An Open, Hardware-Independent Protocol for Reproducible Data-Budget Evaluation of Semi-Supervised Learning Methods

Anonymous Authors[1]

Anonymous Institution
anonymous@example.org

Abstract. Comparing semi-supervised learning (SSL) methods reproducibly is harder than it appears. The community has converged on a benchmarking convention—fixed architecture, 2^{20} training iterations, final accuracy on CIFAR-10/SVHN/STL-10—that is easy to re-run but hides a quantity that varies by more than an order of magnitude between methods: the actual amount of data each training run consumes. As a consequence, two implementations that faithfully reproduce the numbers reported in their respective papers can still be incomparable to each other, and to a supervised baseline run on the same hardware. We describe a small, drop-in instrumentation that records the number of processed samples $N_{\text{processed}}$ alongside the standard iteration counter, together with a companion evaluation protocol that fixes a common data budget across methods. The instrumentation is method-agnostic, adds a few lines to an existing training loop, and produces deterministic, hardware-independent traces of the form $\mathcal{A}(N)$. We apply it to MixMatch, FixMatch and SequenceMatch on CIFAR-10, SVHN and STL-10, and report a complete reproducibility record covering software versions, seeds, hyperparameters, divergence cases, and the post-processing pipeline that turns raw traces into the tables and figures of this paper. We then contrast our controlled-budget results with the iteration-based numbers reported in the original papers and observe that the two views can lead to different rankings; the protocol is offered as a complementary, auditable view rather than a replacement. All code, configuration files, seed lists, and figure-generation scripts are released, and the artefact is designed to support the RRPR badge process. The code is available at: https://anonymous.4open.science/r/data-budget-eval-ssl-3758/.

Keywords: Reproducible research · Semi-supervised learning · Benchmarking · Evaluation protocols · Training-loop instrumentation.

1 Introduction

Semi-supervised learning (SSL) is a long-standing tool in the pattern recognition (PR) toolkit. Whenever labelling is expensive—as in hand-annotated document images, biomedical microscopy, biometric enrolment, or fine-grained visual

categorisation—practitioners turn to SSL to exploit the much larger pool of unlabeled samples that is typically available. The recent wave of deep SSL methods, evaluated almost exclusively on the canonical benchmarks CIFAR-10, SVHN and STL-10, has reported steady accuracy gains and, in several cases, performance that approaches or exceeds fully supervised baselines. If robust, these claims matter most under limited annotation, where they suggest that the labelling budget can be reduced significantly without sacrificing recognition performance. This is the regime of many PR applications: histopathology tiles, dermoscopic lesions, satellite scenes, handwritten document images and fine-grained categorisation tasks all share the same profile, namely large unlabeled pools that are cheap to collect and expert annotations that are expensive. The gap between the two is what SSL aims to bridge, which makes the fairness of SSL benchmarking not only a machine-learning concern but a practical one: it conditions whether a benchmark result can be confidently transposed to an application where every label costs domain-expert time.

The primary deliverable of this study is a reproducible artefact: an instrumented training loop, a budget-controlled protocol, and an open release with raw logs and a post-processing pipeline. The methodological discussion is kept light by design, the goal being to offer peers an auditable, complementary protocol that drops into existing benchmarks with minimal effort.

The robustness of SSL accuracy claims depends on how methods are compared, which in turn depends on the reproducibility of the evaluation protocol. Reproducing a published SSL result typically involves three stages: (i) matching the architecture and the optimiser, (ii) matching the loss and the augmentation pipeline, and (iii) training for the prescribed number of iterations and reading off the final accuracy. Stages (i) and (ii) are now well supported by reference implementations and unified benchmarks [17,15]; stage (iii) is where most ambiguity remains, because *the iteration count is not, by itself, a complete specification of a training run.* In SSL, each training step consumes a labeled mini-batch of size B_l together with μB_l unlabeled samples, each typically augmented α times. The total amount of data seen by the model after T iterations is therefore

$$N_{\text{processed}} \;=\; T \cdot B_l \cdot (1 + \alpha\mu), \tag{1}$$

where α and μ are method-specific design choices rather than free hyperparameters. Two faithful re-implementations following the standard $T = 2^{20}$ protocol will thus process datasets of very different sizes, ranging on CIFAR-10 from roughly 0.2 B samples for MixMatch [1] to 1.5 B for SequenceMatch [11]. This discrepancy is not a flaw of any individual paper; it is a property of the shared convention, and one that must be made explicit if results are to be compared on equal terms.

Reproducibility is usually framed as the ability to obtain the same numbers from the same code and seeds [14]. We argue that for benchmarks structured around comparison—which is the case of nearly all SSL papers—a complete reproducibility report should also enable *equivalent* runs across methods, not only *identical* reruns of a single method. Eq. (1) provides the missing accounting

unit: making $N_{\text{processed}}$ explicit costs essentially nothing and lets a reviewer audit comparisons that the iteration count alone cannot. This is the kind of contribution the IAPR TC-22 reproducibility programme has been calling for since its inception [8]: not a new algorithm, but a small, transferable instrument that makes existing PR benchmarks easier to audit.

Contributions. Building on this observation, the main contributions of this paper are the following:

1. a method-agnostic *instrumentation* that augments any SSL training loop with an $N_{\text{processed}}$ counter, together with utilities producing $\mathcal{A}(N)$ traces, summary tables and figures;
2. a *reproduction protocol* that fixes a common data budget across methods, expressed as an iteration schedule derived from Eq. (1), so that the heaviest re-implementation effort (training loops, augmentations, hyperparameters) is left unchanged;
3. a *full reproducibility report* on three standard SSL benchmarks (CIFAR-10, SVHN, STL-10) and three SSL methods (MixMatch, FixMatch, SequenceMatch), covering hardware, software, hyperparameters, seeds, observed divergence cases, and the post-processing pipeline;
4. a well-structured *open release* of the codebase, configuration files and raw logs.

The scope is deliberately narrow—three methods, three seeds per configuration, three datasets—which is sufficient to demonstrate that an $N_{\text{processed}}$-based protocol is implementable, auditable, and produces interpretable traces, but is not a meta-analysis of SSL; broader coverage is tracked in the repository's `ROADMAP.md`[1].

The remainder of this paper is organised as follows. Section 2 situates our contribution within prior work on reproducibility and SSL evaluation. Section 3 describes the instrumentation and the budget-controlled protocol. Section 4 details the experimental setup, and Section 5 reports the controlled-budget results and contrasts them with previously published numbers. Section 6 discusses the scope and implications of the protocol, and Section 7 concludes.

2 Related Work

Reproducibility has been an explicit concern of the PR community well before its recent prominence in the broader ML literature. Long-standing venues such as IPOL (Image Processing On Line) require executable code alongside every paper, and the IAPR TC-22 has been organising the RRPR workshop series since 2016 specifically to encourage companion artefacts in pattern recognition [8]. The introduction of the RRPR Badge at ICPR 2026 [7] extends this tradition by

[1] https://anonymous.4open.science/r/data-budget-eval-ssl-3758/ROADMAP.md

making the reproducibility of accepted PR papers visible to the community. In the broader ML community, the NeurIPS reproducibility checklist [14] and the ML Reproducibility Challenge [13] have formalised the expectations for code, data and experimental description. Our contribution is in the spirit of the RRPR series: a small, transferable artefact that helps reviewers and practitioners audit existing claims, rather than a new method.

Semi-supervised learning has been a recurring topic at PR venues—ICPR, ICIP, BMVC, and IAPR-sponsored workshops—and the methods evaluated here, MixMatch [1], FixMatch [15] and SequenceMatch [11], have become reference points for visual recognition tasks where labelling is the bottleneck. Several works have already pointed to evaluation issues with these benchmarks. Oliver et al. [12] showed that strong supervised baselines and realistic validation set sizes change the SSL/supervised gap substantially. The USB benchmark [17] unifies pipelines across 15 tasks spanning vision, text and audio, and integrates pretrained models to reduce evaluation cost. Su et al. [16] stress-test SSL on long-tailed and out-of-distribution data, conditions that are arguably more representative of real PR applications than balanced CIFAR. Huang et al. [5] replace the fixed-trials hyperparameter search with a fixed wall-clock budget, arguing that wall-clock is a fairer unit when training speed varies; Chen et al. [2] reduce SSL compute via a curriculum on the unlabeled batch size; and FlexMatch [19] shows that per-class adaptive thresholding reaches FixMatch-level accuracy in a fraction of the training time.

Our contribution is complementary to these efforts. Compute-time normalisations such as [5,2] are useful but couple the evaluation to a specific hardware and software stack, and possibly to profiling artefacts; they are difficult to re-run by a third party with a different GPU, a recurring obstacle in PR labs that typically operate on heterogeneous hardware. By contrast, the number of processed samples $N_{\text{processed}}$ in Eq. (1) is fully determined by the training configuration $(B_l,\, \mu,\, \alpha,\, T)$, and is therefore deterministic and hardware-independent. It is also fully compatible with the existing unified benchmarks: USB-style configurations can be re-instrumented in a few lines, and any PR practitioner using one of the public reference implementations can adopt the protocol with minimal effort.

3 Instrumentation and Protocol

The first artefact is a minimal instrumentation of the training loop, designed so that an existing reference implementation (e.g. the public MixMatch, FixMatch or SequenceMatch code) runs unchanged except for two additions: a counter and a logger. At iteration t, after the labeled and unlabeled batches have been assembled, the counter is updated according to

$$N_{\text{processed}}^{(t)} \;=\; N_{\text{processed}}^{(t-1)} \;+\; B_l \;+\; \alpha\,\mu\,B_l, \tag{2}$$

in line with Eq. (1). The constants α, μ and B_l are read from the method configuration rather than hard-coded, so that a method that internally switches between different numbers of augmented views (for instance under a curriculum scheme)

only needs to expose its current α to the counter. The counter is therefore a property of the data pipeline, not of the loss function. The standard evaluation hook already reports test accuracy as a function of t; we add a second log that reports it as a function of $N^{(t)}_{\text{processed}}$, with no additional model forward passes, so that the two views of the same trace are read off the same evaluation calls. The output of a single run is then a JSON file containing tuples $(N^{(t)}_{\text{processed}}, \text{acc}^{(t)})$, sufficient to reconstruct the $\mathcal{A}(N)$ curve and any iteration-based curve without re-running the experiment.

The second artefact is the companion budget-controlled protocol. For a chosen reference budget N_{ref}, typically derived from a supervised training schedule as $N_{\text{ref}} = E \cdot N_{\text{train}}$ for E epochs over a training set of size N_{train}, the iteration schedule of method $\mathcal{M}$ is rescaled as

$$T_{\mathcal{M}} \;=\; \frac{N_{\text{ref}}}{B_l \cdot (1 + \alpha\mu)}. \tag{3}$$

Crucially, all other hyperparameters of the method—learning-rate schedule, ramp-up of the unsupervised loss, EMA decay, and so on—are *not* rescaled. Retuning methods to favour the controlled-budget protocol would defeat the purpose of a reproducibility study and is therefore avoided. One exception is worth noting: cosine annealing is computed with respect to $T_{\mathcal{M}}$ rather than the original 2^{20}. This is flagged in the release notes as a design choice that could be debated; in our experiments it produced smoother $\mathcal{A}(N)$ curves and avoided the artefact of training methods with a near-zero learning rate during most of their schedule.

4 Experimental Setup

This section provides the information necessary to reproduce every number and every figure of Section 5. All experiments were run on a single NVIDIA GeForce RTX 5060 Ti (8 GB VRAM) without distributed training; wall-clock numbers are reported in the release notes for this configuration but are not used in any of the metrics discussed below. The code is implemented in PyTorch, with package versions listed in the `requirements.txt` file of the release.

Datasets and splits. We use three canonical image recognition benchmarks long established in the pattern recognition community: CIFAR-10 [9] (natural object categories), SVHN [10] (street-view digit recognition on noisy real-world imagery), and STL-10 [3] (a small labeled set with a large unlabeled pool, originally designed for the semi-supervised regime). The label-count regimes follow the standard SSL literature: 40, 250 and 4 000 labels for CIFAR-10; 40, 250 and 1 000 for SVHN; and 40 and 1 000 for STL-10. Three labeled/unlabeled splits are sampled per regime using fixed seeds, and *the splits are not class-balanced*, reflecting realistic deployment conditions where labeled data is typically skewed across classes.

Table 1. Augmentation stack used per method. *Weak* denotes *RandomHorizontalFlip + RandomCrop*. *RandAugment(N)* draws N transformations from a fixed pool with uniformly sampled magnitudes.

Method	Augmentation stack
Supervised	Weak
MixMatch	Weak
FixMatch	Weak + RandAugment(2) + Cutout
SequenceMatch	Weak + RandAugment(1) + Cutout + RandAugment(3) + Cutout

Architecture and method configurations. All experiments use WideResNet-28-2 [18], even on STL-10 where some papers use WideResNet-37-2: we prefer a single architecture across datasets to remove a confound, and STL-10 images are resized to 32×32. Optimisation is SGD with momentum 0.9 (Nesterov enabled), weight decay 5×10^{-4} and initial learning rate 0.03, with cosine annealing down to $\eta_{\min} = 0$. No exponential moving average of parameters is used; this departs from some reference implementations and is explicitly logged in the release. Method hyperparameters are taken from the original papers without retuning. MixMatch [1] uses two weak augmentations, temperature $T = 0.5$, unsupervised loss weight $\lambda_u = 100$ ramped linearly over the first 16 000 iterations, and Mixup with Beta($\alpha = 0.75$). FixMatch [15] uses confidence threshold $\tau = 0.95$, unsupervised loss weight $\lambda_u = 1$, and pseudo-labels generated from weak augmentations applied to strongly augmented counterparts. SequenceMatch [11] uses $\tau = 0.95$, $\lambda_u = 1$ and temperature $T = 0.5$ for the prediction-sharpening term inside the KL divergences. Weak augmentation is the standard *RandomHorizontalFlip + RandomCrop*, and Cutout is implemented via PyTorch's `RandomErasing`. The per-method augmentation stacks are summarised in Table 1.

Data budgets and iteration counts. The protocol of Section 3 allocates each method the iteration count that matches a common reference budget N_{ref}, following Eq. (3). Here N_{ref} is derived from a supervised training of E epochs over the full training set, with $E = 100$ for CIFAR-10, 40 for SVHN and 20 for STL-10. The corresponding per-method iteration counts, obtained from Eq. (3) with $B_l = 64$, are summarised in Table 2.

Divergence handling, evaluation cadence, and reporting. SSL methods are known to occasionally diverge in extremely low-label regimes, so to avoid silently corrupting averages we apply two deterministic stopping rules that are identical across all methods. For labeled-only baselines, training stops when test accuracy fails to improve for five consecutive evaluations; for SSL methods in the 40-label regime, training stops when test accuracy remains below 11% for five consecutive evaluations, a clear divergence signature on CIFAR-10, SVHN and STL-10. Diverged runs are reported explicitly in Table 4 via a $^{(k)}$ annotation indicating k out of 3 diverged seeds, and they are kept in the mean±std reports rather than silently dropped, since divergence is itself information about robustness.

Table 2. Iteration counts $T_{\mathcal{M}}$ assigned to each method to enforce a common data budget, computed from Eq. (3) with $B_l = 64$ and N_{ref} corresponding to $E = 100/40/20$ supervised epochs on CIFAR-10/SVHN/STL-10. Values are rounded to the nearest ten.

Method	CIFAR-10	SVHN	STL-10
Supervised	78 100	45 800	32 800
MixMatch	26 040	15 270	10 940
FixMatch	5 210	3 050	2 190
FixMatch ($\mu = 1$)	26 040	15 270	10 940
SequenceMatch	3 550	2 080	1 490
SequenceMatch ($\mu = 1$)	19 530	11 450	8 200

Test accuracy is recorded every 1 500 batches processed on CIFAR-10, 900 on SVHN and 700 on STL-10; these cadences are stored as fields in the configuration files and are dataset-dependent because the per-method iteration budgets differ. Each configuration is run with three fixed seeds (`torch.manual_seed`), tables report mean±std, and figures plot mean curves with shaded $\pm 1\,\sigma$ regions, with no outlier removal and no best-of-N selection. We make no claim that three seeds suffice for a power-analysis-grade comparison of SSL methods; they suffice for the artefact's purpose, namely to demonstrate the protocol and provide an auditable reference trace for each configuration.

Repository. The repository is made of executable Jupyter notebooks: each dataset-related folder contains notebooks that produce results (in JSON format) stored alongside the experiments they correspond to, together with one visualisation notebook. The whole experimental pipeline is explained step by step, and every hyperparameter is straightforward to edit. The artefact is released as a public repository that includes the source code, the raw JSON logs of every reported run, the post-processing pipeline, and a `README.md` describing the repository structure.

5 Results

This section reports the numerical results obtained with the instrumentation of Section 3 and the protocol of Section 4. Its role is twofold: to demonstrate that $N_{\text{processed}}$-based reporting yields an auditable view that complements iteration-based reporting, and to give a reference point against which a reviewer can validate a re-run of the artefact.

We begin with a visual illustration of the two axes. Figure 1 shows the same set of runs plotted against the number of training iterations on the right and against the number of processed samples on the left. Both panels are derived from the same logs, with no re-training: the two views are simply two renderings of the same JSON file. The figure illustrates why $N_{\text{processed}}$ is worth recording, since the two axes can yield different orderings of the same runs; reporting both

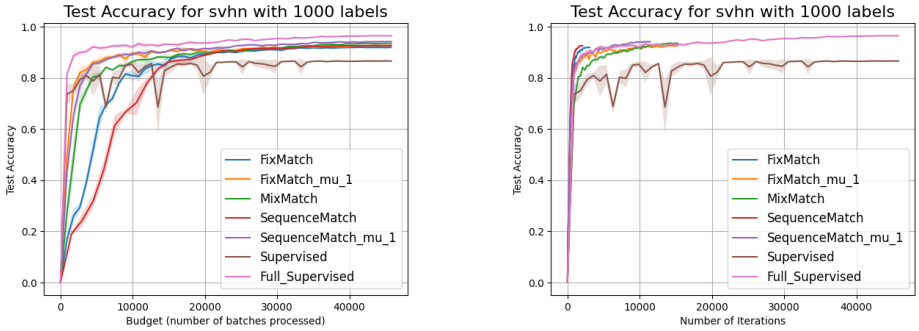

Fig. 1. SVHN, 1000 labels. The same runs plotted against the number of processed samples (left) and the number of iterations (right). The two views can lead to different orderings of the same runs.

Table 3. Error rates (%) as reported in the original papers under the standard $T = 2^{20}$ iteration-based protocol. Values are reproduced verbatim from [4,11,6] and are not re-runs of ours; the supervised line is a re-run from the present artefact.

Dataset	CIFAR-10			SVHN			STL-10	
# Labels	40	250	4000	40	250	1000	40	1000
MixMatch [1]	36.19±6.48	13.63±0.59	6.66±0.26	30.60±8.39	4.56±0.32	3.69±0.37	54.93±0.96	21.70±0.68
FixMatch [15]	7.47±0.28	4.86±0.05	4.21±0.08	3.81±1.18	2.02±0.02	1.96±0.03	35.97±4.14	6.25±0.33
SequenceMatch [11]	4.80±0.01	4.75±0.05	4.15±0.01	1.96±0.23	1.89±0.31	1.79±0.02	15.45±1.40	5.56±0.35
Fully-Supervised		4.62±0.05			2.13±0.01			-

lets a reviewer see which conclusions are stable across the choice of axis and which are not.

For context, Table 3 lists the error rates reported in the original SSL papers under the standard $T = 2^{20}$ iteration-based protocol. These numbers are not re-run by us: they are reproduced verbatim from [4,11,6] and serve as a reference baseline against which the controlled-budget numbers in Table 4 can be contrasted. The supervised line is our own re-run. We deliberately do not attempt to re-run the SSL methods at $T = 2^{20}$ on our hardware: doing so on a single RTX 5060 Ti would require weeks of GPU time per configuration and is orthogonal to the artefact's goal. Auditing the original 2^{20} numbers with sufficient compute would be a worthwhile follow-up and is listed in `ROADMAP.md`.

Turning to the controlled-budget results, Table 4 reports the error rates obtained when every method is allocated the iteration count of Table 2, and Figures 2 and 3–5 show the corresponding $\mathcal{A}(N)$ curves with shaded standard-deviation regions over three seeds. Each cell of Table 4 reports the minimum error rate along the recorded $\mathcal{A}(N)$ trace, averaged over three seeds, taken before any deterministic divergence stop. Two supervised baselines are distinguished: *Supervised (labeled-only)* trains on the same labeled split as the SSL methods under the controlled iteration budget of Table 2, and is the only baseline that competes

Table 4. Minimum error rates (%) under the controlled-budget protocol, mean±std over three seeds. $^{(k)}$ flags $k/3$ divergence during runs. Best per column in bold. See the surrounding text for the definition of *Supervised (labeled-only)* vs. *Fully-Supervised*.

Dataset	CIFAR-10			SVHN			STL-10	
# Labels	40	250	4000	40	250	1000	40	1000
Supervised (labeled-only)	80.28±1.03	56.65±0.25	18.53±0.45	87.22±1.38	28.71±0.74	13.28±0.30	79.51±0.73	46.62±1.19
MixMatch	**66.62**$^{(3)}$±6.88	29.85±3.46	**11.27±0.29**	**66.83±9.43**	12.82±3.33	6.41±0.24	**73.14**$^{(2)}$**±3.12**	37.22±0.80
FixMatch	78.61$^{(3)}$±1.90	42.23±1.88	18.40±0.22	85.74±1.88	21.47±5.25	8.01±0.37	78.18$^{(3)}$±0.92	44.57±1.23
FixMatch ($\mu = 1$)	75.90$^{(2)}$±4.31	29.29±2.14	12.25±0.08	85.38$^{(2)}$±4.25	14.12±0.32	7.14±0.19	78.80$^{(3)}$±1.70	40.10±1.68
SequenceMatch	75.44$^{(1)}$±6.23	33.35±0.83	17.46±0.18	85.88$^{(2)}$±3.88	10.41±0.67	7.34±0.31	81.85$^{(3)}$±1.00	44.59±1.43
SequenceMatch ($\mu = 1$)	72.68$^{(2)}$±10.95	**22.52±0.71**	12.17±0.16	82.84±1.72	**7.12±0.36**	**5.75±0.21**	82.32$^{(3)}$±1.49	**37.13±1.08**
Fully-Supervised	5.67±0.17			3.53±0.07			-	

Table 5. Ranking comparison: rank_s on the previously reported iteration-based numbers of Table 3 vs. rank_e on the controlled-budget results of Table 4. $\Delta_{\text{rank}} = \text{rank}_s - \text{rank}_e$. Largest gains (blue) and largest changes (red) are bolded. The table is informational, not a verdict on SSL: it shows where the two views agree and where they disagree.

Dataset	CIFAR-10			SVHN			STL-10		Mean rank	Mean Δ_{rank}
# Labels	40	250	4000	40	250	1000	40	1000		
MixMatch	4→2	4→2	4→2	4→2	4→3	4→2	3→1	3→1	3.75→1.875	**+1.875**
FixMatch	3→4	3→4	3→4	3→3	3→4	3→4	2→2	2→2	2.75→3.375	−0.625
SequenceMatch	2→3	2→3	1→3	1→4	1→2	1→3	1→3	1→3	1.25→2.5	**−1.25**
Fully-Supervised	1→1	1→1	2→1	2→1	2→1	2→1	-	-	1.6→1	+0.6

with SSL on equal footing; *Fully-Supervised* trains on the full training set (50 k images for CIFAR-10, 73 k for SVHN) with no unlabeled data, and is included only as an upper-performance reference, which is why STL-10 (which has no comparably-sized labeled training set) shows a dash. The best value per column is set in bold, except when the candidate has $k = 3$ diverged runs out of 3, in which case the column receives no bolded entry.

Four observations summarise what the controlled-budget traces show, with the broader interpretation deferred to Section 6. First, method rankings under the controlled-budget protocol differ from rankings under the iteration-based protocol (Table 5); on average across the eight (dataset, label-count) settings, MixMatch's mean rank improves and SequenceMatch's mean rank declines. Second, for both FixMatch and SequenceMatch, the $\mu = 1$ variants match or improve on the standard $\mu = 7$ configurations under matched budgets, and this is consistent across datasets and label counts in our experiments. Third, under matched data budgets, the labeled-only supervised baseline—same labels as the SSL methods, no unlabeled data—is competitive with the evaluated SSL methods on the configurations we ran; the *Fully-Supervised* reference, which uses the entire training set, remains a clear upper bound and is not part of this comparison. We report the labeled-only competitiveness as an observation of the protocol applied to three methods, not as a general claim about SSL. Fourth, divergence in the 40-label

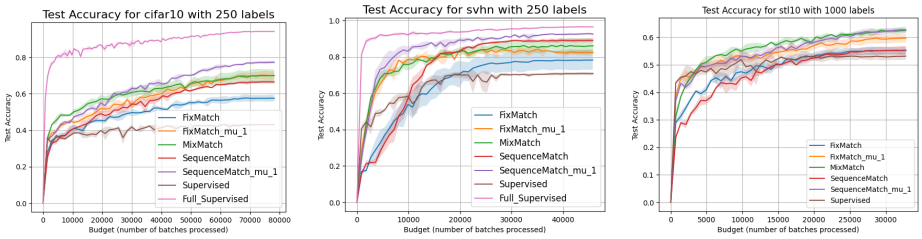

Fig. 2. Data-budget curves $\mathcal{A}(N)$ at 250 labels on CIFAR-10 (left) and SVHN (middle), and at 1000 labels on STL-10 (right). Shaded areas: $\pm 1\,\sigma$ over three seeds.

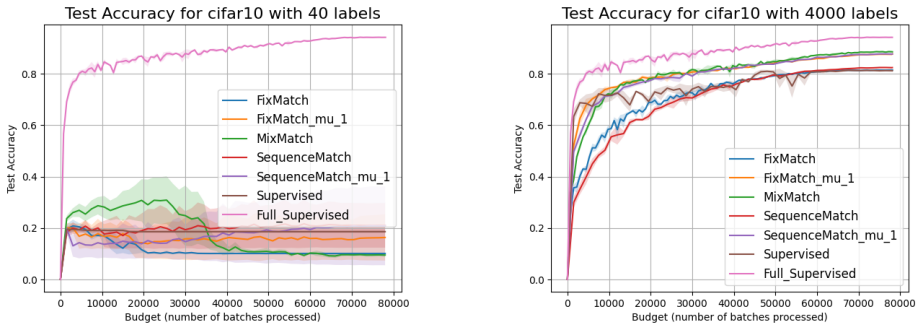

Fig. 3. CIFAR-10, 40 labels (left) and 4000 labels (right). Shaded areas: $\pm 1\,\sigma$ over three seeds.

regime is the rule rather than the exception on CIFAR-10 and STL-10: every SSL configuration sees at least two of three seeds hit the divergence-stopping rule, and the numerical entries in these columns should be read with that caveat in mind, which is why columns dominated by $k = 3$ candidates receive no bolded "best" in Table 4. MixMatch is comparatively more stable in the regimes where the labeled-only baseline already provides a non-trivial signal.

For completeness, Figures 3, 4 and 5 report the data-budget curves $\mathcal{A}(N)$ for the remaining label regimes not shown in Figure 2; the observations summarised above are visible across regimes, and we do not comment on each panel individually.

6 Discussion

Iteration-based reporting and compute-time reporting both have legitimate uses: the former is convenient for reproducing a published configuration, and the latter is informative for deployment. $N_{\text{processed}}$-based reporting sits between them. It is hardware-independent, unlike wall-clock time, but it accounts for the actual learning signal seen by the model, unlike the iteration count, and reporting all three costs nothing once the instrumentation is in place. We therefore do

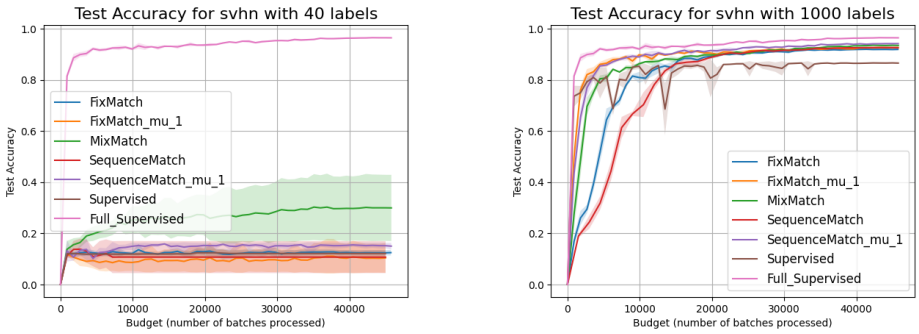

Fig. 4. SVHN, 40 labels (left) and 1000 labels (right). Shaded areas: $\pm 1\,\sigma$ over three seeds.

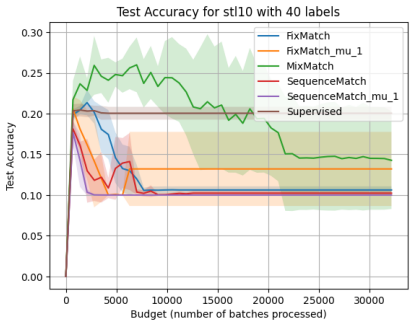

Fig. 5. STL-10, 40 labels. Shaded area: $\pm 1\,\sigma$ over three seeds.

not argue that the controlled-budget view should supersede the standard 2^{20} protocol; we argue that the two views are useful together, and that publishing the raw $N_{\text{processed}}$ trace lets reviewers and follow-up work choose the axis that best fits their question. A processed sample is not, in general, equivalent to an independent training example: multiple augmented views of the same image are correlated, and pseudo-labeled samples carry less reliable signal than labeled ones. $N_{\text{processed}}$ should therefore be read as a measure of exposure rather than of intrinsic information content, and we see this as a feature, because it keeps the metric simple, auditable from the configuration alone, and well-defined across methods.

We do not claim that the controlled-budget protocol is the right way to evaluate SSL in all contexts; we claim that it is a useful additional view, that it costs almost nothing to compute, and that the artefact makes the cost of adopting it negligible for future SSL papers. The scope is deliberately narrow—three SSL methods, three seeds per configuration, three datasets, one architecture, one hardware target—and the artefact is designed to be a reference implementation of the protocol, not a meta-analysis of SSL. The latter would require

many more methods, many more seeds, and ideally re-runs of the original 2^{20} protocol on high-end hardware; these extensions are tracked in `ROADMAP.md`. Beyond image classification, the instrumentation itself is task-agnostic: it depends only on the data pipeline (B_l, μ, α) and not on the loss or the backbone, so it extends naturally to other PR problems where SSL has been used, including document analysis, biometric recognition, biomedical image analysis and fine-grained categorisation, and where labelling cost is typically the limiting factor. Adapting the release to a new task requires only re-pointing the data loader and re-declaring the augmentation count α; these adaptations are listed in the repository's `ROADMAP.md`, alongside extensions to larger architectures (transformer backbones in particular) and longer training schedules.

7 Conclusion

We have described a small, method-agnostic instrumentation of the SSL training loop, a companion protocol that fixes a common data budget across methods, and the open release of the resulting artefact. The instrumentation adds a few lines to existing reference implementations, its output is deterministic and hardware-independent, and the protocol requires no retuning of method hyperparameters. Applied to MixMatch, FixMatch and SequenceMatch on CIFAR-10, SVHN and STL-10, the artefact produces a complete set of controlled-budget results and contrasts them with the iteration-based numbers previously reported in the original papers. The release ships the raw logs, the post-processing pipeline that turns them into the tables and figures of this paper, and the scripts that re-run every experiment from scratch on a single GPU, and the repository is structured to support the RRPR badge verification workflow. Beyond SSL, the same accounting principle—making the actual data exposure explicit, separately from the iteration count—is relevant to any pattern recognition pipeline that combines labeled and unlabeled (or weakly labeled) inputs, a regime that has become common across the modern PR landscape.

Acknowledgements. Acknowledgments omitted for double-blind review.

References

1. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.: MixMatch: A holistic approach to semi-supervised learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2019)
2. Chen, Y., et al.: Fast FixMatch: Faster semi-supervised learning with curriculum batch size. In: European Conference on Computer Vision (ECCV) (2024)
3. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: International Conference on Artificial Intelligence and Statistics (AISTATS) (2011)
4. Han, H., et al.: RegMixMatch: Optimizing mixup for semi-supervised learning. In: AAAI Conference on Artificial Intelligence (2025)

5. Huang, Y., et al.: A systematic benchmarking analysis of transfer learning for medical image analysis and semi-supervised learning. arXiv preprint arXiv:2403.xxxxx (2024)
6. Huang, Z., et al.: FlatMatch: Bridging labeled and unlabeled data with cross-sharpness for semi-supervised learning. In: Advances in Neural Information Processing Systems (NeurIPS) (2023)
7. ICPR 2026 Reproducibility Chairs: ICPR 2026 reproducible research in pattern recognition (RRPR) badge. https://icpr2026.org/rrprBadges.html (2026), accessed 2026
8. Kerautret, B., Colom, M., Lopresti, D., Monasse, P., Talbot, H.: Reproducible research in pattern recognition: Overview and goals. In: Reproducible Research in Pattern Recognition (RRPR), LNCS. Springer (2019)
9. Krizhevsky, A.: Learning multiple layers of features from tiny images. Tech. rep., University of Toronto (2009)
10. Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., Ng, A.Y.: Reading digits in natural images with unsupervised feature learning. In: NIPS Workshop on Deep Learning and Unsupervised Feature Learning (2011)
11. Nguyen, K.B., Lee, J.: SequenceMatch: Revisiting the design of weak-strong augmentations for semi-supervised learning. In: IEEE/CVF Winter Conference on Applications of Computer Vision (WACV) (2024)
12. Oliver, A., Odena, A., Raffel, C., Cubuk, E.D., Goodfellow, I.J.: Realistic evaluation of deep semi-supervised learning algorithms. In: Advances in Neural Information Processing Systems (NeurIPS) (2018)
13. Pineau, J., Sinha, K., Forde, J.Z., Pal, C.: The ML reproducibility challenge. https://paperswithcode.com/rc2020 (2019)
14. Pineau, J., Vincent-Lamarre, P., Sinha, K., Larivière, V., Beygelzimer, A., d'Alché Buc, F., Fox, E., Larochelle, H.: Improving reproducibility in machine learning research (a report from the NeurIPS 2019 reproducibility program). Journal of Machine Learning Research **22**(164), 1–20 (2021)
15. Sohn, K., Berthelot, D., Li, C.L., Zhang, Z., Carlini, N., Cubuk, E.D., Kurakin, A., Zhang, H., Raffel, C.: FixMatch: Simplifying semi-supervised learning with consistency and confidence. In: Advances in Neural Information Processing Systems (NeurIPS) (2020)
16. Su, J.C., Cheng, Z., Maji, S.: Realistic evaluation of semi-supervised learning algorithms in open environments. In: British Machine Vision Conference (BMVC) (2021)
17. Wang, Y., Chen, H., Heng, Q., Hou, W., Fan, Y., Wu, Z., Wang, J., Savvides, M., Shinozaki, T., Raj, B., Schiele, B., Xie, X.: USB: A unified semi-supervised learning benchmark for classification. In: Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track (2022)
18. Zagoruyko, S., Komodakis, N.: Wide residual networks. In: British Machine Vision Conference (BMVC) (2016)
19. Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., Shinozaki, T.: FlexMatch: Boosting semi-supervised learning with curriculum pseudo labeling. In: Advances in Neural Information Processing Systems (NeurIPS) (2021)