EFFICIENTMATCH: FASTER CONVERGENCE IN SEMI-SUPERVISED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Semi-supervised learning (SSL) for image classification is most often evaluated under a now-standard protocol consisting of a fixed budget of 2^{20} iterations and a shared architecture. While this benchmark has facilitated comparisons, it has also made asymptotic accuracy the primary design criterion, often at the cost of substantial computation. A complementary question remains comparatively unexplored: which method reaches a useful performance level at the lowest computational cost? We address this gap through two contributions. First, we introduce a unified evaluation protocol that measures the computational budget required to reach a predefined pre-asymptotic accuracy threshold. This budget is assessed jointly in terms of iterations, FLOPs, and wall-clock time, capturing complementary aspects of efficiency and revealing trade-offs that are obscured when considering a single metric. Second, we propose EfficientMatch, an SSL method specifically designed for this efficiency-oriented regime. Its design is motivated by a simple observation: Mixup can accelerate learning when pseudo-labels are reliable, but may hinder convergence otherwise. To address this issue, EfficientMatch exploits pseudo-label confidence to selectively weight the Mixup loss, reusing the consistency mask without introducing additional thresholds. Experiments on CIFAR-10, CIFAR-100, and SVHN show that EfficientMatch reaches high target performance thresholds faster than MixMatch, FixMatch, FlexMatch, and RegMixMatch when measured in both FLOPs and wall-clock time, while maintaining a low per-iteration computational cost. Code is available at: `https://anonymous.4open.science/r/EfficientMatch-CCE5`

1 INTRODUCTION

Semi-supervised learning (SSL) for image classification has advanced remarkably over the past decade. From early pseudo-labeling approaches (Lee et al., 2013) to methods such as MixMatch (Berthelot et al., 2019b), FixMatch (Sohn et al., 2020), and FlexMatch (Zhang et al., 2021), SSL now achieves performance close to that of fully supervised learning while relying on only a handful of labeled examples. These advances have been evaluated under a standardized protocol (Wang et al., 2022a), based on a fixed budget of 2^{20} iterations, a shared WideResNet architecture (Zagoruyko & Komodakis, 2016), and common hyperparameter settings. While this protocol ensures fair comparison across methods, it also reduces evaluation to a single criterion: the final performance achieved after training. As a result, method design has increasingly focused on maximizing asymptotic accuracy, often overlooking the computational cost required to reach it.

The computational cost of this protocol is more than an implementation detail. A budget of 2^{20} iterations can require several days of training on a single GPU and, when multiplied across seeds, label budgets, and ablation studies, places reproduction and exploration beyond the reach of many research groups. This concern is explicitly acknowledged by USB (Wang et al., 2022a), whose authors motivate the benchmark in part by the growing computational and environmental cost of SSL research (Strubell et al., 2019).

More fundamentally, asymptotic performance is not always the most relevant criterion. Practitioners often need to select a method before training it to convergence, while resource-constrained deployments value a useful level of accuracy achieved quickly over a marginally higher accuracy obtained at much greater cost. The question "which method achieves the highest accuracy?" therefore has a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

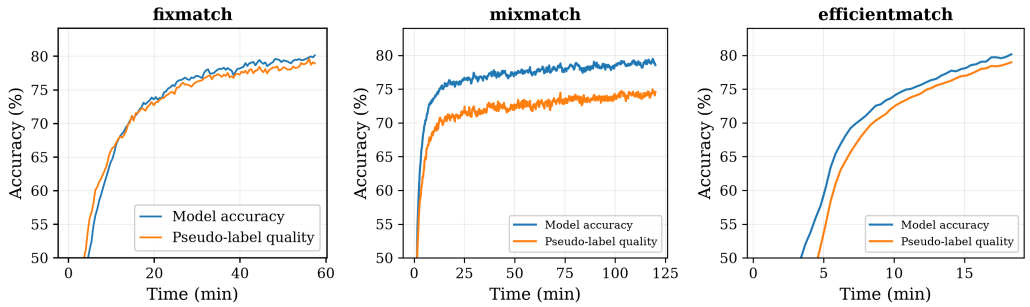


Figure 1: Evolution over training of pseudo-label quality (used in the loss) versus model accuracy, for FixMatch (left), MixMatch (middle) and EfficientMatch (right), on a CIFAR-10, 250-label configuration. Model accuracy closely tracks the accuracy of the retained pseudo-labels for FixMatch, whereas the model surpasses pseudo-label quality for MixMatch. We can see that EfficientMatch inherits both the pseudo-label quality and the improvement from the Mixup.

natural counterpart that remains largely unexplored: *which method reaches a useful accuracy at the lowest computational cost?*

Recent works have begun to challenge this focus on asymptotic performance. FlexMatch (Zhang et al., 2021), for instance, shows that only a fraction of the standard training budget is needed to surpass FixMatch's final accuracy in some settings, while Fast FixMatch (Chen et al., 2024) explicitly reduces training compute through a curriculum batch-size (CBS) schedule. However, these efforts remain inconsistent; each adopts its own definition of efficiency, preventing direct comparison across methods. Moreover, existing approaches are typically designed for the standard long-budget regime and only evaluated afterward under reduced training budgets, rather than being developed specifically for fast convergence.

The contrast between asymptotic performance and convergence efficiency is particularly visible when comparing MixMatch and FixMatch. As shown in our experiments (§4), MixMatch's convergence speed is highly sensitive to the label regime: it fails to reach τ_{acc} within budget at 250 labels on CIFAR-10, yet converges reliably and becomes the fastest method at 4000 labels. FixMatch's filtered consistency loss, by contrast, converges more slowly but reaches τ_{acc} across most of the configurations tested here.

This behavior stems from the different mechanisms through which the two methods control confirmation bias. MixMatch gradually increases the weight of the unsupervised loss, thereby limiting, but never excluding, unreliable pseudo-labels during early training. FixMatch instead relies on confidence thresholding, which prevents low-quality pseudo-labels from contributing to the loss, but does so at the cost of a low admission rate and consequently slower convergence in the early stages of training.

Our hypothesis is that MixMatch's instability in the low-label regime does not originate from Mixup (Zhang et al., 2017) itself, but from the absence of an explicit mechanism to prevent unreliable pseudo-labels from contributing to the mixed signal. Therefore, we propose to apply confidence filtering directly within the Mixup loss, rather than before the mixing operation as in a naive combination of MixMatch and FixMatch. By filtering the mixed signal itself, we aim to retain the regularization benefits of Mixup while reducing the adverse impact of low-confidence pseudo-labels.

EfficientMatch is designed around this idea: to retain Mixup's benefits while avoiding both MixMatch's instability and the slow early convergence induced by strict confidence thresholding.

Our contributions are threefold:

- **A unified protocol for evaluating convergence efficiency**, which measures the computational budget required to reach a predefined pre-asymptotic accuracy threshold shared across all compared methods. We report iterations, FLOPs, and wall-clock time jointly, as these metrics capture complementary aspects of efficiency.

- **EfficientMatch**, an SSL method explicitly designed for this regime. Its core mechanism filters the Mixup loss using the confidence mask of the mixing partner, reusing the threshold already computed for consistency regularization.
- **A comprehensive empirical study** on CIFAR-10, SVHN, and CIFAR-100, showing that EfficientMatch consistently reaches target performance levels faster than existing SSL methods while maintaining a low per-iteration computational cost.

Throughout this work, methods are compared by the computational budget required to reach a fixed, pre-asymptotic accuracy target, not by the accuracy they eventually attain: no result reported here is a claim about asymptotic performance under the full 2^{20}-iteration protocol. That target is chosen rather than given, and it is therefore part of the comparison: a different target can produce a different ranking. We measure how much it does, and delimit the regime in which EfficientMatch's advantage holds, in §6.

2 Related Work

SSL mechanisms. Modern SSL methods typically combine four practices: data augmentation policies such as RandAugment (Cubuk et al., 2019) and CTAugment (Berthelot et al., 2019a); Mixup (Zhang et al., 2017), which promotes smoother decision boundaries through convex interpolation (Oh & Yun, 2023); confidence thresholding to mitigate confirmation bias (Arazo et al., 2020); and consistency regularization between weakly and strongly augmented views (Laine & Aila, 2016; Tarvainen & Valpola, 2017; Miyato et al., 2018). FixMatch (Sohn et al., 2020) combines confidence thresholding with consistency regularization, while FlexMatch (Zhang et al., 2021) extends this idea with adaptive class-wise thresholds. MixMatch (Berthelot et al., 2019b) and ReMixMatch (Berthelot et al., 2019a) instead rely on Mixup without explicit confidence filtering, making them more sensitive to unreliable pseudo-labels in low-label settings. RegMixMatch (Han et al., 2025) identifies the same limitation and addresses it through class-aware weighting, preserving both high- and low-confidence pseudo-labels to maximize final performance under the standard long-budget protocol. In contrast, our proposed method reuses the consistency mask to filter the Mixup signal itself, explicitly targeting convergence efficiency under reduced computational budgets.

The standardized protocol and its cost. The evaluation protocol inherited from FixMatch, based on a fixed training budget and a shared architecture, was formalized and extended by USB (Wang et al., 2022a). Beyond improving reproducibility and comparability, USB highlights the growing computational and environmental cost of training SSL methods from scratch. While this standardization has enabled fair benchmarking, it has also reinforced end-of-training performance as the dominant evaluation criterion, leaving convergence efficiency largely unexplored.

Convergence efficiency. FlexMatch (Zhang et al., 2021) and Fast FixMatch (Chen et al., 2024) both challenge the exclusive focus on asymptotic performance by showing that substantial gains can be achieved under reduced training budgets. These works nevertheless remain tied to the FixMatch framework and adopt their own notions of efficiency, making direct comparison difficult. By contrast, our work introduces both a unified evaluation protocol for convergence efficiency and a dedicated regularization mechanism designed specifically for this regime.

Positioning. Table 1 situates EfficientMatch along two axes: confidence control and use of Mixup. FixMatch and FlexMatch filter pseudo-labels but do not mix samples; MixMatch and RegMixMatch mix samples but target asymptotic performance, differing in how they handle low-confidence ones: propagated as-is by MixMatch, reweighted by RegMixMatch. EfficientMatch combines both axes toward a different objective: reusing the consistency mask to filter the Mixup partner, it targets convergence efficiency rather than asymptotic accuracy.

3 EfficientMatch

Let $\mathcal{X} = \{(x_b, y_b)\}_{b=1}^{B}$ denote a labeled batch and $\mathcal{U} = \{u_b\}_{b=1}^{\mu B}$ an unlabeled batch. We denote by $\alpha(\cdot)$ and $\mathcal{A}(\cdot)$ the weak and strong augmentation operators, respectively. For an unlabeled sample u_b, $q_b = p_\theta(y \mid \alpha(u_b))$ denotes the predicted class distribution, $\hat{q}_b$ its hard pseudo-label, and $m_b(\tau) =$

Table 1: Positioning of the main SSL methods with respect to their treatment of pseudo-labels and their evaluation objective.

Method	Confidence control	Use of Mixup	Main objective
MixMatch (Berthelot et al., 2019b)	None	Mixes all pseudo-labels	Max. performance
FixMatch (Sohn et al., 2020)	Fixed threshold	None	Max. performance
FlexMatch (Zhang et al., 2021)	Per-class adaptive	None	Convergence & max. performance
Fast FixMatch (Chen et al., 2024)	Fixed (inherited from FixMatch)	None	Compute reduction
RegMixMatch (Han et al., 2025)	Dual (adaptive + fixed)	Differentiated by confidence	Max. performance
EfficientMatch	Reused fixed threshold	Filters partner post-mix	Convergence, reduced budget

$\mathbb{1}(\max_k q_{b,k} \geq \tau)$ the confidence mask. The threshold τ is fixed in FixMatch and EfficientMatch, whereas FlexMatch adapts it on a per-class basis (Zhang et al., 2021). Finally, $H(\cdot, \cdot)$ denotes the cross-entropy loss.

EfficientMatch combines two complementary components that have independently proven effective in semi-supervised learning: confidence-based consistency regularization, inherited from FixMatch, and Mixup-based regularization, inherited from MixMatch. Its key idea is to reuse the confidence mask already computed for consistency regularization to control which pseudo-labeled samples contribute to the Mixup loss. This design aims to preserve the regularization benefits of Mixup while limiting the influence of unreliable pseudo-labels.

At each iteration, EfficientMatch processes a labeled batch $\mathcal{X}$ together with two views of the unlabeled batch: a weakly-augmented view $\mathcal{U}_w$ and a strongly-augmented view $\mathcal{U}_s$. The resulting objective combines three components: a supervised loss, a consistency loss, and a confidence-filtered Mixup loss. Full pseudocode is given in Algorithm 1.

Supervised loss. As in most SSL methods, we keep the standard cross-entropy supervised loss on the labeled samples:

$$\mathcal{L}_s = \frac{1}{B} \sum_b H(y_b, p_\theta(y \mid \alpha(x_b))) \tag{1}$$

Consistency loss. EfficientMatch reuses the consistency-regularization loss introduced by FixMatch, which remains the foundation of many strong SSL methods (Nguyen, 2024; Han et al., 2025; Wang et al., 2022b). For each unlabeled sample, a pseudo-label $\hat{q}_b$ is computed from the weakly augmented view and retained only if its confidence exceeds a fixed threshold $\tau = 0.95$:

$$\mathcal{L}_u = \frac{1}{\mu B} \sum_b m_b(\tau)\, H(\hat{q}_b, p_\theta(y \mid \mathcal{A}(u_b))) \tag{2}$$

Pseudo-label generation (i.e., q_b, $\hat{q}_b$, and $m_b(\tau)$) is performed through a forward pass on the weakly augmented unlabeled batch $\mathcal{U}_w$ without gradient tracking.

Confidence-filtered Mixup. Our contribution augments the consistency loss with a Mixup regularization channel. We first construct the set $\mathcal{X} \cup \mathcal{U}_w$, where labeled samples are associated with their ground-truth labels and a confidence mask equal to 1, while unlabeled samples are associated with their hard pseudo-labels $\hat{q}_j$ and confidence masks $m_j(\tau)$. The resulting tuples (x_i, p_i, m_i) are randomly shuffled to obtain mixing partners (x'_i, p'_i, m'_i). Each sample is then mixed with its partner using :

$$\lambda_i \sim \mathrm{Beta}(\alpha_m, \alpha_m), \qquad \lambda_i \leftarrow \max(\lambda_i, 1 - \lambda_i), \tag{3}$$

which enforces $\lambda_i \geq 0.5$ and therefore preserves the dominance of the anchor sample. The mixed input and target are defined as :

Algorithm 1 EfficientMatch – one training iteration

Require: Labeled batch $\mathcal{X} = \{(x_b, y_b)\}_{b=1}^{B}$, unlabeled batch $\mathcal{U} = \{u_b\}_{b=1}^{\mu B}$, threshold τ, Beta parameter α_m, weights $\lambda_u, \lambda_{\text{mix}}$

Pseudo-labeling (no gradient)

$q_b \leftarrow p_\theta(y \mid \alpha(u_b)) = \text{logits}_u^w$, $\hat{q}_b \leftarrow \arg\max(q_b)$, $m_b \leftarrow \mathbb{1}(\max(q_b) \geq \tau)$

Fused forward and FixMatch loss

$[\text{logits}_x;\ \text{logits}_u^s] \leftarrow f_\theta([\alpha(x_{1:B});\ \mathcal{A}(u_{1:\mu B})])$

$\mathcal{L}_s \leftarrow \frac{1}{B} \sum_b H(y_b, \text{logits}_x^{(b)})$ ▷ Eq. (1)

$\mathcal{L}_u \leftarrow \frac{1}{\mu B} \sum_b m_b\, H(\hat{q}_b, \text{logits}_u^{s,(b)})$ ▷ Eq. (2)

Filtered Mixup

$\mathcal{W} \leftarrow \text{Shuffle}\big((x_i, y_i, 1)_{i=1}^{B} \cup (u_j^w, \hat{q}_j, m_j)_{j=1}^{\mu B}\big)$ ▷ mixing partners (x_i', p_i', m_i')

$\lambda_i \sim \text{Beta}(\alpha_m, \alpha_m), \quad \lambda_i \leftarrow \max(\lambda_i, 1 - \lambda_i)$

$\tilde{x}_i \leftarrow \lambda_i x_i + (1 - \lambda_i) x_i', \quad \tilde{p}_i \leftarrow \lambda_i p_i + (1 - \lambda_i) p_i'$ ▷ Eq. (4)

$\text{logits}_{\text{mix}} \leftarrow f_\theta(\tilde{x}_{1:B+\mu B})$

$\mathcal{L}_{\text{mix}} \leftarrow \frac{1}{B+\mu B} \sum_i m_i'\, H(\tilde{p}_i, \text{logits}_{\text{mix}}^{(i)})$ ▷ Eq. (5)

Update

$\mathcal{L} \leftarrow \mathcal{L}_s + \lambda_u \mathcal{L}_u + \lambda_{\text{mix}} \mathcal{L}_{\text{mix}}$

Backpropagate $\mathcal{L}$, update θ

$$\tilde{x}_i = \lambda_i x_i + (1 - \lambda_i)\, x_i', \qquad \tilde{p}_i = \lambda_i\, p_i + (1 - \lambda_i)\, p_i', \tag{4}$$

Unlike MixMatch, confidence filtering is applied directly within the Mixup channel. Specifically, the contribution of a mixed sample is controlled by the confidence mask of its mixing partner, m_i'. The resulting loss is

$$\mathcal{L}_{\text{mix}} = \frac{1}{B + \mu B} \sum_{i=1}^{B+\mu B} m_i'\, H\big(\tilde{p}_i,\, p_\theta(y \mid \tilde{x}_i)\big). \tag{5}$$

This mechanism reuses the confidence mask already computed for consistency regularization and therefore introduces no additional thresholding procedure. Variants of this design are evaluated in the ablation study.

Total loss and optimization.

$$\mathcal{L} = \mathcal{L}_s + \lambda_u\, \mathcal{L}_u + \lambda_{\text{mix}}\, \mathcal{L}_{\text{mix}} \tag{6}$$

Differentiation. Unlike MixMatch/ReMixMatch (Berthelot et al., 2019b;a), the signal injected into $\mathcal{L}_{\text{mix}}$ is never built from a mixing partner the model deems unreliable. Unlike FixMatch/FlexMatch (Sohn et al., 2020; Zhang et al., 2021), this channel adds extra regularization at the cost of an additional network forward pass on the mixed data. The net overhead of EfficientMatch relative to FixMatch is quantified empirically via direct FLOP measurement in §4, rather than assumed negligible a priori: both methods share the same pseudo-labeling forward pass (size μB, no gradient), but EfficientMatch replaces FixMatch's two remaining separate passes (labeled, size B; strong-view consistency, size μB) with one fused pass of size $B + \mu B$, plus one additional Mixup pass of the same size; a net addition of one $(B + \mu B)$-sized forward pass per iteration. RegMixMatch (Han et al., 2025) shares our diagnosis regarding unfiltered Mixup but differs on several mechanistic points: a dual threshold (τ_c for consistency, a markedly stricter τ_m for admission into the mix) versus our single, reused threshold; a cut-and-paste ResizeMix mixing operator (Qin et al., 2020) rather than the pixel-space convex interpolation of the original Mixup that we retain; and a strong mixing intensity ($\alpha_h = 1.0$) versus an intensity dominated by the original example ($\lambda \geq 0.5$ enforced) in our case.

Choice of μ per method. Unlike Tab. 14 in an earlier version of this work, μ is not shared across methods: each keeps the value established in the literature or the one we determined for EfficientMatch (FixMatch/FlexMatch: $\mu = 7$ (Sohn et al., 2020; Zhang et al., 2021); MixMatch: $\mu = 1$ (Berthelot et al., 2019b)). For EfficientMatch, $\mu = 3$ is retained. On both 250 and 4000 labels, and across all 3 seeds, $\mu = 3$ minimizes total FLOPs among configurations that converge reliably (Appendices 5 and 6). The relationship is not monotonic: $\mu = 1$ fails on 2 of 3 seeds at 250 labels (the consistency signal is too sparse and unstable across seeds) but converges reliably at 4000 labels, where the richer supervised signal compensates. Beyond $\mu = 3$, the FLOP cost grows without measurable and consistent benefit, it is more a resource trade-off rather than a quality one.

4 EXPERIMENTS

Protocol. Three datasets: CIFAR-10, SVHN, and CIFAR-100. WideResNet-28-2 (Zagoruyko & Komodakis, 2016) for CIFAR-10 and SVHN, in the low-label regime with 250 labels, over 3 random seeds. On CIFAR-100, we use a WideResNet-28-4. We compare EfficientMatch against the methods discussed above: MixMatch, using the original implementation of Berthelot et al. (2019b), FixMatch, FlexMatch, and RegMixMatch.

Choice of label budgets and target thresholds. Label budgets and thresholds τ_{acc} result from a trade-off between experiment time (single GPU), the visibility of a fast-convergence phase rather than an asymptotic plateau, and feasibility across all compared methods. On SVHN and CIFAR-10, we exclude the 40-label regime (strong inter-seed instability in pilot runs); on SVHN, we also exclude 1000 labels (convergence too fast, not discriminative). We therefore retain SVHN/250 ($\tau_{\text{acc}} = 90\%$), CIFAR-10/250 ($\tau_{\text{acc}} = 80\%$), CIFAR-10/4000 ($\tau_{\text{acc}} = 90\%$, an easier task), and CIFAR-100/2500 ($\tau_{\text{acc}} = 50\%$, the hardest task). These thresholds are deliberately pre-asymptotic: since the accuracy gain per iteration is markedly larger early in training than after the plateau, measuring the time to reach a high but pre-asymptotic threshold captures convergence speed without paying the cost of that plateau; particularly long on SVHN given the relative simplicity of the data. A direct consequence is that this choice does not inform us about EfficientMatch's asymptotic performance, discussed in §6.

Stopping criterion. Each run is stopped as soon as the EMA accuracy reaches a fixed target threshold τ_{acc}, or once the run exceeds 2 hours (sufficient to perform a large number of iterations for every method), to avoid plateaus lasting hundreds of thousands of iterations. We evaluate methods every $2^9 = 512$ iterations across all configurations.

Complementarity of the three metrics. We consistently report three metrics: iterations, wall-clock time, FLOPs. Each one isolates a distinct source of efficiency, with none dominating the other two. The number of iterations measures algorithmic efficiency per optimization step, independent of the compute cost of each step: a method can converge in very few iterations while remaining expensive if each iteration is itself heavy (cf. RegMixMatch, Tab. 2, which converges in few steps but at a per-iteration FLOP cost markedly higher than EfficientMatch). Wall-clock time, conversely, reveals a cost that neither iterations nor FLOPs capture: that of the mechanisms executed inside the training loop (adaptive thresholding, curricula, sorting, control operations), whose real-world impact depends on how efficiently they can be implemented rather than on their raw compute load. This is, for instance, why the FLOP gain of Fast FixMatch (Chen et al., 2024) (batch-size curriculum) does not necessarily translate into a wall-clock gain in our pipeline (§4.1). FLOPs, finally, measure a compute cost independent of hardware and software stack (compilation, precision, parallelization), enabling a comparison that does not artificially favor a method better optimized for a given hardware configuration. A method that is efficient in the sense of this work must therefore be so on all three axes simultaneously, not on a single one taken in isolation.

4.1 LIMITATIONS

All experiments rely on a single NVIDIA RTX 5060 Ti GPU, which constrains us to the four label-budget/dataset combinations reported here, each averaged over 3 seeds. The optimizations applied to reduce experiment duration are listed in Appendix B.

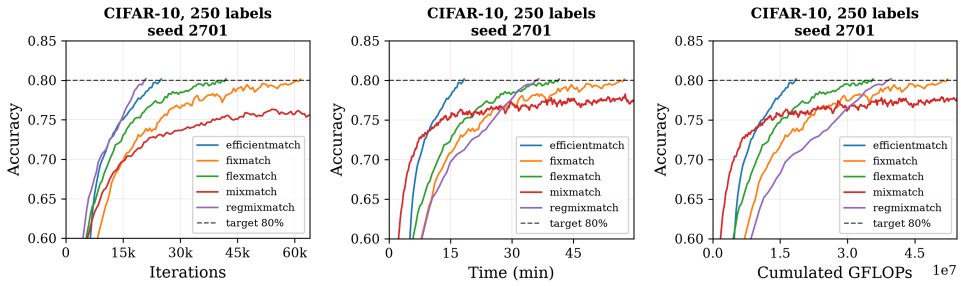


Figure 2: Convergence of EfficientMatch compared to MixMatch, FixMatch, and FlexMatch on CIFAR-10/250, seed 2701, along the three criteria: iterations (left), time (middle), and FLOPs (right). The three criteria rank methods differently. RegMixMatch appears highly efficient per iteration, whereas EfficientMatch is more efficient on the other two criteria, except early in training, where MixMatch dominates before reaching its plateau.

Table 2: Steps (x1000), wall-clock time (min), and FLOPs (PFLOPs) to reach τ_{acc}, 250-label regime, per seed (2312, 0308, 2701). **Bold** = best value in the column; underline = second-best († = never reached, excluded from ranking).

	SVHN, 250l, $\tau = 90\%$									**CIFAR-10, 250l, $\tau = 80\%$**								
	Steps			Time			FLOPs			Steps			Time			FLOPs		
Method	2312	0308	2701	2312	0308	2701	2312	0308	2701	2312	0308	2701	2312	0308	2701	2312	0308	2701
MixMatch	18.5	52.5	35.0	6.5	17.4	11.8	5.6	15.8	10.6	†	†	†	†	†	†	†	†	†
FixMatch	8.0	6.5	8.0	7.7	6.3	7.7	6.8	5.5	6.8	127.0	64.0	61.5	113.7	58.2	56.3	108.0	54.4	52.3
FlexMatch	†	†	†	†	†	†	†	†	†	65.5	52.5	42.0	63.3	51.3	40.7	55.7	44.6	35.7
RegMixMatch	**5.5**	**4.5**	**6.0**	10.2	8.2	11.2	10.4	8.5	11.4	**35.0**	**25.5**	**21.0**	60.6	43.8	36.1	66.2	48.2	39.7
EfficientMatch	7.0	6.5	8.0	**5.7**	**5.2**	**6.3**	**5.2**	**4.8**	**5.9**	42.0	28.5	25.0	**29.6**	**20.3**	**17.9**	**31.1**	**21.1**	**18.5**

Fast FixMatch Chen et al. (2024), discussed in §2 as the closest work in objective, is not included in the main empirical comparison (Tab. 2, 3). Its curriculum batch size (CBS) schedule explicitly varies batch size at every iteration according to a B-EXP schedule (Appendix C), producing non-fixed tensor shapes from one iteration to the next. In a targeted diagnostic (CIFAR-10, 250 labels, 3000 iterations), Fast FixMatch does reduce cumulative FLOPs by 62.6% relative to FixMatch at equal iteration count, consistent with the gain reported by the authors; but runs 2.13× slower in wall-clock time, which neutralizes and reverses this FLOP gain on our software stack (Appendix A.4). The cause is identified directly in the execution logs: CUDA Graphs compilation (`reduce-overhead` mode) records a new graph for every distinct batch shape encountered rather than reusing an already-compiled graph, a mechanism fundamentally incompatible with variable tensor shapes. This is a limitation of our compilation pipeline, not a challenge to the authors' original result on the FLOP metric, which we ourselves retain as one of our three main criteria (§4).

5 Results

EfficientMatch's speed. Across all twelve seed/configuration combinations studied in this work, EfficientMatch achieves the lowest wall-clock time and the lowest FLOP cost of the five reference methods, without a single exception (Tables 2, 3). RegMixMatch is the only method that ever converges in fewer iterations, but only by paying a substantially higher per-iteration cost (§4, "Evaluation bias"), which is exactly what the wall-clock and FLOP metrics are designed to expose. This consistency holds across the full range of conditions tested here from 250 to 4000 labels, from 10 to 100 classes, on two architectures (WRN-28-2, WRN-28-4) and even including SVHN/250, the one configuration where FlexMatch and RegMixMatch's adaptive thresholding otherwise dominates in isolation (§6).

The MixMatch reversal as a validation of our diagnosis. On CIFAR-10, MixMatch's relative position reverses between the two label budgets tested (Tab. 2, 3): at 250 labels, MixMatch never reaches $\tau_{\text{acc}} = 80\%$ within the allowed budget on any of the 3 seeds (Tab. 2); at 4000 labels, it converges reliably across all 3 seeds and becomes $1.05\times$ to $1.9\times$ faster than FixMatch and FlexMatch,

Table 3: Steps (x1000), wall-clock time (min), and FLOPs (PFLOPs) to reach τ_{acc}, per seed (2312, 0308, 2701). **Bold** = best value in the column; underline = second-best († = never reached, excluded from ranking).

	CIFAR-10, 4000l, $\tau = 90\%$									**CIFAR-100, 2500l,** $\tau = 50\%$								
	Steps			Time			FLOPs			Steps			Time			FLOPs		
Method	2312	0308	2701	2312	0308	2701	2312	0308	2701	2312	0308	2701	2312	0308	2701	2312	0308	2701
MixMatch	198.5	213.5	165.5	55.5	59.6	46.3	59.9	64.4	49.9	†	†	†	†	†	†	†	†	†
FixMatch	99.0	91.5	100.5	85.2	79.1	86.9	84.2	77.8	85.4	†	†	†	†	†	†	†	†	†
FlexMatch	60.5	90.0	83.5	58.1	86.4	76.8	51.4	76.5	71.0	31.5	38.0	31.5	69.2	82.8	68.0	105.7	127.5	105.7
RegMixMatch	**25.0**	**27.0**	**27.0**	42.9	46.2	47.2	47.3	51.1	51.1	**17.5**	**15.0**	**20.0**	61.0	52.5	70.1	130.7	112.0	149.3
EfficientMatch	41.0	41.0	43.5	**28.2**	**29.3**	**29.8**	**30.4**	**30.4**	**32.2**	30.5	26.5	32.0	**55.1**	**48.1**	**58.0**	**89.0**	**77.4**	**93.5**

depending on the seed. SVHN at 250 labels shows a similar but less uniform pattern: MixMatch is markedly slower and more FLOP-costly than FixMatch on two of the three seeds (up to $2.9\times$), but on the third seed it is instead slightly faster and cheaper ($0.8\times$), consistent with the higher inter-seed variance already documented for MixMatch throughout this work (Appendix A.7). This reversal directly supports our diagnosis (§1): Mixup extracts more useful signal the higher the quality of the labels it has access to, but becomes a liability, to the point of preventing convergence within the allowed budget, when the pseudo-labels available early in training are scarce and unreliable. EfficientMatch is designed precisely to retain this growing benefit of Mixup while making it robust to the low-label regime in which MixMatch fails.

Evaluation bias of the standard protocol. This is where the bias raised in the introduction reappears. RegMixMatch is consistently the most efficient method per iteration across our four configurations (Tab. 2, 3); yet when evaluated in terms of compute time and FLOPs, the ranking changes entirely depending on the configuration, as shown in Figure 2.

Slow convergence failures. Our configurations reveal several methodological limitations, of different nature depending on the method. The simplest configuration, CIFAR-10/4000, allows every method to converge within the allowed time, unlike the other three. On SVHN/250, FlexMatch's adaptive thresholding (distinct from RegMixMatch's) plateaus at 82–87%, never reaching 90% regardless of run duration. This is a convergence failure rather than a speed one, consistent with the results of Zhang et al. (2021) on this dataset. On CIFAR-10/250, by contrast, MixMatch fails by slowness rather than by a ceiling: the 80% threshold remains below its known asymptotic performance under the full protocol (§2), but convergence there is too slow to cross this threshold within the 2-hour budget on every seed plateauing at 76.5%, 77.7%, and 79.5%. The CIFAR-100/2500 configuration reproduces this same speed failure for MixMatch, and also causes FixMatch to fail for the same reason, both plateauing at 45–48%.

RegMixMatch and FixMatch at matched μ. To isolate the contribution of the filtered Mixup channel from that of the choice of μ (§3), we test RegMixMatch (Han et al., 2025) and FixMatch (Sohn et al., 2020) at $\mu = 3$, matching EfficientMatch, on CIFAR-10 250 labels, confirmed across all 3 seeds. EfficientMatch remains $5.1\times$ to $7.9\times$ less FLOP-costly than FixMatch at matched μ. FixMatch compensates for the reduced μ with a substantially higher number of iterations. RegMixMatch at $\mu = 3$ becomes unstable: it never reaches 80% on 2 of the 3 seeds. This result confirms, under controlled μ, that EfficientMatch's advantage does not stem solely from a reduced unlabeled budget per iteration. Details are given in Appendix A.6.

6 Discussion

EfficientMatch's advantage is specific to the targeted performance level. Lowering the target threshold by 5 to 10 points shifts which method is fastest, but not uniformly. On SVHN/250, EfficientMatch remains fastest in time at a 5-point reduction (2/3 seeds), but MixMatch takes over at 10 points, in both time and FLOPs (2/3 seeds each); even at 5 points, though, FLOPs shows no clear majority, with three seeds yielding three different winners. On CIFAR-10/250, EfficientMatch stays fastest in time even at 10 points (2/3 seeds); only the FLOPs ranking flips to MixMatch, and only at the largest reduction. CIFAR-10/4000, where MixMatch is already competitive at the original threshold (§6), shows little further change. CIFAR-100/2500 is the clearest case: MixMatch

fails outright at the original threshold (Tab. 3), yet becomes the unanimous winner in both time and FLOPs at a 10-point reduction, moving from total failure to total dominance as the target threshold alone changes (at 5 points, no clear majority emerges, splitting between MixMatch, EfficientMatch, and FlexMatch by seed). This is consistent with our diagnosis (§1): the pseudo-label admission bottleneck EfficientMatch targets matters most when the label budget is small *and* the target is high; relaxing either condition erodes this advantage. EfficientMatch's advantage should thus be read as confined to this regime, not as superiority at every quality level or label budget.

Adaptive thresholding: a speed gain at the cost of robustness. A natural question is whether EfficientMatch's fixed, reused threshold (§3) is itself a source of unnecessary slowness. We test a variant substituting FreeMatch's adaptive thresholding (Wang et al., 2022b) for it. Averaged over 3 seeds, this reduces the iterations needed to reach τ_{acc} by 16% on SVHN/250, 16% on CIFAR-10/250 (1–25% depending on seed), 32% on CIFAR-10/4000, and 48% on CIFAR-100/2500 (Table 4), with the gain growing as labels and classes increase. This comes at the cost of a dataset-specific patch, however: on SVHN, the threshold must be clamped to $[0.9, 0.95]$, following a correction FreeMatch's authors document for this dataset (Appendix A.5). Without it, the variant never reaches 90% on any seed, plateauing at 84.2–86.3%, a failure we confirm is not specific to EfficientMatch: RegMixMatch, sharing the same thresholding mechanism, exhibits the same failure mode on SVHN without the clamp (Appendix A.5). Adaptive thresholding can thus accelerate convergence substantially, but at the cost of a per-dataset adjustment that compromises robustness, precisely what EfficientMatch's reused fixed threshold avoids, unpatched, across all four configurations tested here.

A preliminary look beyond the tested regime. A single-seed, extended-horizon diagnostic (Appendix A.8) suggests EfficientMatch's absolute time and FLOP savings over RegMixMatch keep growing well past our pre-asymptotic thresholds, up to at least 88% accuracy, consistent with the metric complementarity of §4: RegMixMatch's larger $\mu = 7$ extracts more signal per iteration at a proportionally higher cost. Beyond 88%, evaluation noise and RegMixMatch's incomplete run prevent further conclusions; this single-seed result should be read as indicative only.

Table 4: Iteration-count reduction of the FreeMatch-thresholding variant relative to EfficientMatch, per seed (time gives near-identical gaps).

Configuration	2312	0308	2701	Mean
SVHN, 250l (clamp active)	-14%	-8%	-25%	-16%
CIFAR-10, 250l	-1%	-25%	-22%	-16%
CIFAR-10, 4000l	-38%	-31%	-29%	-32%
CIFAR-100, 2500l	-47%	-45%	-50%	-48%

7 Conclusion

Convergence efficiency should be considered a design objective in its own right, not just a complementary evaluation criterion. By jointly measuring the number of iterations, wall-clock time, and FLOPs required to reach a useful accuracy, our protocol shows that the ranking established from asymptotic performance does not directly translate to the reduced-compute-budget regime. This is most visibly in MixMatch's reversal between 250 and 4000 labels on CIFAR-10, as well as in the sensitivity of this ranking to the choice of target accuracy threshold (§6).

Within this framework, we proposed EfficientMatch, which combines a consistency loss with a Mixup term filtered by the mixing partner's confidence. Our results support the hypothesis that the position of filtering relative to the mixing operator shapes the trade-off between pseudo-label stability and the speed at which the unlabeled signal is exploited. This filtering can be turned into an adaptive threshold for a further speed gain, but only at the cost of a per-dataset adjustment that compromises robustness, a trade-off EfficientMatch's fixed, reused threshold avoids by design (§6).

Two directions remain open. The first concerns the articulation between the fast-convergence phase targeted by EfficientMatch and the asymptotic regime studied by standard SSL protocols. The second is whether the observed mechanism generalizes to other mixing operators, architectures, and datasets.

REFERENCES

Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International joint conference on neural networks (IJCNN)*, pp. 1–8. IEEE, 2020.

David Berthelot, Nicholas Carlini, Ekin D Cubuk, Alex Kurakin, Kihyuk Sohn, Han Zhang, and Colin Raffel. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. *arXiv preprint arXiv:1911.09785*, 2019a.

David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. Mixmatch: A holistic approach to semi-supervised learning. *Advances in neural information processing systems*, 32, 2019b.

Yu Chen et al. Fast FixMatch: Faster semi-supervised learning with curriculum batch size. In *European Conference on Computer Vision (ECCV)*, 2024.

Ekin D. Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V. Le. Randaugment: Practical automated data augmentation with a reduced search space, 2019. URL https://arxiv.org/abs/1909.13719.

Haorong Han, Jidong Yuan, Chixuan Wei, and Zhongyang Yu. Regmixmatch: Optimizing mixup utilization in semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 17032–17040, 2025.

Samuli Laine and Timo Aila. Temporal ensembling for semi-supervised learning. *arXiv preprint arXiv:1610.02242*, 2016.

Dong-Hyun Lee et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In *Workshop on challenges in representation learning, ICML*, volume 3, pp. 896. Atlanta, 2013.

Takeru Miyato, Shin-ichi Maeda, Masanori Koyama, and Shin Ishii. Virtual adversarial training: a regularization method for supervised and semi-supervised learning. *IEEE transactions on pattern analysis and machine intelligence*, 41(8):1979–1993, 2018.

Khanh-Binh Nguyen. Sequencematch: Revisiting the design of weak-strong augmentations for semi-supervised learning. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pp. 96–106, 2024.

Junsoo Oh and Chulhee Yun. Provable benefit of mixup for finding optimal decision boundaries, 2023. URL https://arxiv.org/abs/2306.00267.

Jie Qin, Jiemin Fang, Qian Zhang, Wenyu Liu, Xingang Wang, and Xinggang Wang. Resizemix: Mixing data with preserved object information and true labels, 2020. URL https://arxiv.org/abs/2012.11101.

Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. *Advances in neural information processing systems*, 33:596–608, 2020.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for deep learning in nlp, 2019. URL https://arxiv.org/abs/1906.02243.

Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

Yidong Wang, Hao Chen, Yue Fan, Wang Sun, Ran Tao, Wenxin Hou, Renjie Wang, Linyi Yang, Zhi Zhou, Lan-Zhe Guo, Heli Qi, Zhen Wu, Yu-Feng Li, Satoshi Nakamura, Wei Ye, Marios Savvides, Bhiksha Raj, Takahiro Shinozaki, Bernt Schiele, Jindong Wang, Xing Xie, and Yue Zhang. USB: A unified semi-supervised learning benchmark for classification. In *Advances in Neural Information Processing Systems*, volume 35, pp. 3938–3961, 2022a.

Table 5: Steps (x1000), wall-clock time (min), and FLOPs (PFLOPs) for μ ablation (EfficientMatch, CIFAR-10, 250 labels, $\tau = 80\%$), per seed (2312, 0308, 2701). **Bold** = best value in the column; underline = second-best († = never reached, excluded from ranking, max accuracy in parentheses).

	Steps			Time			FLOPs		
μ	2312	0308	2701	2312	0308	2701	2312	0308	2701
1	† (77.6%)	† (79.8%)	215.5	†	†	87.4	†	†	76.8
5	28.5	**17.0**	18.5	30.0	**18.1**	19.6	32.0	**19.1**	20.8
7	**25.5**	18.5	**15.5**	35.8	26.9	22.6	38.5	27.9	23.4
3 (default)	42.0	28.5	25.0	**29.6**	20.3	**17.9**	**31.1**	21.1	**18.5**

Table 6: Steps (x1000), wall-clock time (min), and FLOPs (PFLOPs) for μ ablation (EfficientMatch, CIFAR-10, 4000 labels, $\tau = 90\%$), per seed (2312, 0308, 2701). **Bold** = best value in the column; underline = second-best.

	Steps			Time			FLOPs		
μ	2312	0308	2701	2312	0308	2701	2312	0308	2701
1	99.5	101.5	100.5	39.6	42.3	39.6	35.5	36.2	35.8
5	32.0	34.5	30.5	33.6	37.7	32.0	36.0	38.8	34.3
7	**26.0**	**25.5**	**26.0**	36.4	35.7	36.5	39.2	38.5	39.2
3 (default)	41.0	41.0	43.5	**28.2**	**29.3**	**29.8**	**30.4**	**30.4**	**32.2**

Yidong Wang, Hao Chen, Qiang Heng, Wenxin Hou, Yue Fan, Zhen Wu, Jindong Wang, Marios Savvides, Takahiro Shinozaki, Bhiksha Raj, et al. Freematch: Self-adaptive thresholding for semi-supervised learning. *arXiv preprint arXiv:2205.07246*, 2022b.

Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. *arXiv preprint arXiv:1605.07146*, 2016.

Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. FlexMatch: Boosting semi-supervised learning with curriculum pseudo labeling. In *Advances in Neural Information Processing Systems*, volume 34, pp. 18408–18419, 2021.

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

A Additional Results

This section presents in detail results mentioned in the main text, along with additional experiments.

A.1 μ Ablation.

This study compares which value of μ, controlling the number of unlabeled samples introduced at each iteration, is optimal for our method under the three evaluation criteria retained. Results are shown in Tables 5 and 6.

A.2 λ_{MIX} Ablation.

This ablation compares EfficientMatch's results as the weight λ_{mix} assigned to the mixing loss in the total loss is varied. $\lambda \approx 0$ pushes EfficientMatch toward a FixMatch with $\mu = 3$, while $\lambda \gg 1$ overweights the mixing term and brings EfficientMatch closer to standard MixMatch behavior with a higher gap between pseudo-label quality and model accuracy and a faster convergence at the beginning of the training. (cf. Figure 3) . Results are shown in Table 7.

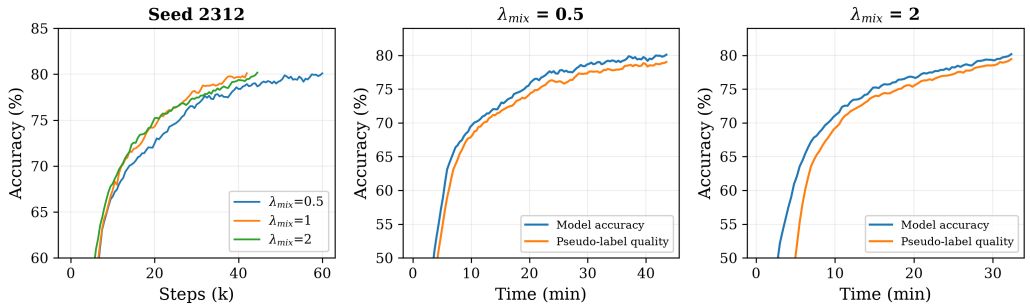


Figure 3: Evolution over training of accuracy (right) for the three Mixup weights (0.5, 1, 2), pseudo-label quality (used in the loss) versus model accuracy for $\lambda_{\text{mix}} = 0.5$ and $\lambda_{\text{mix}} = 2$ on CIFAR-10 250 labels seed 2312 configuration.

Table 7: Wall-clock time (min) and FLOPs (PFLOPs) for λ_{mix} ablation (EfficientMatch, CIFAR-10, 250 labels, $\tau = 80\%$), per seed (2312, 0308, 2701). **Bold** = best value in the column; underline = second-best.

	Time			FLOPs		
λ_{mix}	2312	0308	2701	2312	0308	2701
0.5	42.4	25.6	21.8	44.4	26.3	22.6
2	31.5	26.0	19.7	32.9	27.0	20.4
1 (default)	**29.6**	**20.3**	**17.9**	**31.1**	**21.1**	**18.5**

A.3 Mixing Target Ablation.

We evaluate which configuration of the mixing target yields the best results among three alternatives for the label used in the mixing loss:

1. Hard targets: the label is identical to that of the original sample, treating the mixed sample as a plain augmentation rather than a true Mixup.
2. Semi-soft targets: the label is the linear combination of the anchor sample's label and the mixed sample's label. This is the version retained in our work.
3. Soft targets: the label is the linear combination of the anchor sample's label and the softmax distribution of the mixed sample, yielding a noisier label.

Results are shown in Table 8.

Table 8: Wall-clock time (min) and FLOPs (PFLOPs) for different unlabeled target representations in the Mixup channel on CIFAR-10, 250 labels, $\tau = 80\%$, per seed. **Bold** = best value in the column; underline = second-best (tie underlined twice if the gap is zero).

	Time			FLOPs		
Variant	2312	0308	2701	2312	0308	2701
Hard	32.6	39.1	23.7	33.3	41.1	25.2
Soft	32.6	20.7	**15.4**	34.1	21.5	**15.9**
Semi-soft (default)	**29.6**	**20.3**	17.9	**31.1**	**21.1**	18.5

A.4 Fast FixMatch: FLOPs Gain versus Wall-Clock Overhead

The FLOPs gain (62.6% less than FixMatch at equal iteration count) confirms the result reported by the authors. The wall-clock overhead (2.13×) neutralizes and reverses this gain on our pipeline: Fast

Table 9: FixMatch (fixed batch) versus Fast FixMatch (CBS), at 3000 identical iterations.

Method	Time	Accumulated GFLOPs	ms/iteration
FixMatch	197.9 s	2 550 300	65.96
Fast FixMatch (CBS)	421.0 s	953 154	140.32
Ratio (CBS / FixMatch)	**2.13**×	**0.37**×	**2.13**×

FixMatch completes these 3000 iterations more than twice as slowly in practice, despite a strictly lower compute budget. The execution log explicitly attributes this overhead to CUDA Graphs recompilation triggered by every new batch shape (9 distinct sizes encountered over the 8–448 range), rather than to an intrinsic compute cost of the method itself.

This diagnostic quantifies the claim in §4.1 on a specific, targeted point rather than on full convergence: CIFAR-10, 250 labels, seed 2312, 3000 iterations, using the standard pipeline applied to all main results (cf. Appendix B).

A.5 RegMixMatch's Adaptive Thresholding on SVHN

RegMixMatch's (Han et al., 2025) adaptive thresholding mechanism follows FreeMatch (Wang et al., 2022b): the reference code (`freematch_utils.consistency_loss`) clamps the self-adaptive threshold to $[0.9, 0.95]$ specifically on SVHN, following a correction the FreeMatch authors document for this dataset; a low threshold early in training there prevents the model from properly clustering unlabeled data. We verify this dependency empirically on both RegMixMatch and our own adaptive-thresholding variant (§6). On SVHN/250 (seed 2312), RegMixMatch without the clamp peaks at 84.65% after 19,000 steps (33.6 min), then degrades to 81.8% by 70,500 steps (119 min), never approaching $\tau_{\text{acc}} = 90\%$ against 90.51% in 5,500 steps (10.3 min) with the clamp in place. Our EfficientMatch+FreeMatch-thresholding variant shows the same pattern: without the clamp, it never reaches 90% on any of the 3 seeds, peaking at 84.2–86.3% before degrading to 82–83% as the mask keeps admitting an increasing share of unreliable pseudo-labels. This is not specific to our implementation: it reproduces a documented, dataset-specific limitation of adaptive thresholding shared by both reference methods that rely on it.

A.6 Equivalence Study at Matched μ

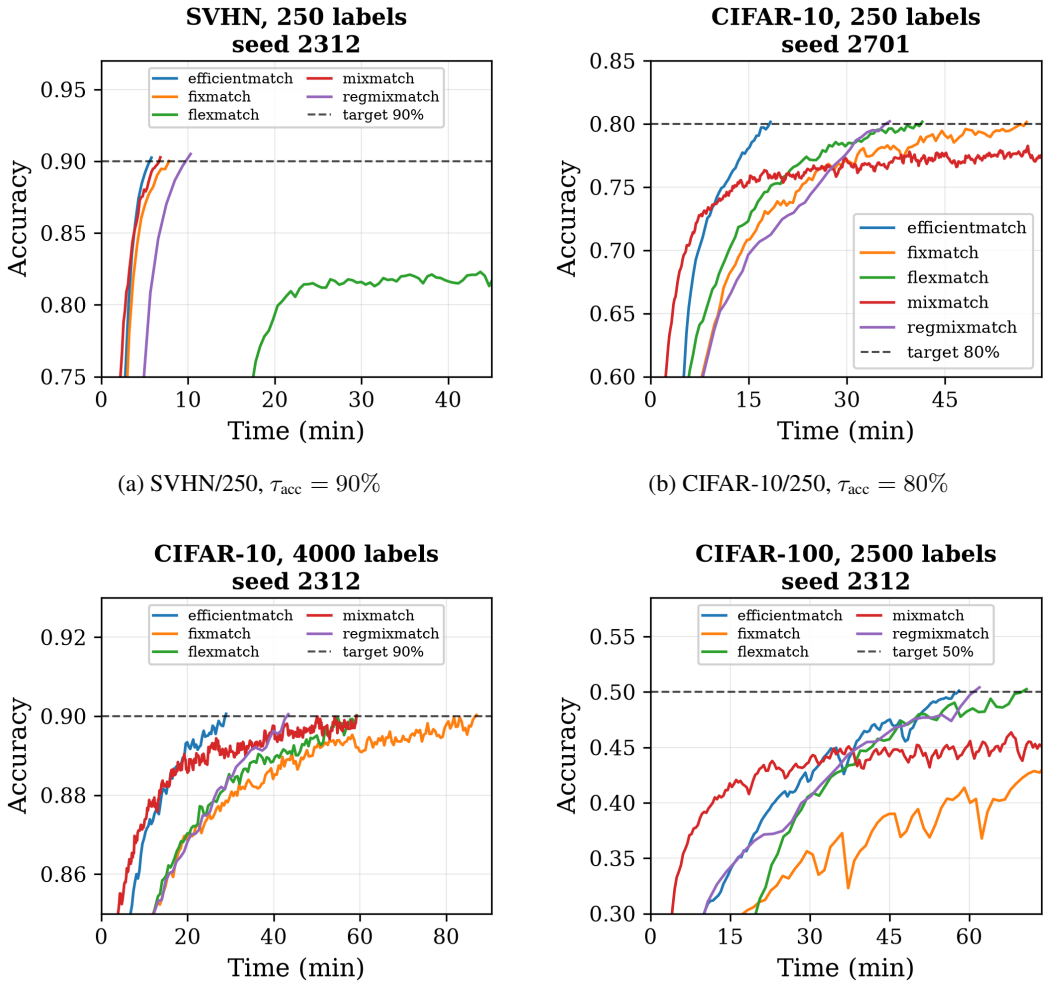
Table 10 accompanies the discussion in §4: to verify that EfficientMatch's advantage does not simply come from its smaller per-iteration unlabeled batch (§3), we run FixMatch and RegMixMatch at $\mu = 3$, matching EfficientMatch, on CIFAR-10/250. At matched μ, FixMatch remains far more expensive than EfficientMatch, compensating for the reduced unlabeled batch with substantially more iterations rather than converging faster overall. RegMixMatch, in contrast, becomes unstable at this reduced μ: it fails to reach $\tau_{\text{acc}} = 80\%$ on two of the three seeds (76.3% and 72.2% maxima), and on the third seed only converges after an initial attempt crashed almost immediately. This confirms that EfficientMatch's Mixup channel supplies a structurally richer per-iteration signal, rather than merely trading off unlabeled-batch size against speed.

Table 10: Steps (x1000), wall-clock time (min), and FLOPs (PFLOPs) for equivalence study at matched $\mu = 3$ on CIFAR-10, 250 labels, $\tau = 80\%$, per seed. **Bold** = best value in the column; underline = second-best († = never reached, excluded from ranking, max accuracy in parentheses).

	Steps			Time			FLOPs		
Method ($\mu = 3$)	2312	0308	2701	2312	0308	2701	2312	0308	2701
FixMatch	202.5	197.0	110.0	93.6	91.3	51.1	172.1	167.5	93.5
RegMixMatch	† (76.3%)	† (72.2%)	35.5	†	†	28.2	†	†	32.1
EfficientMatch	**42.0**	**28.5**	**25.0**	**29.6**	**20.3**	**17.9**	**31.1**	**21.1**	**18.5**

A.7 Method Behavior Across Configurations and Inter-Seed Variability

Method behavior across configurations. Figure 4 shows accuracy vs. time for all five reference methods on seed 2312, one panel per configuration. These illustrate the reasoning behind §4's choice of thresholds and summarize the qualitative patterns discussed throughout §4 and §6: on SVHN, FlexMatch never approaches $\tau_{\text{acc}} = 90\%$ even as other methods converge comfortably; on CIFAR-10/250, MixMatch fails to reach $\tau_{\text{acc}} = 80\%$ within a budget far exceeding what other methods require, while EfficientMatch is fastest; on CIFAR-10/4000 and CIFAR-100/2500, MixMatch is competitive or dominant, illustrating the reversal discussed in §6.



(a) SVHN/250, $\tau_{\text{acc}} = 90\%$

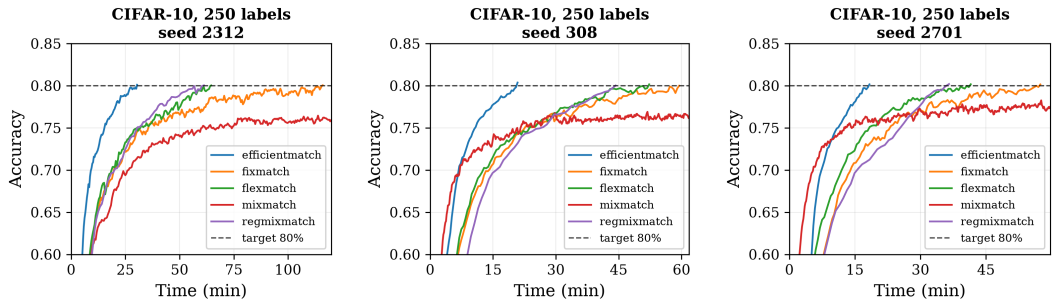
(b) CIFAR-10/250, $\tau_{\text{acc}} = 80\%$

(c) CIFAR-10/4000, $\tau_{\text{acc}} = 90\%$

(d) CIFAR-100/2500, $\tau_{\text{acc}} = 50\%$

Figure 4: Accuracy vs. time, all five reference methods, seed 2312, one panel per configuration. Dashed lines mark τ_{acc}.

Inter-seed variability. Figure 5 shows the same configuration, CIFAR-10/250, with all methods, one panel per seed. Both the ranking and the qualitative behavior of each method change substantially from seed to seed. MixMatch fails to reach τ_{acc} within the 2-hour budget on every seed, yet plateaus at markedly different levels (76.5%, 77.7%, 79.5%; Tab. 2); which is why we do not report averaged performance in this work. This sensitivity stems from the seed controlling which samples are drawn into the labeled subset at the start of training: depending on the seed, this subset is more or less representative of each class, directly shaping how reliable the initial pseudo-labels are and, in turn, how quickly each method converges.



(a) Seed 2312 (b) Seed 0308 (c) Seed 2701

Figure 5: Accuracy vs. time, all methods, CIFAR-10/250, one panel per seed. Panel (a) is identical to Figure 4(b), shown again here for direct seed-to-seed comparison.

A.8 A Preliminary Look at Asymptotic Behavior

To probe how far our speed advantage extends beyond the pre-asymptotic thresholds used elsewhere in this work, we run EfficientMatch and RegMixMatch on CIFAR-10/250 without a target-accuracy stopping criterion, for the full 2^{20}-iteration horizon (seed 42, distinct from the three seeds used throughout the rest of this work, due to the compute cost of full-horizon runs, each requiring over 12 hours). RegMixMatch's run was manually stopped after 12.5 hours, at 40% of its scheduled horizon, with its learning rate still far from annealing; its asymptotic accuracy is therefore not established, and this comparison should be read as indicative only.

EfficientMatch plateaus around 91–92% (best accuracy 91.62%). From 70% to 88% accuracy, its absolute time and FLOP savings over RegMixMatch grow monotonically (Table 11), while the speed factor itself stays in a stable 1.9×–2.6× band. Beyond 88%, evaluation noise (accuracy oscillates by ± 0.3 points) makes crossing times unreliable, and RegMixMatch's incomplete run prevents any conclusion in that range. At matched iteration count rather than matched accuracy, RegMixMatch is instead slightly more accurate from 100,000 steps onward, consistent with its larger $\mu = 7$ extracting more signal per iteration than EfficientMatch's $\mu = 3$, at a proportionally higher per-iteration cost (§3).

Table 11: Absolute time and FLOP savings of EfficientMatch over RegMixMatch to first reach each accuracy milestone, CIFAR-10/250, seed 42 (single seed).

Accuracy	Time saved (min)	FLOPs saved (TFLOPs)
70%	9	9 994
75%	17	18 013
80%	29	31 502
85%	65	70 157
88%	221	236 343

B Implementation Details

Wall-clock time: raw versus corrected. We report two distinct wall-clock measurements to ensure a fairer comparison between methods. *Raw time* is the total elapsed time of a run, from its first training step to the step at which τ_{acc} is reached (or the run is stopped by the 2-hour cap); it includes both training-loop computation and periodic evaluation, which occurs every $2^9 = 512$ iterations for every method and configuration (§4). *Corrected time* subtracts this evaluation overhead:

$$t_{\text{corrected}} = t_{\text{raw}} - n_{\text{eval}} \times c_{\text{eval}},$$

where n_{eval} is the number of evaluation passes performed during the run and c_{eval} is the average wall-clock cost of a single evaluation pass, measured empirically under real conditions (full pass over the evaluation set, same hardware and software stack as training): 553.7 ms for WRN-28-2 and 1,499.1

Table 12: Raw wall-clock time (min) to reach τ_{acc}, 250-label regime, per seed (2312, 0308, 2701). **Bold** = best; underline = second-best († = never reached, excluded from ranking). See Table 2 for corrected time, steps, and FLOPs.

	SVHN, 250l, $\tau = 90\%$			**CIFAR-10, 250l,** $\tau = 80\%$		
Method	2312	0308	2701	2312	0308	2701
MixMatch	6.8	18.4	12.4	†	†	†
FixMatch	7.8	6.4	7.9	116.0	59.4	57.4
FlexMatch	†	†	†	64.6	52.3	41.5
RegMixMatch	10.3	8.2	11.3	61.2	44.3	36.5
EfficientMatch	**5.8**	**5.3**	**6.5**	**30.4**	**20.9**	**18.3**

Table 13: Raw wall-clock time (min) to reach τ_{acc}, per seed (2312, 0308, 2701). **Bold** = best; underline = second-best († = never reached, excluded from ranking, max accuracy in parentheses). See Table 3 for corrected time, steps, and FLOPs.

	CIFAR-10, 4000l, $\tau = 90\%$			**CIFAR-100, 2500l,** $\tau = 50\%$		
Method	2312	0308	2701	2312	0308	2701
MixMatch	59.2	63.6	49.3	† (46.35%)	† (46.48%)	† (45.72%)
FixMatch	87.1	80.8	88.8	† (46.81%)	† (48.13%)	† (47.04%)
FlexMatch	59.2	88.1	78.3	70.8	84.8	69.6
RegMixMatch	43.4	46.7	47.7	61.9	53.3	71.1
EfficientMatch	**29.0**	**30.1**	**30.6**	**58.1**	**49.5**	**59.6**

ms for WRN-28-4. Since evaluation cost is constant across methods for a given architecture and does not reflect any property of the training mechanism being compared, subtracting it isolates the wall-clock cost attributable to the training loop itself; the quantity we intend to capture in §4's discussion of per-iteration overheads (e.g. adaptive thresholding, curricula). **The *Time* column reported in Tables 2 and 3 is corrected time**; raw time remains the reference measurement of total observed run duration and is given per seed in Tables 12 and 13.

Optimizations. This paragraph documents the engineering optimizations applied across all experiments, for reproducibility purposes. They concern runtime and memory usage, not the algorithmic definition of the compared methods (Tab. 2, 3).

- **Automatic convolution algorithm selection** (`torch.backends.cudnn.benchmark`): cuDNN benchmarks several convolution algorithms on the first call and retains the fastest one for fixed-shape tensors; relevant here since our batch shapes (labeled, unlabeled, weak/strong augmentation) remain constant throughout a run.
- **TF32 precision** (`torch.set_float32_matmul_precision("high")`): reduces the precision of FP32 matrix products on Ampere/Ada GPUs, benefiting compute throughput rather than memory usage per se.
- **`torchvision.transforms.v2`** for the augmentation pipeline, faster than the `v1` API on batched tensors rather than individual PIL images.
- **Data loading**: multiple workers, `pin_memory`, and `prefetch_factor` to overlap CPU loading/augmentation with GPU compute and reduce idle time between iterations.
- **Model compilation** (`torch.compile`, `reduce-overhead` mode) before training, to amortize compilation cost over all iterations.
- **Mixed-precision computation** (bfloat16 `autocast`) for the model's forward/backward passes.

Table 14: Hyperparameters used in the experiments. Shared values follow the standard protocol (Sohn et al., 2020; Zhang et al., 2021), except for the iteration budget K (recalibrated for our reduced-budget protocol) and the early-stopping criterion specific to this work.[a]

Hyperparameter	**Value**
Shared across all methods	
Optimizer	SGD, momentum 0.9, Nesterov
Initial learning rate η_0	0.03
Schedule	$\eta = \eta_0 \cos\left(\frac{7\pi k}{16K}\right)$, $K = 2^{20}$
Weight decay	5e-4 (CIFAR-10, SVHN) / 1e-3 (CIFAR-100)
Labeled batch size B	64
Confidence threshold τ	0.95
EMA momentum (evaluation)	0.999
Iteration budget ceiling K	2^{20} (1024 iterations/epoch)
Random seeds	3
Mixup-specific (MixMatch, EfficientMatch)	
α_m (Beta parameter)	0.75
λ_{mix} (EfficientMatch only)	1 (ablation, Appendix 7)
Curriculum Batch Size-specific (Chen et al., 2024)	
Parameter α (B-EXP schedule)	0.7
Minimum batch size	8
Architectures	
Main study	WRN-28-2 (CIFAR-10, SVHN), WRN-28-4 (CIFAR-100)
Unlabeled ratio μ (method-specific, cf. §3)	
EfficientMatch	$\mu = 3$
FixMatch, FlexMatch, RegMixMatch	$\mu = 7$
MixMatch	$\mu = 1$

C HYPERPARAMETERS

[a] $K = 2^{20}$ only shapes the cosine learning-rate schedule; it is never actually reached, since every run stops earlier via the target-accuracy threshold or the 2-hour cap (§4).

D AI DISCLOSURE STATEMENT

In this work, we used generative AI tools to assist with drafting and debugging experiment code, to suggest improvements to the manuscript's writing and structure, and to search for relevant bibliographic references. We did not use generative AI for research ideation, for generating or interpreting experimental results, or for any purpose beyond those listed above. All AI-assisted output; code, text, and reference suggestion; was reviewed, tested, and where necessary corrected or rewritten by the authors before inclusion; in particular, all AI-suggested code was executed and its outputs checked against expected behavior, and all factual or bibliographic claims were verified against primary sources. The authors take full responsibility for the final content of this work, including all text, claims, and artifacts produced with the aid of generative AI.