} epochs.
- *Self-supervised learning + supervised learning (SSL+SL).* Conduct self-supervised learning on the backbone and the reconstruction head with MSE loss for e_{ssl} epochs and then conduct standard supervised learning with cross-entropy loss for e_{sl} epochs.

The comparison between SL and SSL+SL reflects the compute-performance trade-off: SSL+SL achieves better performance at the cost of the added computation in the SSL-alone step. We aim to provide a better compute-performance trade-off with MixTraining, i.e., a Pareto improvement over the SSL+SL baseline. We adopt a standard set of augmentations following the practices of He et al. (2016) and Charisoudis et al. (2023), as detailed in Appendix A.3. We remark that our goal is not to maximize benchmark accuracy, but to enable a controlled and reproducible comparison across different training pipelines.

Evaluation Metrics. For each method, we measure its performance by the accuracy on the downstream classification task and its computation cost by the training latency (i.e., the total wall-clock time). We report the average accuracy and latency over 4 runs with different random seeds. We calculate the speedups of our method as the ratio between the latency of SSL+SL and MixTraining.

Other Implementation Details. We conduct experiments across various data limitation levels by randomly select a fraction of p data points from the original dataset; we choose $p \in \{10\%, 25\%, 50\%, 75\%, 100\%\}$. In our main experiments, we set training epoch $e_{\mathsf{sl}} = e_{\mathsf{ssl}} = 100$, loss-ratio $\alpha = 0.5$, and mix-ratio $\rho = 0.5$; we conduct detailed parameter studies for these quantities in Section 4.3. We defer additional experimental details to Appendix A.

Table 1: Accuracy and latency comparison using the ViT-T model. We evaluate performance across various data limitation levels $p \in \{10\%, 25\%, 50\%, 75\%, 100\%\}$. The highest accuracy in each setting is highlighted in **bold**). MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in 15 out of 15 settings.**

Datasets	Methods	10%		25%		50%		75%		100%	
		Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
TinyImageNet	**SL**	7.24%	1102.42	20.91%	2619.14	30.36%	5181.44	36.98%	7709.21	41.96%	10257.12
	SSL+SL	10.21%	2382.06	21.78%	5813.27	34.30%	11449.43	42.73%	17296.22	46.65%	22917.29
	MixTraining	**20.99%**	1913.75	**31.43%**	4516.69	**42.77%**	8868.88	**49.30%**	13304.60	**55.46%**	17795.47
CIFAR-10	**SL**	53.40%	589.44	66.03%	1335.79	75.00%	2593.27	79.22%	3844.27	81.52%	5155.59
	SSL+SL	56.79%	1253.65	67.95%	2774.24	77.06%	5354.08	82.11%	7993.72	84.69%	10730.52
	MixTraining	**60.45%**	962.16	**72.61%**	2206.13	**79.95%**	4247.78	**83.97%**	6234.03	**87.13%**	8274.45
CIFAR-100	**SL**	19.05%	609.11	31.49%	1346.96	42.50%	2580.08	48.97%	3814.54	54.72%	5022.96
	SSL+SL	22.27%	1279.78	34.93%	2882.62	46.02%	5551.88	53.79%	8226.83	57.92%	10960.08
	MixTraining	**25.19%**	1010.94	**38.55%**	2226.92	**48.60%**	4298.75	**55.84%**	6354.76	**59.95%**	8457.93

Table 2: Accuracy and latency comparison using the ResNet-18 model. We evaluate performance across various data limitation levels $p \in \{10\%, 25\%, 50\%, 75\%, 100\%\}$. The highest accuracy in each setting is highlighted in **bold** (if gain $\geq 0.3\%$). MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in 13 out of 15 settings.** In the other 2 settings, MixTraining maintains comparable or slightly better accuracy while reducing training latency.

Datasets	Methods	10%		25%		50%		75%		100%	
		Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
TinyImageNet	**SL**	21.82%	730.56	32.42%	903.32	43.28%	1645.96	49.02%	2135.75	52.67%	2562.98
	SSL+SL	22.16%	1510.34	32.89%	2577.98	43.58%	3893.67	48.96%	5770.75	52.93%	7412.92
	MixTraining	**22.66%**	1298.08	**34.37%**	2392.82	**44.62%**	3591.22	**49.43%**	5276.08	52.98%	6329.25
CIFAR-10	**SL**	58.91%	393.29	72.63%	567.39	81.19%	964.66	84.81%	1243.09	86.89%	1524.69
	SSL+SL	64.43%	879.90	76.42%	1169.64	81.23%	1496.17	84.97%	1716.29	86.87%	2177.03
	MixTraining	**67.53%**	824.77	**77.68%**	1028.61	**83.49%**	1384.42	85.15%	1543.96	**87.20%**	2011.36
CIFAR-100	**SL**	25.49%	461.37	36.79%	671.45	48.71%	1066.57	54.17%	1204.52	58.24%	1546.02
	SSL+SL	27.21%	841.69	40.44%	1102.13	48.91%	1439.27	53.91%	1687.70	58.33%	2071.19
	MixTraining	**27.94%**	781.07	**41.92%**	1066.44	**50.63%**	1349.06	**56.04%**	1577.67	**59.86%**	1929.84

4.2 Main Results

We present our main experiment results in this section. Results across different data limitation levels are provided in Section 4.2.1, and results with varying SSL and SL datasets are presented in Section 4.2.2.

4.2.1 Performance Analysis across Various Data Limitation Levels

In this section, we evaluate the performance of MixTraining across various data limitation levels. We conduct experiments on CIFAR-10, CIFAR-100, and TinyImageNet datasets using different model architectures: we present results with the ViT-T model in Table 1, and results with the ResNet-18 model in Table 2. As expected, the comparison between SSL+SL and SL reflects a compute-performance trade-off: SSL+SL achieves better accuracy at the cost of higher training latency. Our MixTraining method achieves a Pareto improvement over the SSL+SL baseline: **MixTraining achieves higher accuracy and lower latency in 28 out of 30 settings.** In the other 2 settings, MixTraining maintains comparable or slightly better accuracy while reducing training latency.

Compared to baselines, MixTraining achieves significant accuracy gains: for instance, on the full TinyImageNet dataset with the ViT-T model, MixTraining achieves 18.89% relative accuracy gain (8.81% absolute accuracy gain) over SSL+SL and 32.17% relative accuracy gain (13.50% absolute accuracy gain) over SL. The accuracy gains are more significant under limited data: for instance, on the TinyImageNet dataset and at data limitation level of 10%, MixTraining achieves 105.58% relative accuracy gain (10.78% absolute accuracy gain) over SSL+SL and 189.92% relative accuracy gain (13.75% absolute accuracy gain) over SL. MixTraining also saves more compute (reflected as training latency) compared to SSL+SL. For instance, on the full TinyImageNet dataset with the ViT-T model, MixTraining acheives 1.29× speedup compared to SSL+SL. These results show that MixTraining provides a better compute-performance trade-off compared to the standard SSL+SL pipeline across various data limitation levels.

Table 3: Accuracy and latency comparison with different self-supervised and supervised datasets using the ViT-T model. We use TinyImageNet as the self-supervised learning dataset and use CIFAR-10/CIFAR-100 as the supervised learning dataset. The highest accuracy in each setting is highlighted in **bold**. MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in both settings.**

Methods	TinyImageNet to CIFAR-10		TinyImageNet to CIFAR-100	
	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
SL	81.76%	5203.76	54.22%	5149.24
SSL+SL	84.48%	16815.33	56.22%	16582.02
MixTraining	**89.19%**	12098.50	**58.49%**	13742.15

Table 4: Accuracy and latency comparison with different self-supervised and supervised datasets using the ResNet-18 model. We use TinyImageNet as the self-supervised learning dataset and use CIFAR-10/CIFAR-100 as the supervised learning dataset. The highest accuracy in each setting is highlighted in **bold**. MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in both settings.**

Methods	TinyImageNet to CIFAR-10		TinyImageNet to CIFAR-100	
	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
SL	85.41%	1532.46	62.49%	1496.27
SSL+SL	88.79%	6247.53	66.19%	6364.51
MixTraining	**89.95%**	4865.75	**67.74%**	4921.08

4.2.2 Self-Supervised Learning and Supervised Learning on Different Datasets

In this section, we evaluate the performance of MixTraining in settings where the self-supervised learning dataset and supervised learning dataset are different. We perform self-supervised learning on the TinyImageNet dataset and supervised learning on the CIFAR-10 or CIFAR-100 datasets. We present the results using both the ViT-T model (Table 3) and the ResNet-18 model (Table 4). The comparison between SSL+SL and SL still reflects the compute-performance trade-off: SSL+SL achieves better accuracy at the cost of higher training latency. Our MixTraining method achieves a Pareto improvement over the SSL+SL baseline: **MixTraining achieves higher accuracy and lower latency in all 4 settings.**

Compared to baselines, MixTraining achieves significant accuracy gains: for instance, on the CIFAR-10 dataset with the ViT-T model, MixTraining achieves 5.58% relative accuracy gain (4.71% absolute accuracy gain) over SSL+SL and 9.09% relative accuracy gain (7.43% absolute accuracy gain) over SL. In terms of computation cost (reflected as training latency), MixTraining also saves more compute compared to SSL+SL. For instance, on the CIFAR-10 dataset with the ViT-T model, MixTraining can achieve 1.39× speedup compared to SSL+SL. These results show that MixTraining can provide a better compute-performance trade-off compared to the standard SSL+SL pipeline, even when using different datasets for self-supervised learning and supervised learning.

4.3 Parameter Study

In this section, we explore the impacts of varying loss-ratio α, mix-ratio ρ, and training epochs e_{ssl} and e_{sl} for MixTraining.

4.3.1 Impact of loss-ratio α

We study the impact of varying hyperparameter loss-ratio α on model accuracy in this section. We conduct experiments with $\alpha \in \{0.01, 0.1, 0.5, 0.9, 0.99\}$ and report the accuracy in Table 5; we didn't report the latency since varying α doesn't change the overall computation cost. As shown in Table 5, loss-ratio $\alpha = 0.5$

Table 5: Parameter study on LOSS-RATIO α. We train the ViT-T model with full data and set $\rho = 0.75$; other experimental settings remain the same as Table 1. The best and second-best accuracies are highlighted in **bold** and underline, respectively.

α	TinyImageNet	CIFAR-10	CIFAR-100
0.01	41.58%	79.61%	52.58%
0.1	<u>53.17%</u>	82.19%	55.21%
0.5	**53.86%**	<u>85.75%</u>	**58.88%**
0.9	48.18%	**86.13%**	<u>58.47%</u>
0.99	42.67%	83.75%	56.04%

Table 6: Parameter study on MIX-RATIO ρ. We train the ViT-T model with full data and set $\alpha = 0.5$; other experimental settings remain the same as Table 1. The best and second-best accuracies are highlighted in **bold** and underline, respectively.

ρ	TinyImageNet	CIFAR-10	CIFAR-100
0.25	<u>55.08%</u>	85.49%	**60.98%**
0.50	**55.40%**	<u>85.54%</u>	<u>60.15%</u>
0.75	53.86%	**85.75%**	58.88%
1.00	49.21%	84.33%	55.80%

generally leads to good accuracy gains—either achieving the highest accuracy (2 out of 3) or achieving the second-best accuracy (1 out of 3). This indicates that a well-chosen α should appropriately balance self-supervised learning and supervised learning objectives in MIXTRAINING. An $\alpha = 0.5$ allows the model to focus on both self-supervised learning and supervised learning objectives. Therefore, we recommend setting $\alpha = 0.5$ in experiments.

4.3.2 Impact of mix-ratio ρ

We study the impact of varying the hyperparameter MIX-RATIO ρ on model accuracy in this section. We conduct experiments with $\rho \in \{0.25, 0.5, 0.75, 1\}$ and report the accuracy in Table 6; we didn't report latency since it is straightforward to see that large ρ reduces latency (Section 3.2). Values of $\rho = 0.5$ or $\rho = 0.25$ generally leads to good accuracy gains. Since larger ρ reduces latency, its selection should be guided by individual priorities, such as accelerating the learning process or achieving higher accuracy.

4.3.3 Impact of Training Epochs

We study the impact of varying training epochs e_{ssl}, e_{sl} on model accuracy and training latency. For simplicity, we set $e_{\mathsf{ssl}} = e_{\mathsf{sl}}$ and choose its value from {50, 75, 100}.[1] We conduct experiments across various choices of data limitation levels $p \in \{10\%, 25\%, 50\%, 75\%, 100\%\}$. Due to space limitations, we present results for learning with full data ($p = 100\%$) in the main content, and defer the complete results to Appendix A.4.

We present results with full data in Table 7. We observe that, compared to SL and SSL+SL, MIXTRAINING generally yields greater accuracy gains for added computation; for example, doubling the training epochs from 50 to 100 results in a larger improvement in accuracy. Interestingly, MIXTRAINING with $e_{\mathsf{ssl}} = e_{\mathsf{sl}} = 50$ outperforms SL with $e_{\mathsf{sl}} = 100$ and SSL+SL with $e_{\mathsf{ssl}} = e_{\mathsf{sl}} = 100$ in terms of both accuracy and compute: MIXTRAINING simultaneously achieves higher accuracy and lower training latency. MIXTRAINING consistently acheives a Pareto improvement over the SSL+SL baseline: **MixTraining achieves higher accuracy and**

[1] When $e_{\mathsf{ssl}} = e_{\mathsf{sl}} = 100$, all validation loss curves have converged. Some validation losses may not fully converge when the number of training epochs is reduced; however, this is intentional, as we aim to study compute-performance trade-offs under compute-constrained settings.

Table 7: Parameter study on training epochs with full data using the ViT-T model. Results under other data limitation levels $p \in \{10\%, 25\%, 50\%, 75\%\}$ are deferred to Appendix A.4. The highest accuracy in each setting is highlighted in **bold**. MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in 9 out of 9 settings.**

Datasets	Computation (epoch)	SL		SSL+SL		MixTraining	
		Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
TinyImageNet	50	39.75%	4957.82	42.16%	10714.36	**50.96%**	8410.97
	75	41.74%	7560.40	44.22%	16498.39	**53.09%**	13111.49
	100	41.96%	10257.12	46.65%	22917.29	**55.46%**	17795.47
CIFAR-10	50	80.78%	2489.38	83.75%	5365.33	**85.60%**	4193.42
	75	81.59%	3818.88	84.48%	8070.96	**86.79%**	6515.22
	100	81.52%	5155.59	84.69%	10730.52	**87.13%**	8274.45
CIFAR-100	50	54.25%	2713.55	56.91%	5763.71	**58.67%**	4421.63
	75	54.71%	3881.84	56.95%	8104.50	**59.11%**	6459.23
	100	54.72%	5022.96	57.92%	10960.08	**59.95%**	8457.93

Table 8: Accuracy comparison in semi-supervised setting with 50% labeled data and 50% unlabeled data. We conduct experiments on the CIFAR-10 dataset with the ViT-T model. MixTraining achieves the **best** result over the baselines.

Metric	SL	SSL+SL	MixTraining
Accuracy↑	75.59	83.89	**87.35**

lower latency in all 9 out of 9 settings. Combining results in Appendix A.4 with data limitations levels $p \in \{25\%, 50\%, 75\%, 100\%\}$, **MixTraining achieves higher accuracy and lower latency in all 45 out of 45 settings.** These improvements again demonstrate that MixTraining is a superior training pipeline compared to the standard SSL+SL approach.

4.4 Additional Analyses

In this section, we provide additional analyses on MixTraining. Our experiments below are conducted on the CIFAR-10 dataset using the ViT-T model.

4.4.1 Learning in Semi-Supervised Settings

We extend our experiments to the semi-supervised setting where 50% data are labeled and the other 50% data are unlabeled. In this setup, the SL baseline trains only on the labeled data, while the SSL+SL baseline first applies reconstruction-based SSL on all data and then performs supervised learning on the labeled subset. In our MixTraining framework, the SSL and SL phases remain the same as in the standard SSL+SL pipeline, while the intermediate mixtraining phase randomly mixes labeled data and train using the objective in Eq. (3). As shown in Table 8, MixTraining achieves the highest accuracy in this semi-supervised setting, with even larger gains over the baselines. This further highlights the effectiveness of MixTraining.

4.4.2 Learning with Data Noise

We simulate a real-world environment by adding random label noise to the dataset: for each data point, with probability $p = 0.2$, we randomly change its label. Results in Table 9 show that our MixTraining framework continues to achieve the best performance with data noise, verifying its robustness in practical settings.

4.4.3 MixTraining is More Powerful Than Standard Data Mixups

To examine the advantages of MixTraining over standard data mixups (Zhang et al., 2017), we further equip the SL and SSL+SL baselines with mixup data augmentation (Table 10). While mixup slightly improves the baselines' performance, our proposed MixTraining method consistently achieves the best performance

Table 9: Accuracy comparison on CIFAR-10 (with ViT-T) with random data noise. We evaluate performance across various data limitation levels $p \in \{10\%, 25\%, 50\%, 75\%, 100\%\}$. The highest accuracy is highlighted in **bold**. MixTraining achieves the best performance over the baselines.

Methods	10%	25%	50%	75%	100%
SL	55.80%	64.64%	72.96%	78.33%	81.27%
SSL+SL	56.61%	68.45%	72.94%	80.98%	84.17%
MixTraining	**61.18%**	**70.44%**	**75.97%**	**84.01%**	**86.13%**

Table 10: Accuracy comparison on CIFAR-10 (with ViT-T) with mixup augmentations applied to the SL and SSL+SL baselines. We evaluate performance across various data limitation levels $p \in \{10\%, 25\%, 50\%, 75\%, 100\%\}$. We conduct mixup for both of the other two baselines. The highest accuracy is highlighted in **bold**. MixTraining achieves the best performance over the baselines.

Methods	10%	25%	50%	75%	100%
SL w/ mixup	56.45%	68.41%	76.18%	80.91%	83.87%
SSL+SL w/ mixup	57.92%	70.08%	77.50%	82.83%	85.81%
MixTraining	**60.45%**	**72.61%**	**79.95%**	**83.97%**	**87.13%**

across all data levels. This demonstrates that MixTraining provides advantages that standard mixup alone cannot achieve—the gains stem from the unified MixTraining pipeline described in Section 3.2, rather than from augmentation effects.

4.4.4 Smooth Interpolation of α in the Mixtraining Phase

While Eq. (3) uses a fixed loss weight $\alpha \in [0, 1]$, it can be naturally extended to include a continuously annealed weight α that interpolates between the SSL and SL objectives. In Table 11, we conduct ablations with both linear and cosine annealing schedules. Since both variants achieve comparable performance to our original MixTraining approach, this indicates that the key advantage lies in the existence of the mixtraining phase itself, rather than in the specific scheduling of the weight α.

Table 11: Ablation of linear and cosine annealing schedules for loss weight α in MixTraining. We conduct experiments on the CIFAR-10 dataset with the ViT-T model. All variants achieve comparable performance.

Methods	10%	25%	50%	75%	100%
Algorithm 1	60.45%	72.61%	79.95%	83.97%	87.13%
Algorithm 1 w/ linear	61.49%	72.21%	79.07%	84.82%	86.90%
Algorithm 1 w/ cosine	60.65%	72.03%	79.16%	83.40%	87.00%

5 Conclusion

We introduced MixTraining, an innovative framework that interleaves multiple self-supervised learning (SSL) and supervised learning (SL) epochs within a unified training phase, enabling a smooth transition between the two learning objectives. By enhancing the synergy between SSL and SL, MixTraining achieves significant accuracy improvements while consolidating shared computation steps to reduce computational cost. Extensive experiments demonstrate that MixTraining offers a superior compute-performance trade-off compared to the conventional SSL+SL pipeline: MixTraining achieves substantial improvements in model accuracy while significantly reducing training latency. Given our limited compute resources, we focus our study on relatively smaller-scale settings and provide extensive ablation analyses to support our findings. We leave large-scale evaluations for future work.

References

Javad Zolfaghari Bengar, Joost van de Weijer, Bartlomiej Twardowski, and Bogdan Raducanu. Reducing label effort: Self-supervised meets active learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 1631–1639, 2021.

David Berthelot, Rebecca Roelofs, Kihyuk Sohn, Nicholas Carlini, and Alex Kurakin. Adamatch: A unified approach to semi-supervised learning and domain adaptation. *arXiv preprint arXiv:2106.04732*, 2021.

Gantavya Bhatt, Yifang Chen, Arnav M Das, Jifan Zhang, Sang T Truong, Stephen Mussmann, Yinglun Zhu, Jeffrey Bilmes, Simon S Du, Kevin Jamieson, et al. An experimental design framework for label-efficient supervised finetuning of large language models. *arXiv preprint arXiv:2401.06692*, 2024.

Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

Athanasios Charisoudis, Simon Ekman von Huth, and Emil Jansson. [re] masked autoencoders are small scale vision learners: A reproduction under resource constraints. In *ML Reproducibility Challenge 2022*, 2023.

Sanyuan Chen, Chengyi Wang, Zhengyang Chen, Yu Wu, Shujie Liu, Zhuo Chen, Jinyu Li, Naoyuki Kanda, Takuya Yoshioka, Xiong Xiao, et al. Wavlm: Large-scale self-supervised pre-training for full stack speech processing. *IEEE Journal of Selected Topics in Signal Processing*, 16(6):1505–1518, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pp. 1597–1607. PMLR, 2020.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 4171–4186, 2019.

Ning Ding, Yujia Qin, Guang Yang, Fuchao Wei, Zonghan Yang, Yusheng Su, Shengding Hu, Yulin Chen, Chi-Min Chan, Weize Chen, et al. Parameter-efficient fine-tuning of large-scale pre-trained language models. *Nature machine intelligence*, 5(3):220–235, 2023.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Xianzhi Du, Barret Zoph, Wei-Chih Hung, and Tsung-Yi Lin. Simple training strategies and model scaling for object detection. *arXiv preprint arXiv:2107.00057*, 2021.

Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015a.

Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015b.

Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

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*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 1121–1133, 2021.

Geoffrey E Hinton and Ruslan R Salakhutdinov. Reducing the dimensionality of data with neural networks. *science*, 313(5786):504–507, 2006.

Geoffrey E Hinton and Richard Zemel. Autoencoders, minimum description length and helmholtz free energy. *Advances in neural information processing systems*, 6, 1993.

Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM transactions on audio, speech, and language processing*, 29:3451–3460, 2021.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2704–2713, 2018.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Prannay Khosla, Piotr Teterwak, Chen Wang, Aaron Sarna, Yonglong Tian, Phillip Isola, Aaron Maschinot, Ce Liu, and Dilip Krishnan. Supervised contrastive learning. *Advances in neural information processing systems*, 33:18661–18673, 2020.

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. *Advances in neural information processing systems*, 25, 2012.

Ananya Kumar, Aditi Raghunathan, Robbie Jones, Tengyu Ma, and Percy Liang. Fine-tuning can distort pretrained features and underperform out-of-distribution. *arXiv preprint arXiv:2202.10054*, 2022.

Ya Le and Xuan Yang. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Zhuo Li, Hengyi Li, and Lin Meng. Model compression for deep neural networks: A survey. *Computers*, 12 (3):60, 2023.

Andy T Liu, Yi-Cheng Lin, Haibin Wu, Stefan Winkler, and Hung-yi Lee. Efficient training of self-supervised speech foundation models on a compute budget. In *2024 IEEE Spoken Language Technology Workshop (SLT)*, pp. 961–968. IEEE, 2024.

Zicheng Liu, Siyuan Li, Di Wu, Zhiyuan Chen, Lirong Wu, Jianzhu Guo, and Stan Z. Li. Automix: Unveiling the power of mixup for stronger classifiers. In *European Conference on Computer Vision*, pp. 441–458, 2022.

Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214):1–50, 2023.

Umberto Michelucci. An introduction to autoencoders. *arXiv preprint arXiv:2201.03898*, 2022.

Bonan Min, Hayley Ross, Elior Sulem, Amir Pouran Ben Veyseh, Thien Huu Nguyen, Oscar Sainz, Eneko Agirre, Ilana Heintz, and Dan Roth. Recent advances in natural language processing via large pre-trained language models: A survey. *ACM Comput. Surv.*, 56(2), sep 2023. ISSN 0360-0300. doi: 10.1145/3605943.

Sören Mindermann, Jan M Brauner, Muhammed T Razzak, Mrinank Sharma, Andreas Kirsch, Winnie Xu, Benedikt Höltgen, Aidan N Gomez, Adrien Morisot, Sebastian Farquhar, et al. Prioritized training on points that are learnable, worth learning, and not yet learnt. In *International Conference on Machine Learning*, pp. 15630–15649. PMLR, 2022.

Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. On the stability of fine-tuning bert: Misconceptions, explanations, and strong baselines. *arXiv preprint arXiv:2006.04884*, 2020.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

Fei Pan, Inkyu Shin, Francois Rameau, Seokju Lee, and In So Kweon. Unsupervised intra-domain adaptation for semantic segmentation through self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3764–3773, 2020.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

Matthew E Peters, Sebastian Ruder, and Noah A Smith. To tune or not to tune? adapting pretrained representations to diverse tasks. *arXiv preprint arXiv:1903.05987*, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

Vikas Verma, Alex Lamb, Christopher Beckham, Amir Najafi, Ioannis Mitliagkas, David Lopez-Paz, and Yoshua Bengio. Manifold mixup: Better representations by interpolating hidden states. In *International conference on machine learning*, pp. 6438–6447. PMLR, 2019.

Peihao Wang, Rameswar Panda, and Zhangyang Wang. Data efficient neural scaling law via model reusing. In *International Conference on Machine Learning*, pp. 36193–36204. PMLR, 2023a.

Xiao Wang, Guangyao Chen, Guangwu Qian, Pengcheng Gao, Xiao-Yong Wei, Yaowei Wang, Yonghong Tian, and Wen Gao. Large-scale multi-modal pre-trained models: A comprehensive survey. *Machine Intelligence Research*, pp. 1–36, 2023b.

Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021.

Kan Wu, Jinnian Zhang, Houwen Peng, Mengchen Liu, Bin Xiao, Jianlong Fu, and Lu Yuan. Tinyvit: Fast pretraining distillation for small vision transformers. In *European Conference on Computer Vision*, pp. 68–85. Springer, 2022.

Yu-Qi Yang, Yu-Xiao Guo, Jian-Yu Xiong, Yang Liu, Hao Pan, Peng-Shuai Wang, Xin Tong, and Baining Guo. Swin3d: A pretrained transformer backbone for 3d indoor scene understanding. *arXiv preprint arXiv:2304.06906*, 2023.

Xingcheng Yao, Yanan Zheng, Xiaocong Yang, and Zhilin Yang. Nlp from scratch without large-scale pretraining: A simple and efficient framework. In *International Conference on Machine Learning*, pp. 25438–25451. PMLR, 2022.

Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 6023–6032, 2019.

Matthew D Zeiler, Dilip Krishnan, Graham W Taylor, and Rob Fergus. Deconvolutional networks. In *2010 IEEE Computer Society Conference on computer vision and pattern recognition*, pp. 2528–2535. IEEE, 2010.

Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 1476–1485, 2019.

Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

Jifan Zhang, Yifang Chen, Gregory Canal, Arnav Mohanty Das, Gantavya Bhatt, Stephen Mussmann, Yinglun Zhu, Jeff Bilmes, Simon Shaolei Du, Kevin Jamieson, et al. Labelbench: A comprehensive framework for benchmarking adaptive label-efficient learning. *Journal of Data-centric Machine Learning Research*, 2024.

Yingtian Zou, Vikas Verma, Sarthak Mittal, Wai Hoh Tang, Hieu Pham, Juho Kannala, Yoshua Bengio, Arno Solin, and Kenji Kawaguchi. Mixupe: Understanding and improving mixup from directional derivative perspective. In *Uncertainty in Artificial Intelligence*, pp. 2597–2607. PMLR, 2023.

A Additional Implementation Details and Experimental Results

A.1 Software and Hardware Dependencies

All of the codes are based on `PyTorch`[2] (Paszke et al., 2019) with `timm` library[3] (Wightman et al., 2021). All experiments in this paper are running on one NVIDIA RTX 6000 Ada GPU.

A.2 Computation Reuse for Different Self-Supervised Learning Approaches

Standard AE. Computation reuse in the standard AE approach is straightforward: the backbone outputs are fed directly to both the reconstruction head and the classification head.

MAE. To enable compute reuse in MixTraining, we use a modified version of MAE. All input patches are first processed by the backbone to produce feature representations. For the classification head, we use all feature representations directly. For the reconstruction head, we randomly mask a subset of the feature representations according to a predefined mask ratio and compute the reconstruction loss only on the masked patches. This modification changes the reconstruction behavior of MAE but leaves the supervised learning stage unchanged, which is the basis for performance evaluation in our experiments.

A.3 Implementation Details of Computer Vision Models

ResNet-18. Our ResNet-18 implementations are derived from the seminal work from He et al. (2016) on deep residual networks. To facilitate self-supervised learning through autoencoders (AE), we augment the ResNet architectures with two additional two-dimensional transposed convolution layers post-backbone for the reconstruction task.

ViT-T. Our ViT-T implementations are largely derived from the seminal work from Wu et al. (2022) and shrink the decoder to 2 layers as suggested in the reproduction challenge by Charisoudis et al. (2023). Specifically, we set the embedding dimension ('emb_dim') to 192, with the encoder and decoder configured to 12 and 2 layers respectively, alongside 3 heads each for both encoder and decoder. The masking ratio is maintained at 0.75 as He et al. (2022) suggests. For the CIFAR-10, CIFAR-100 datasets, the image resolution is standardized to 32×32 pixels with a patch size of 2, while the image resolution is 64×64 pixels with a patch size of 4 for TinyImageNet to ensure uniform computational complexity across all experiments.

Data Augmentation and Preprocessing. For the preprocessing of CIFAR-10 and CIFAR-100, we adopt simple preprocessing as in He et al. (2016), which randomly crops the images to a size of 32×32 pixels, with a padding of 4 pixels on each side of the image, then randomly flips the images horizontally with a 50% probability. For TinyImageNet, we follow preprocessing in the reproduction challenge by Charisoudis et al. (2023), aiming to maintain consistency with established benchmarks and facilitate fair comparison.

Hyperparameters. Detailed training hyperparameters used in our experiments are summarized in Table 12 (ViT-T) and Table 13 (ResNet-18). For experiments on the TinyImageNet dataset using ViT-T, following Charisoudis et al. (2023), we slightly modify the hyperparameters in Table 12: we set the base learning rate to 1×10^{-3} and 2×10^{-3}, the Adam betas to (0.9, 0.95) and (0.9, 0.999), and the weight decay to 0.15 and 0.05 for self-supervised and supervised learning, respectively.

A.4 Additional Experiment Results

We report additional experimental results across various data limitation levels and training epochs. Results with full data are provided in main text (Section 4.3), and results with $p \in \{10\%, 25\%, 50\%, 75\%\}$ data limitation levels are presented in Tables 14 to 17. These results show that MixTraining provides a better compute-performance trade-off compared to the standard SSL+SL pipeline: **MixTraining achieves higher accuracy and lower latency in all 36 out of 36 settings over SSL+SL.**

[2] https://pytorch.org/
[3] https://huggingface.co/timm

Table 12: Hyperparameters for ViT-T across self-supervised and supervised settings.

Hyperparameters	**Self-Supervised learning**	**Supervised Learning**
Batch Size	256	256
Base Learning Rate	1.5×10^{-4}	1×10^{-3}
Learning Rate Scheduler	CosineAnnealing	CosineAnnealing
Optimizer	AdamW	AdamW
Betas	(0.9, 0.95)	(0.9, 0.95)
Weight Decay	0.05	0.05
Warmup Epoch	20	5

Table 13: Hyperparameters for ResNet-18 across self-supervised and supervised settings.

Hyperparameters	**Self-Supervised learning**	**Supervised Learning**
Batch Size	256	256
Base Learning Rate	1×10^{-4}	1×10^{-3}
Learning Rate Scheduler	CosineAnnealing	CosineAnnealing
Optimizer	AdamW	AdamW
Betas	(0.9, 0.95)	(0.9, 0.95)
Weight Decay	0.05	0.05
Warmup Epoch	20	5

Table 14: Parameter study on training epochs under 10% data limitation using the ViT-T model. The highest accuracy in each setting is highlighted in **bold**. MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in 9 out of 9 settings.**

Datasets	**Computation (epoch)**	**SL**		**SSL+SL**		**MixTraining**	
		Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
TinyImageNet	50	7.69%	536.82	8.67%	1136.96	**16.53%**	896.16
	75	7.62%	809.11	9.89%	1762.56	**20.23%**	1376.22
	100	7.24%	1102.42	10.21%	2382.06	**20.99%**	1913.75
CIFAR-10	50	53.28%	287.83	54.75%	637.27	**57.60%**	473.80
	75	53.07%	434.86	55.59%	937.01	**58.59%**	742.53
	100	53.40%	589.44	56.79%	1253.65	**60.45%**	962.16
CIFAR-100	50	19.08%	294.14	20.26%	677.26	**23.36%**	516.54
	75	19.39%	464.97	21.15%	981.43	**24.27%**	770.75
	100	19.05%	609.11	22.27%	1279.78	**25.19%**	1010.94

Table 15: Parameter study on training epochs under 25% data limitation using the ViT-T model. The highest accuracy in each setting is highlighted in **bold**. MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in 9 out of 9 settings.**

Datasets	**Computation (epoch)**	**SL**		**SSL+SL**		**MixTraining**	
		Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
TinyImageNet	50	17.60%	1268.41	17.31%	2771.72	**31.69%**	2129.89
	75	19.49%	1921.23	19.80%	4223.24	**31.90%**	3281.00
	100	20.91%	2619.14	21.78%	5813.27	**31.43%**	4516.69
CIFAR-10	50	65.03%	646.45	67.65%	1409.32	**70.69%**	1076.16
	75	65.69%	985.54	68.57%	2135.05	**71.29%**	1719.24
	100	66.03%	1335.79	67.95%	2774.24	**72.61%**	2206.13
CIFAR-100	50	30.87%	651.25	34.46%	1517.26	**36.87%**	1161.89
	75	31.21%	1033.91	35.05%	2237.72	**38.43%**	1720.56
	100	31.49%	1346.96	34.93%	2882.62	**38.55%**	2226.92

Table 16: Parameter study on training epochs under 50% data limitation using the ViT-T model. The highest accuracy in each setting is highlighted in **bold**. MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in 9 out of 9 settings.**

Datasets	Computation (epoch)	SL		SSL+SL		MixTraining	
		Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
TinyImageNet	50	24.95%	2496.29	27.27%	5401.40	**40.37%**	4280.53
	75	28.81%	3827.39	31.15%	8363.36	**41.99%**	6530.16
	100	30.36%	5181.44	34.30%	11449.43	**42.77%**	8868.88
CIFAR-10	50	73.70%	1252.78	75.26%	2703.05	**78.83%**	2133.61
	75	74.63%	1931.89	76.40%	4155.73	**79.31%**	3255.72
	100	75.00%	2593.27	77.06%	5354.08	**79.95%**	4247.78
CIFAR-100	50	42.32%	1248.02	44.95%	2924.08	**48.53%**	2246.85
	75	42.46%	1981.46	45.23%	4355.99	**48.87%**	3275.14
	100	42.50%	2580.08	46.02%	5551.88	**48.60%**	4298.75

Table 17: Parameter study on training epochs under 75% data limitation using the ViT-T model. The highest accuracy in each setting is highlighted in **bold**. MixTraining achieves a Pareto improvement over the SSL+SL baseline, **achieving higher accuracy and lower latency in 9 out of 9 settings.**

Datasets	Computation (epoch)	SL		SSL+SL		MixTraining	
		Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓	Accuracy↑	Latency(s)↓
TinyImageNet	50	33.16%	3713.97	34.81%	8062.47	**46.44%**	6369.23
	75	36.64%	5708.06	40.31%	12538.24	**49.29%**	9779.95
	100	36.98%	7709.21	42.73%	17296.22	**49.30%**	13304.60
CIFAR-10	50	78.08%	1868.81	79.60%	4033.65	**82.91%**	3163.51
	75	78.74%	2856.25	81.24%	6216.65	**83.68%**	4762.06
	100	79.22%	3844.27	82.11%	7993.72	**83.97%**	6234.03
CIFAR-100	50	48.99%	2009.17	51.85%	4345.47	**54.30%**	3329.29
	75	49.00%	2929.78	53.25%	6312.75	**55.05%**	4857.74
	100	48.97%	3814.54	53.79%	8226.83	**55.84%**	6354.76