USB: A Unified Semi-supervised Learning Benchmark for Classification

Yidong Wang^{1,2,3*}, **Hao Chen**^{4*}, **Yue Fan**^{5*}, **Wang Sun**^{6}, **Ran Tao**^{4}, **Wenxin Hou**^{7}, **Renjie Wang**^{8}, **Linyi Yang**^{2}, **Zhi Zhou**^{8}, **Lan-Zhe Guo**^{8}, **Heli Qi**^{9}, **Zhen Wu**^{8}, **Yu-Feng Li**^{8}, **Satoshi Nakamura**^{9}, **Wei Ye**^{10}, **Marios Savvides**^{4}, **Bhiksha Raj**^{4}, **Takahiro Shinozaki**^{3}, **Bernt Schiele**^{5}, **Jindong Wang**^{1†}, **Xing Xie**^{1}, **Yue Zhang**^{2†}

^{1}Microsoft Research Asia, ^{2}Westlake University, ^{3}Tokyo Institute of Technology, ^{4}Carnegie Mellon University, ^{5}Max-Planck-Institut für Informatik, ^{6}Tsinghua University, ^{7}Microsoft STCA, ^{8}Nanjing University, ^{9}Nara Institute of Science and Technology, ^{10}Peking University

Abstract

Semi-supervised learning (SSL) improves model generalization by leveraging massive unlabeled data to augment limited labeled samples. However, currently, popular SSL evaluation protocols are often constrained to computer vision (CV) tasks. In addition, previous work typically trains deep neural networks from scratch, which is time-consuming and environmentally unfriendly. To address the above issues, we construct a Unified SSL Benchmark (USB) for classification by selecting 15 diverse, challenging, and comprehensive tasks from CV, natural language processing (NLP), and audio processing (Audio), on which we systematically evaluate the dominant SSL methods, and also open-source a modular and extensible codebase for fair evaluation of these SSL methods. We further provide the pre-trained versions of the state-of-the-art neural models for CV tasks to make the cost affordable for further tuning. USB enables the evaluation of a single SSL algorithm on more tasks from multiple domains but with less cost. **Specifically, on a single NVIDIA V100, only 39 GPU days are required to evaluate FixMatch on 15 tasks in USB while 335 GPU days (279 GPU days on 4 CV datasets except for ImageNet) are needed on 5 CV tasks with TorchSSL.**

1 Introduction

Neural models give competitive results when trained using supervised learning on sufficient high-quality labeled data [1, 2, 3, 4, 5, 6, 7]. However, it can be laborious and expensive to obtain abundant annotations for model training [8, 9]. To address this issue, **semi-supervised learning (SSL)** emerges as an effective paradigm to improve model generalization with limited labeled data and massive unlabeled data [10, 11, 12, 13, 14, 15].

SSL has made remarkable progress in recent years [16, 17, 18, 19, 20, 21], yet there are still several limitations with the popular evaluation protocol in the literature [22, 20, 21]. First, existing benchmarks are mostly constrained to plain computer vision (CV) tasks (i.e., CIFAR-10/100, SVHN, STL-10, and ImageNet classification [22, 23, 20, 24, 21], as summarized in TorchSSL [21]), precluding consistent and diverse evaluation over tasks in natural language processing (NLP), audio processing (Audio), etc., where the lack of labeled data is a general issue and SSL has gained increasing research attention recently [25, 26, 27]. Second, the existing protocol (e.g., TorchSSL [21]) can be mainly time-consuming and environmentally unfriendly because it typically trains deep neural

*Equal contribution. Yidong Wang did this work during his internship at MSRA and Westlake University.

†Correspondence to: jindong.wang@microsoft.com, zhangyue@westlake.edu.cn.

Table 1: A summary of datasets and training cost used in (a) the existing popular protocol and (b) USB. USB largely reduces the training cost while providing a diverse, challenging, and comprehensive benchmark covering a wide range of datasets from various domains. Training cost is estimated by using FixMatch [20] on a single NVIDIA V100 GPU from Microsoft Azure Machine Learning platform, except for ImageNet where 4 V100s are used. Experiments in (a) follow the settings in [21]. More results with different pre-trained backbones are available in Appendix D.

(a) TorchSSL [21]

Domain & Backbone	Dataset	Classification Task	Hours × Settings × Seeds	Total GPU Hours	Total GPU Hours w/o ImageNet
CV, ResNets	CIFAR-10	Natural Image	$110 \times 3 \times 3$	8031 GPU Hours (335 GPU Days)	6687 GPU Hours (279 GPU Days)
	CIFAR-100	Natural Image	$300 \times 3 \times 3$		
	SVNH	Digital	$108 \times 3 \times 3$		
	STL-10	Natural Image	$225 \times 3 \times 3$		
	ImageNet	Natural Image	336 hours × 4 GPUs		

(b) USB

Domain & Backbone	Dataset	Classification Task	Hours × Settings × Seeds	Total GPU Hours
CV, ViTs	CIFAR-100	Natural Image	$11 \times 2 \times 3$	924 GPU Hours (39 GPU Days)
	STL-10	Natural Image	$18 \times 2 \times 3$	
	EuroSAT	Satellite Image	$10 \times 2 \times 3$	
	TissueMNIST	Medical Image	$8 \times 2 \times 3$	
	Semi-Aves	Fine-grained, Long-tailed Natural Image	$13 \times 1 \times 3$	
NLP, Bert	IMDB	Movie Review Sentiment	$8 \times 2 \times 3$	
	AG News	News Topic	$6 \times 2 \times 3$	
	Amazon Review	Product Review Sentiment	$8 \times 2 \times 3$	
	Yahoo! Answer	QA Topic	$7 \times 2 \times 3$	
	Yelp Review	Restaurant Review Sentiment	$8 \times 2 \times 3$	
Audio, Wave2Vec 2.0 and HuBert	GTZAN	Music Genre	$12 \times 2 \times 3$	
	UrtraSound8k	Urban Sound Event	$15 \times 2 \times 3$	
	FSDnoisy18k	Sound Event	$17 \times 1 \times 3$	
	Keyword Spotting	Keyword	$10 \times 2 \times 3$	
	ESC-50	Environmental Sound Event	$18 \times 2 \times 3$	

models from scratch [28, 23, 29, 20, 24, 21]. Specifically, as shown in Table 1a, it takes about 335 GPU days (279 GPU days without ImageNet) to evaluate FixMatch [20] with TorchSSL [21]. Such a high cost can make it unaffordable for research labs (particularly in academia) to conduct SSL research. Recently, the pre-training and fine-tuning paradigm [30, 31, 32, 33] achieves promising results. Compared with training from scratch, pre-training has much reduced cost in SSL. However, there are relatively few benchmarks that offer a fair test bed for SSL with the pre-trained versions of neural models.

To address the above issues and facilitate general SSL research, we propose **USB: a Unified SSL Benchmark** for classification [3]. USB offers a *diverse* and *challenging* benchmark across five CV datasets, five NLP datasets, and five Audio datasets (Table 1b), enabling consistent evaluation over multiple tasks from different domains. Moreover, USB provides comprehensive evaluations of SSL algorithms with even fewer labeled data compared with TorchSSL, as the performance gap between SSL algorithms diminishes when the amount of labeled samples becomes large. Benefiting from the rapidly developed neural architectures, we introduce pre-trained Transformers [4] into SSL instead of training ResNets [1] from scratch to reduce the training cost for CV tasks. Specifically, we find that using pre-trained Vision Transformers (ViT) [34] can largely reduce the number of training iterations (e.g., by 80% from 1,000k to 200k on CV tasks) without hurting the performance, and most SSL algorithms achieve even better performance with less training iterations.

As illustrated in Table 1b, using USB, we spend only **39 GPU days** to evaluate the performance of an SSL algorithm (i.e., FixMatch) on a single NVIDIA V100 over these **15 datasets**, in contrast to TorchSSL, which costs about **335 GPU days** on only **5 CV datasets** (279 GPU days on 4 CV datasets except for ImageNet). To further facilitate SSL research, we open-source the codebase and pre-trained models [4] for unified and consistent evaluation of SSL methods. In addition, we also provide config files that contain all the hyper-parameters to easily reproduce our results reported in

[3]The word ‘unified’ means the unification of different algorithms on various application domains.

[4]`https://github.com/microsoft/Semi-supervised-learning`. We also provide the training logs of the experiments in this paper. Note that the results and training logs will be continuously updated/provided if we reorganize the codes for better use or add more algorithms and datasets. Microsoft Research Asia (MSRA) will provide both the support and resources for future updates.

Table 2: The comparison between USB and other related benchmarks.

Benchmark	# SSL algorithms	Domian	# Tasks	Pre-trained	Training hours using FixMatch
Realistic SSL evaluation [22]	4	CV	3	✗	-
TorchSSL [21]	9	CV	5	✗	6687
USB	14	CV, NLP, Audio	15	✓	924

this work. We obtain some interesting findings by evaluating 14 SSL algorithms (Section 5.4): (1) introducing diverse tasks from diverse domains can be beneficial to comprehensive evaluation of an SSL algorithm; (2) pre-training is more efficient and can improve the generalization; (3) unlabeled data do not consistently improve the performance especially when labeled data is scarce.

To conclude, our contributions are three-fold:

- We propose USB: a unified and challenging semi-supervised learning benchmark for classification with 15 tasks on CV, NLP, and Audio for fair and consistent evaluations. To our humble knowledge, we are the first to discuss whether current SSL methods that work well on CV tasks generalize to NLP and Audio tasks.
- We provide an environmentally friendly and low-cost evaluation protocol with pre-training & fine-tuning paradigm, reducing the cost of SSL experiments. The advantages of USB as compared to other related benchmarks are shown in Table 2.
- We implement 14 SSL algorithms and open-source a modular codebase and config files for easy reproduction of the reported results in this work. we also provide documents and tutorials for easy modification. Our codebase is extensible and open for continued development through community effort, where we expect new algorithms, models, config files and results are constantly added.

2 Related Work

Deep semi-supervised learning originates from Π model [35], where it solves the task of image classification by using consistency regularization that forces the model to output similar predictions when fed two augmented versions of the same unlabeled data. Subsequent methods can be classified as the variants of Π model, where the difference lies in enforcing the consistency between model perturbation [36], data perturbation [37, 29], and exploiting unlabeled data [20, 21]. Since the best results in both CV and NLP are given by such algorithms, we choose them as typical representative methods in USB. While most SSL methods have seen their use in CV tasks, NLP has witnessed recent growth in SSL solutions [29, 25]. However, only some of the popular methods [29] in CV have been used in the NLP literature, probably because other methods give lower results or have not been investigated. This gives us motivation for evaluation of SSL methods on various domains in USB.

As shown in Table 2, related benchmarks include Realistic SSL evaluation [22] and TorchSSL [21]. Realistic SSL evaluation [22] has 4 SSL algorithms and 3 CV classification tasks and TorchSSL has 9 SSL algorithms and 5 CV classification tasks. Both of them are no longer maintained/updated. Thus it is of significance to build an SSL community that can continuously update SSL algorithms and neural models to boost the development of SSL. Besides, previous benchmarks mainly train the models from scratch, which is computation expensive and time consuming, since SSL algorithms are known to be difficult to converge [38]. In USB, we consider using pre-trained models to boost the performance while being more efficient and friendly to researchers.

In the following, we will first introduce the tasks, datasets, algorithms, and benchmark results of USB. Then, the codebase structure of USB will be presented in Section 6.

3 Tasks and Datasets

USB consists of 15 datasets from CV, NLP, and Audio domains. Every dataset in USB is under a permissive license that allows usage for research purposes. The datasets are chosen based on the following considerations: (1) the tasks should be diverse and cover multiple domains; (2) the tasks should be challenging, leaving room for improvement; (3) the training is reasonably environmentally friendly and affordable to research labs (in both the industry and academia).

Table 3: Details of the datasets in USB. Two *#Label per class* settings are chosen for each dataset except Semi-Aves and FSDnoisy18k, which have long-tailed distributed data. Labeled data are sampled from the training data for each dataset except STL-10, Semi-Aves, and FSDNoisy18k, where the split of labeled and unlabeled data is pre-defined (e.g. 5,959 labeled images and 26,640 unlabeled images in Semi-Aves). Following [20, 21], validation data are not provided for CV datasets. The NLP validation data are sampled from the original training datasets. All test sets are kept unchanged.

Domain	Dataset	#Label per class	#Training data	#Validation data	#Test data	#Class
CV	CIFAR-100	2 / 4	50,000	-	10,000	100
	STL-10	4 / 10	5,000 / 100,000	-	8,000	10
	EuroSat	2 / 4	16,200	-	5,400	10
	TissueMNIST	10 / 50	165,466	-	47,280	8
	Semi-Aves	15-53	5,959 / 26,640	-	4,000	200
NLP	IMDB	10 / 50	23,000	2,000	25,000	2
	Amazon Review	50 / 200	250,000	25,000	65,000	5
	Yelp Review	50 / 200	250,000	25,000	50,000	5
	AG News	10 / 50	100,000	10,000	7,600	4
	Yahoo! Answer	50 / 200	500,000	50,000	60,000	10
Audio	Keyword Spotting	5 / 20	18,538	2,577	2,567	10
	ESC-50	5 / 10	1,200	400	400	50
	UrbanSound8k	10 / 40	7,079	816	837	10
	FSDnoisy18k	52-171	1,772 / 15,813	-	947	20
	GTZAN	10 / 40	7,000	1,500	1,500	10

3.1 CV Tasks

The details of the CV datasets are shown in Table 3. We include CIFAR-100 [39] and STL-10 [40] from TorchSSL since they are still challenging. The TissueMNIST [41, 42], EuroSAT [43, 44], and Semi-Aves [45] are datasets in the domains of medical images, satellite images, and fine-grained natural images. CIFAR-10 [39] and SVHN [46] in TorchSSL are not included in USB because the state-of-the-art SSL algorithms [29, 20, 24] have achieved similar performance on these datasets to fully-supervised training with abundant fully labeled training data [5]. SSL algorithms have a relatively large room for improvement on all chosen CV datasets in USB. More details of these CV datasets in USB can be found in Appendix E.1.

3.2 NLP Tasks

The detailed dataset statistics of NLP tasks in USB are described in Table 3. We mostly followed previous work in the NLP literature, and thus the existing datasets in USB cover most test sets used in the existing work [25, 48, 29]. We include widely used IMDB [49], AG News [50], and Yahoo! Answer [51] from the previous protocol [25, 48, 29], which are still challenging for SSL. Since IMDB is a binary sentiment classification task, we further add Amazon Review [52] and Yelp Review [53] to evaluate SSL algorithms on more fine-grained sentiment classification tasks. DBpedia is removed from the previous protocol [25, 48, 29] because we find that the state-of-the-art SSL algorithms have achieved similar performance on it when compared with fully-supervised training. For all tasks in NLP, we obtain the labeled datasets, unlabeled datasets, and validation sets by randomly sampling from their original training datasets while keeping the original test datasets unchanged, mainly following previous work [25, 48]. More details are in Appendix E.2.

3.3 Audio Tasks

USB includes five audio classification datasets as shown in Table 3. We choose the tasks to cover different domains such as urban sound (UrbanSound8k [54], ESC-50 [55], and FSDNoisy18k [56]), human sound (Keyword Spotting [57]), and music (GTZAN) [58]. All chosen datasets are challenging even for state-of-the-art SSL algorithms. For example, FSDNoisy18k is a realistic dataset containing a small labeled set and a large unlabeled set. To the best of our knowledge, we are the first to systematically evaluate SSL algorithms on Audio tasks. Although there is a concurrent work [27], our study includes more algorithms and more datasets than [27]. More details are in Appendix E.3.

[5]We highly recommend reporting ImageNet [8] results since it is a reasonable dataset for hill-climbing [20, 47, 21]. We also report and discuss ImageNet results in Appendix C.

Table 4: Essential components used in 14 SSL algorithms supported in USB. PL, CR, Dist. Align., and W-S Aug., MSE, CE are the abbreviations for Pseudo Labeling, Consistency Regularization, Distribution Alignment, Weak-Strong Augmentation, Mean Squared Error, and Cross-Entropy, respectively. PL denotes hard 'one-hot' labels adopted in CR Loss.

Algorithm	PL	CR Loss	Thresholding	Dist. Align.	Self-supervised	Mixup	W-S Aug.
Π-Model		MSE					
Pseudo Labeling	✓	CE					
Mean Teacher		MSE					
VAT		CE					
MixMatch		MSE				✓	
ReMixMatch		CE		✓	Rotation	✓	✓
UDA		CE	✓				✓
FixMatch	✓	CE	✓				✓
Dash	✓	CE	✓				✓
CoMatch	✓	CE	✓	✓	Contrastive		✓
CRMatch	✓	CE	✓		Rotation		✓
FlexMatch	✓	CE	✓				✓
AdaMatch	✓	CE	✓	✓			✓
SimMatch	✓	CE	✓	✓	Contrastive		✓

4 SSL Algorithms

We implement 14 SSL algorithms in the codebase for USB, including Π model [35], Pseudo Labeling [59], Mean Teacher [36], VAT [37], MixMatch [28], ReMixMatch [23], UDA [29], FixMatch [20], Dash [24], CoMatch [60], CRMatch [61], FlexMatch [21], AdaMatch [62], and SimMatch [47], all of which exploit unlabeled data by encouraging invariant predictions to input perturbations [13, 14, 63, 64, 65, 66, 67]. Such consistency regularization methods give the strongest performance in SSL since the model is robust to different perturbed versions of unlabeled data, satisfying the smoothness and low-density assumptions in SSL [68].

The above SSL algorithms use Cross-Entropy (CE) loss on labeled data but differ in the way on unlabeled data. As shown in Table 4, Pseudo Labeling [59] turns the predictions of the unlabeled data into hard 'one-hot' labels and treats the 'one-hot' pseudo-labels as the supervision signals. Thresholding reduces the noisy pseudo labels by masking out the unlabeled samples whose maximum probabilities are smaller than the pre-defined threshold. Distribution Alignment aims to correct the output distribution to make it more in line with the target distribution (e.g., uniform distribution). Self-supervised learning, Mixup, and Stronger augmentations techniques also can help learn better representation. More details of these algorithms can be found in Appendix F. We summarize the key components exploited in the implemented consistency regularization based algorithms in Table 4.

5 Benchmark Results

[6] For CV tasks, we follow [21] to report the best number of all checkpoints to avoid unfair comparisons caused by different convergence speeds. For NLP and Audio tasks, we choose the best model using the validation datasets and then evaluate it on the test datasets. In addition to mean error rate over the tasks, we use Friedman rank [69, 70] to fairly compare the performance of different algorithms in various settings:

$$\mathrm{rank}_F = \frac{1}{m}\sum_{i=1}^{m}\mathrm{rank}_i,$$

where m is the number of evaluation settings (i.e., how many experimental settings we use, e.g., $m = 9$ in Table 5), and rank_i is the rank of an SSL algorithm in the i-th setting. We re-rank all algorithms to give final ranks based on their Friedman rankings. Note that all ranks are in ascending order because the lower error rate corresponds to a better performance. The experimental setup is detailed in Appendix G. Note that 'supervised' denotes training with the partially chosen labeled data while 'fully-supervised' refers to training using all data with full annotations in our reported results.

[6]**Note that all experimental results and training logs will be continuously updated in** `https://github.com/microsoft/Semi-supervised-learning`**. Please refer to the latest results for comparison.**

Table 5: Error rate (%) and Rank with CV tasks in USB. For Semi-Aves and STL10, as they have unlabeled sets, we do not report the fully-supervised results. We follow [20, 21, 29] to show error rates as default.

Dataset	CIFAR-100		STL-10		Euro-SAT		TissueMNIST		Semi-Aves	Friedman	Final	Mean
# Label	200	400	20	40	20	40	80	400	5,959	rank	rank	error rate
Fully-Supervised	8.44$_{\pm0.07}$	8.44$_{\pm0.07}$	-	-	0.94$_{\pm0.07}$	0.89$_{\pm0.05}$	29.15$_{\pm0.13}$	29.10$_{\pm0.02}$	-	-	-	-
Supervised	35.63$_{\pm0.36}$	26.08$_{\pm0.50}$	47.02$_{\pm1.48}$	26.02$_{\pm0.72}$	27.12$_{\pm1.26}$	16.90$_{\pm1.48}$	59.91$_{\pm2.93}$	54.10$_{\pm1.52}$	41.55$_{\pm0.29}$	-	-	-
Π-model	36.24$_{\pm0.27}$	26.49$_{\pm0.64}$	44.38$_{\pm1.59}$	25.76$_{\pm2.37}$	24.51$_{\pm1.02}$	11.58$_{\pm1.32}$	56.79$_{\pm5.91}$	**47.50$_{\pm1.71}$**	39.23$_{\pm0.36}$	10.11	11	34.72
Pseudo-Labeling	33.16$_{\pm1.20}$	25.29$_{\pm0.67}$	45.13$_{\pm4.08}$	26.20$_{\pm1.53}$	23.64$_{\pm0.90}$	15.61$_{\pm2.51}$	56.22$_{\pm4.01}$	50.36$_{\pm1.62}$	40.13$_{\pm0.09}$	9.89	10	35.08
Mean Teacher	35.61$_{\pm0.38}$	25.97$_{\pm0.37}$	39.94$_{\pm1.99}$	20.16$_{\pm1.25}$	26.51$_{\pm1.15}$	17.05$_{\pm2.07}$	61.40$_{\pm2.48}$	55.22$_{\pm2.06}$	38.52$_{\pm0.27}$	10.89	14	35.60
VAT	31.61$_{\pm1.37}$	21.29$_{\pm0.32}$	52.03$_{\pm0.48}$	23.10$_{\pm0.72}$	24.77$_{\pm1.94}$	9.30$_{\pm1.23}$	58.50$_{\pm6.41}$	51.31$_{\pm1.66}$	39.00$_{\pm0.30}$	10.11	12	34.55
MixMatch	37.43$_{\pm0.58}$	26.17$_{\pm0.24}$	48.98$_{\pm1.41}$	25.56$_{\pm3.00}$	29.86$_{\pm2.89}$	16.39$_{\pm3.17}$	55.73$_{\pm2.29}$	49.08$_{\pm1.06}$	37.22$_{\pm0.15}$	10.11	12	36.27
ReMixMatch	**20.85$_{\pm1.42}$**	**16.80$_{\pm0.59}$**	30.61$_{\pm3.47}$	18.33$_{\pm1.98}$	**4.53$_{\pm1.60}$**	**4.10$_{\pm0.37}$**	59.29$_{\pm5.16}$	52.92$_{\pm3.93}$	**30.40$_{\pm0.33}$**	4.00	1	26.43
UDA	30.75$_{\pm1.03}$	19.94$_{\pm0.32}$	39.22$_{\pm2.87}$	23.59$_{\pm2.97}$	11.15$_{\pm1.20}$	5.99$_{\pm0.75}$	55.88$_{\pm3.26}$	51.42$_{\pm2.05}$	32.55$_{\pm0.26}$	6.89	7	30.05
FixMatch	30.45$_{\pm0.65}$	19.48$_{\pm0.93}$	42.06$_{\pm3.94}$	24.05$_{\pm1.79}$	12.48$_{\pm2.57}$	6.41$_{\pm1.64}$	55.95$_{\pm4.06}$	50.93$_{\pm1.23}$	31.74$_{\pm0.33}$	6.56	6	30.39
Dash	30.19$_{\pm1.34}$	18.90$_{\pm0.42}$	43.34$_{\pm1.46}$	25.90$_{\pm0.35}$	9.44$_{\pm0.75}$	7.00$_{\pm1.39}$	57.00$_{\pm2.81}$	50.93$_{\pm1.54}$	32.56$_{\pm0.39}$	7.44	9	30.58
CoMatch	35.68$_{\pm0.54}$	26.10$_{\pm0.09}$	**29.70$_{\pm1.17}$**	21.46$_{\pm1.34}$	5.25$_{\pm0.49}$	4.89$_{\pm0.86}$	57.15$_{\pm3.46}$	51.83$_{\pm0.71}$	41.39$_{\pm0.16}$	7.22	8	30.38
CRMatch	29.43$_{\pm1.11}$	18.50$_{\pm0.26}$	30.55$_{\pm2.01}$	**17.43$_{\pm1.96}$**	14.52$_{\pm1.34}$	7.00$_{\pm0.69}$	**54.84$_{\pm3.05}$**	51.10$_{\pm1.59}$	31.97$_{\pm0.10}$	4.67	2	28.37
FlexMatch	27.08$_{\pm0.90}$	17.67$_{\pm0.66}$	37.58$_{\pm2.97}$	23.40$_{\pm1.50}$	7.07$_{\pm2.32}$	5.58$_{\pm0.57}$	57.23$_{\pm2.50}$	52.06$_{\pm1.78}$	33.09$_{\pm0.16}$	6.44	5	28.97
AdaMatch	21.27$_{\pm1.04}$	17.01$_{\pm0.55}$	36.25$_{\pm1.89}$	23.30$_{\pm0.73}$	5.70$_{\pm0.37}$	4.92$_{\pm0.87}$	57.87$_{\pm4.47}$	52.28$_{\pm0.79}$	31.54$_{\pm0.10}$	5.22	3	27.79
SimMatch	23.26$_{\pm1.25}$	16.82$_{\pm0.40}$	34.12$_{\pm1.63}$	22.97$_{\pm2.04}$	6.88$_{\pm1.77}$	5.86$_{\pm1.07}$	57.91$_{\pm4.60}$	51.14$_{\pm1.83}$	34.14$_{\pm0.30}$	5.44	4	28.12

Table 6: Error rate (%) and Rank with NLP tasks in USB.

Dataset	IMDB		AG News		Amazon Review		Yahoo! Answer		Yelp Review		Friedman	Final	Mean
# Label	20	100	40	200	250	1000	500	2000	250	1000	rank	rank	error rate
Fully-Supervised	5.87$_{\pm0.01}$	5.84$_{\pm0.12}$	5.74$_{\pm0.30}$	5.64$_{\pm0.05}$	36.81$_{\pm0.05}$	36.88$_{\pm0.19}$	26.25$_{\pm1.07}$	25.55$_{\pm0.43}$	31.74$_{\pm0.23}$	32.70$_{\pm0.58}$	-	-	-
Supervised	20.63$_{\pm3.13}$	13.47$_{\pm0.55}$	15.01$_{\pm1.21}$	13.00$_{\pm1.00}$	51.74$_{\pm0.63}$	47.34$_{\pm0.66}$	37.10$_{\pm1.22}$	33.56$_{\pm0.08}$	50.27$_{\pm0.51}$	46.96$_{\pm0.42}$	-	-	-
Π-Model	49.02$_{\pm1.37}$	27.57$_{\pm15.85}$	46.84$_{\pm6.20}$	13.44$_{\pm0.76}$	73.53$_{\pm6.92}$	48.27$_{\pm0.48}$	41.37$_{\pm2.15}$	32.96$_{\pm0.16}$	73.35$_{\pm2.31}$	52.02$_{\pm1.48}$	11.80	12	45.84
Pseudo-Labeling	26.38$_{\pm4.04}$	21.38$_{\pm1.34}$	23.86$_{\pm7.63}$	12.29$_{\pm0.40}$	53.00$_{\pm1.48}$	46.49$_{\pm0.45}$	38.60$_{\pm1.09}$	33.44$_{\pm0.24}$	55.70$_{\pm0.95}$	47.72$_{\pm0.37}$	10.60	11	35.89
Mean Teacher	21.27$_{\pm3.72}$	14.11$_{\pm1.77}$	14.98$_{\pm1.10}$	13.23$_{\pm1.12}$	51.67$_{\pm0.45}$	47.51$_{\pm0.24}$	36.97$_{\pm1.02}$	33.43$_{\pm0.22}$	51.07$_{\pm1.44}$	46.61$_{\pm0.34}$	9.30	10	33.09
VAT	32.59$_{\pm4.69}$	14.42$_{\pm2.53}$	15.00$_{\pm1.12}$	11.59$_{\pm0.94}$	50.38$_{\pm0.83}$	46.04$_{\pm0.28}$	35.16$_{\pm0.74}$	31.53$_{\pm0.41}$	52.76$_{\pm0.87}$	45.53$_{\pm0.13}$	8.40	8	33.50
UDA	9.36$_{\pm1.26}$	8.33$_{\pm0.61}$	18.73$_{\pm2.68}$	12.34$_{\pm1.90}$	52.48$_{\pm1.20}$	45.51$_{\pm0.61}$	35.31$_{\pm0.43}$	32.01$_{\pm0.68}$	58.22$_{\pm0.40}$	42.18$_{\pm0.68}$	8.70	9	31.45
FixMatch	8.20$_{\pm0.29}$	**7.36$_{\pm0.07}$**	22.80$_{\pm5.18}$	11.43$_{\pm0.65}$	47.85$_{\pm1.22}$	43.73$_{\pm0.45}$	34.15$_{\pm0.94}$	30.76$_{\pm0.53}$	50.34$_{\pm0.40}$	41.99$_{\pm0.58}$	5.60	7	29.86
Dash	8.93$_{\pm1.27}$	7.97$_{\pm0.53}$	19.30$_{\pm6.73}$	11.20$_{\pm1.12}$	47.79$_{\pm1.03}$	43.52$_{\pm0.07}$	35.10$_{\pm1.36}$	30.51$_{\pm0.47}$	47.99$_{\pm1.05}$	41.59$_{\pm0.61}$	5.10	6	29.39
CoMatch	7.36$_{\pm0.26}$	7.41$_{\pm0.20}$	13.25$_{\pm1.31}$	11.61$_{\pm0.42}$	48.98$_{\pm1.20}$	44.37$_{\pm0.25}$	33.48$_{\pm0.67}$	**30.19$_{\pm0.22}$**	46.49$_{\pm1.42}$	41.11$_{\pm0.53}$	3.80	3	28.43
CRMatch	7.88$_{\pm0.24}$	7.68$_{\pm0.35}$	13.35$_{\pm1.06}$	11.36$_{\pm1.04}$	46.23$_{\pm0.85}$	43.69$_{\pm0.48}$	33.07$_{\pm0.68}$	30.62$_{\pm0.47}$	46.61$_{\pm1.02}$	41.80$_{\pm0.77}$	3.70	2	28.23
FlexMatch	7.35$_{\pm0.10}$	7.80$_{\pm0.24}$	16.90$_{\pm6.76}$	11.43$_{\pm0.91}$	**45.75$_{\pm1.21}$**	43.14$_{\pm0.82}$	35.81$_{\pm1.09}$	31.42$_{\pm0.41}$	46.37$_{\pm0.74}$	**40.86$_{\pm0.74}$**	4.10	5	28.68
AdaMatch	9.62$_{\pm1.26}$	7.81$_{\pm0.46}$	**12.92$_{\pm1.53}$**	**11.03$_{\pm0.62}$**	46.75$_{\pm1.23}$	43.50$_{\pm0.67}$	**32.97$_{\pm0.43}$**	30.82$_{\pm0.29}$	48.16$_{\pm0.80}$	41.71$_{\pm1.08}$	4.00	4	28.53
SimMatch	**7.24$_{\pm0.02}$**	7.44$_{\pm0.20}$	14.80$_{\pm0.57}$	11.12$_{\pm0.15}$	47.27$_{\pm1.73}$	**43.09$_{\pm0.50}$**	34.15$_{\pm0.91}$	30.64$_{\pm0.42}$	**46.40$_{\pm1.71}$**	41.24$_{\pm0.17}$	2.90	1	28.34

The results for the 14 SSL algorithms on the datasets from CV, NLP, and Audio are shown in Table 5, Table 6, and Table 7, respectively. We adopt the pre-trained Vision Transformers (ViT) [4, 34, 30, 71] instead of training ResNets [1] from scratch for CV tasks. For NLP, we adopt Bert [30]. Wav2Vec 2.0 [71] and HuBert [32] are used for Audio.

5.1 CV Results

The results are illustrated in Table 5. Thanks to the good initialization of representation on unlabeled data given by the pre-trained ViT, SSL algorithms, even without using thresholding techniques, often achieve much better performance than the previous performance shown in TorchSSL [21]. Among all the SSL algorithms, ReMixMatch [23] ranks at the first and outperforms other SSL algorithms, due to the usage of Mixup, Distribution Alignment, and rotation self-supervised loss. Its superiority is especially demonstrated in the evaluation of Semi-Aves, a long-tailed and fine-grained CV dataset that is more realistic. Notice that SSL algorithms with self-supervised feature loss generally perform well than other SSL algorithms, e.g., CRMatch [61] and SimMatch [47] rank second and fourth respectively. Adaptive thresholding algorithms also demonstrate their effectiveness, e.g., AdaMatch [62] and FlexMatch [21] rank at third and fifth respectively. While better results of the evaluated SSL algorithms are obtained on CIFAR-100, Euro-SAT, and Semi-Aves, we also observe that the performance is relatively lower on STL-10 and TissueMNIST. The reason for lower performance on STL-10 might result from the usage of the self-supervised pre-trained model [33], rather than the supervised pre-trained model is used in other settings. Since TissueMNIST is a medial-related dataset, the biased pseudo-labels might produce a destructive effect that impedes training and leads to bad performance. The de-biasing of pseudo-labels and safe semi-supervised learning would be interesting topics in future work, especially for medical applications of SSL algorithms.

5.2 NLP Results

The results of NLP tasks are demonstrated in Table 6. The overall ranking of SSL algorithms in NLP is similar to that in CV. However, the SSL algorithm that works well in NLP does not always

Table 7: Error rate (%) and Rank with Audio tasks in USB. Fully-supervised result is not reported for FSDNoisy18k due to the unknown labels of its unlabeled set.

Dataset	GTZAN		UrbanSound8k		Keyword Spotting		ESC-50		FSDnoisy	Friedman	Final	Mean
# Label	100	400	100	400	50	100	250	500	1,772	rank	rank	error rate
Fully-Supervised	$5.98_{\pm0.32}$	$5.98_{\pm0.32}$	$16.65_{\pm1.71}$	$16.61_{\pm1.71}$	$2.12_{\pm0.11}$	$2.25_{\pm0.02}$	$26.00_{\pm2.13}$	$26.00_{\pm2.13}$	-	-	-	-
Supervised	$52.16_{\pm1.83}$	$31.53_{\pm0.52}$	$40.42_{\pm1.00}$	$28.55_{\pm1.90}$	$6.80_{\pm1.16}$	$5.25_{\pm0.56}$	$51.58_{\pm1.12}$	$35.67_{\pm0.42}$	$35.20_{\pm1.50}$	-	-	-
Π-Model	$74.07_{\pm0.62}$	$33.18_{\pm3.64}$	$54.24_{\pm6.01}$	$25.89_{\pm1.51}$	$64.39_{\pm4.10}$	$25.48_{\pm4.94}$	$47.25_{\pm1.14}$	$36.00_{\pm1.62}$	$35.73_{\pm0.87}$	10.67	12	44.03
Pseudo-Labeling	$57.29_{\pm2.80}$	$33.93_{\pm0.69}$	$42.09_{\pm2.41}$	$27.00_{\pm1.34}$	$7.82_{\pm1.64}$	$5.16_{\pm0.14}$	$49.33_{\pm2.52}$	$35.58_{\pm1.05}$	$35.34_{\pm1.60}$	10.00	10	32.62
Mean Teacher	$51.40_{\pm3.48}$	$31.60_{\pm1.46}$	$41.70_{\pm3.39}$	$28.91_{\pm0.93}$	$5.95_{\pm0.44}$	$5.39_{\pm0.42}$	$50.25_{\pm1.95}$	$37.33_{\pm1.20}$	$35.83_{\pm1.22}$	10.33	11	32.04
VAT	$79.51_{\pm1.99}$	$35.38_{\pm7.80}$	$49.62_{\pm2.42}$	$27.68_{\pm1.39}$	$\mathbf{2.18_{\pm0.08}}$	$\mathbf{2.23_{\pm0.08}}$	$46.42_{\pm1.90}$	$36.92_{\pm2.25}$	$32.07_{\pm1.05}$	8.33	9	34.67
UDA	$46.56_{\pm8.69}$	$23.62_{\pm0.63}$	$37.28_{\pm3.17}$	$20.27_{\pm1.58}$	$2.52_{\pm0.15}$	$2.62_{\pm0.10}$	$42.75_{\pm0.89}$	$33.50_{\pm1.95}$	$30.80_{\pm0.47}$	6.33	7	26.66
FixMatch	$36.04_{\pm4.57}$	$22.09_{\pm0.65}$	$36.12_{\pm4.26}$	$21.43_{\pm2.88}$	$4.84_{\pm3.57}$	$2.38_{\pm0.03}$	$\mathbf{37.75_{\pm3.19}}$	$30.67_{\pm1.05}$	$30.31_{\pm1.08}$	4.00	3	24.63
Dash	$47.00_{\pm3.65}$	$23.42_{\pm0.83}$	$42.02_{\pm5.02}$	$22.26_{\pm0.89}$	$5.70_{\pm4.40}$	$2.52_{\pm0.16}$	$48.17_{\pm1.16}$	$32.75_{\pm2.27}$	$33.19_{\pm0.95}$	7.56	8	28.56
CoMatch	$36.93_{\pm1.23}$	$22.20_{\pm1.39}$	$\mathbf{30.59_{\pm2.45}}$	$21.35_{\pm1.49}$	$11.39_{\pm0.85}$	$9.44_{\pm1.52}$	$40.17_{\pm2.08}$	$\mathbf{29.83_{\pm1.31}}$	$27.63_{\pm1.35}$	5.11	6	25.50
CRMatch	$40.58_{\pm3.97}$	$22.64_{\pm1.22}$	$39.47_{\pm4.66}$	$20.11_{\pm2.63}$	$2.40_{\pm0.13}$	$2.49_{\pm0.08}$	$42.67_{\pm0.51}$	$33.58_{\pm1.93}$	$30.45_{\pm1.52}$	5.00	5	26.04
FlexMatch	$34.60_{\pm4.07}$	$21.82_{\pm1.17}$	$40.18_{\pm2.73}$	$22.82_{\pm3.10}$	$2.42_{\pm0.08}$	$2.57_{\pm0.25}$	$39.58_{\pm0.59}$	$29.92_{\pm1.85}$	$\mathbf{26.36_{\pm0.55}}$	4.11	4	24.47
AdaMatch	$\mathbf{31.38_{\pm0.41}}$	$\mathbf{20.73_{\pm0.67}}$	$35.76_{\pm6.39}$	$21.15_{\pm1.22}$	$2.49_{\pm0.08}$	$2.49_{\pm0.10}$	$39.17_{\pm1.74}$	$31.33_{\pm1.23}$	$27.95_{\pm0.74}$	2.89	1	23.61
SimMatch	$32.42_{\pm2.18}$	$20.80_{\pm0.77}$	$31.70_{\pm6.05}$	$\mathbf{19.55_{\pm1.89}}$	$2.57_{\pm0.08}$	$2.53_{\pm0.22}$	$39.92_{\pm2.35}$	$32.83_{\pm1.43}$	$28.16_{\pm0.87}$	3.67	2	23.39

guarantee good performance in CV, which shows that the performance of SSL algorithms will be affected largely by data domains. For example, SimMatch which ranks first in NLP does not have the best performance in CV tasks (ranks fourth). The ranking of CoMatch is also increased in NLP, compared to that in CV. A possible reason is the different pre-training in backbones. For BERT, a masked language modeling objective is used during pre-training [30], thus the self-supervised feature loss might further improve the representation during fine-tuning with SSL algorithms. We observe that adaptive thresholding methods, such as FlexMatch and AdaMatch, consistently achieve good performance on both CV and NLP, even without self-supervised loss. Note that we do not evaluate MixMatch and ReMixMatch on NLP and Audio tasks because we find that mixing sentences with different lengths harms the model's performance.

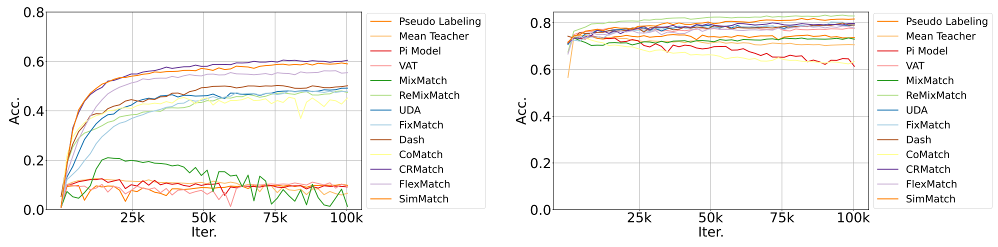
5.3 Audio Results

The results of Audio tasks are shown in Table 7. AdaMatch outperforms other algorithms in Audio tasks, while SimMatch demonstrates a similar performance to AdaMatch. An interesting finding is that CRMatch performs well on CV and NLP tasks, but badly in Audio tasks. We hypothesize that this is partially due to the noisy nature of the raw data in audio tasks. Except for Keyword Spotting, the gap between the performance of fully-supervised learning and that of SSL algorithms in Audio tasks is larger than in CV and NLP tasks. The reason behind this is probably that we exploit models that take waveform as input, rather than Mel spectrogram. Raw waveform might contain more noisy information that would be harmful to semi-supervised training. We identify exploring audio models based on Mel spectrogram as one of the future directions of USB.

5.4 Discussion

The evaluation results of SSL algorithms using USB are generally consistent with the results reported by previous work [22, 28, 23, 29, 20, 21]. However, using USB, we still provide some distinct quantitative and qualitative analysis to inspire the community. This section aims to answer the following questions: (1) Why should we evaluate an SSL algorithm on diverse tasks across domains? (2) Which option is better in the SSL scenario, training from scratch or using pre-training? (3) Does SSL consistently guarantee the performance improvement when using the state-of-the-art neural models as the backbones?

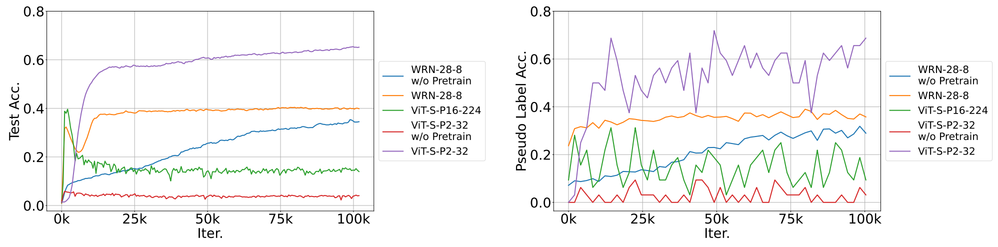
Performance Comparisons Table 8 shows the performance comparison of SSL algorithms in CV, NLP and Audio tasks. Although the ranking of each SSL algorithm in each domain is roughly close, the differences between ranks of SSL algorithms in different domains cannot be ignored. For example, FixMatch, CoMatch and CrMatch show large difference ($Rank_{max} - Rank_{min} \geq 4$) on the ranks across domains, which indicates that NLP and Audio tasks may have different characteristics compared with CV tasks that are more amenable to certain types of SSL algorithms compared with others. From the task perspective, it is important to consider such characteristics for guiding the choice of SSL methods. From the benchmarking perspective, it is useful to introduce diverse tasks from multiple domains when evaluating an SSL algorithm.



(a) WRN-28-8 from scratch.

(b) Pre-trained ViT-S-P2-32.

Figure 1: Comparison of test accuracy of SSL algorithms on CIFAR-100 with 400 labels. (a) Existing protocol which trains WRN-28-8 from scratch; (b) USB CV protocol which trains ImageNet-1K pre-trained ViT-S-P2-32, where S denotes small, P denotes patch size, and 32 is input image size.



(a) Test accuracy.

(b) Pseudo-label accuracy.

Figure 2: Pre-training ablation on CIFAR-400 with 400 labels. Test and pseudo-label accuracy are compared with WRN-28-8 without pre-training, pre-trained WRN-28-8, pre-trained ViT-S-P16-224, ViT-S-P2-32 without pre-training, and pre-trained ViT-S-P2-32.

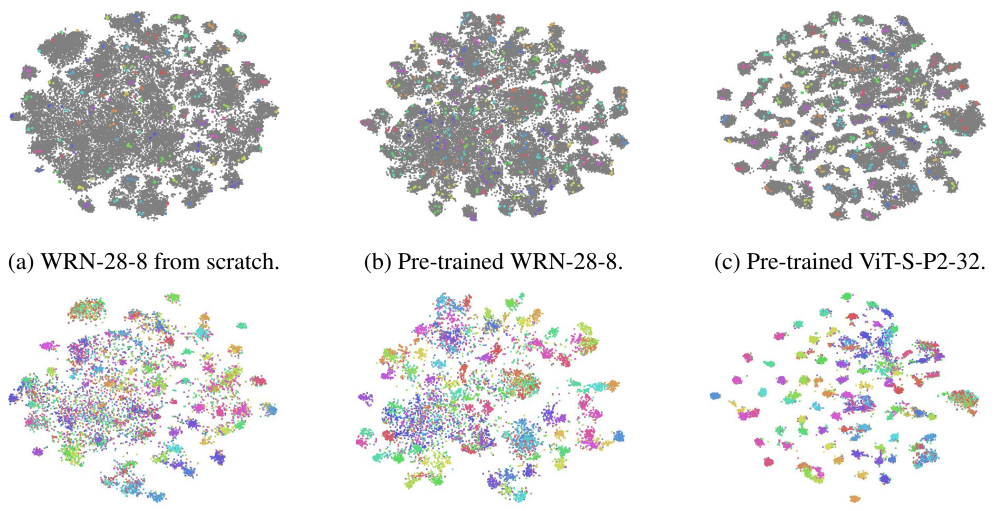
Table 8: Final ranks of SSL algorithms. Note that the rank for CV tasks here is different from the ones in Table 5 because we ignore MixMatch and ReMixMatch here to remove the effects of their missing ranks in NLP and Audio.

	Π-Model	Pseudo-Labeling	Mean Teacher	VAT	UDA	FixMatch	Dash	CoMatch	CRMatch	FlexMatch	AdaMatch	SimMatch
CV	10	9	12	11	6	5	8	7	1	4	2	3
NLP	12	11	10	8	9	7	6	3	2	5	4	1
Audio	12	10	11	9	7	3	8	6	5	4	1	2
$\text{Rank}_{max} - \text{Rank}_{min}$	2	2	2	3	3	**4**	2	**4**	**4**	1	3	2

Table 9: This table shows how many times an SSL algorithm is worse than supervised training, where the numbers of total settings are 9, 10, and 9 for CV, NLP, and Audio respectively.

	Π-Model	Pseudo-Labeling	Mean Teacher	VAT	MixMatch	ReMixMatch	UDA	FixMatch	Dash	CoMatch	CRMatch	FlexMatch	AdaMatch	SimMatch
CV	2	1	3	1	4	0	0	0	0	2	0	0	0	0
NLP	9	7	5	3	-	-	2	1	1	0	0	1	0	0
Audio	7	5	6	4	-	-	0	0	1	2	0	0	0	0

Effectiveness of Pre-training As shown in Figure 1a and Figure 1b, benefiting from the pre-trained ViT, the training becomes more efficient, and most SSL algorithms achieve higher optimal performance. Note that Pseudo Labeling, Mean Teacher, Π model, VAT, and MixMatch barely converge if training WRN-28-8 from scratch. A possible reason is that the scarce labeled data cannot provide enough supervision for unlabeled data to form correct clusters. However, these methods can achieve sufficiently reasonable results when using pre-trained ViT. As illustrated in Figure 2, using ViT without pre-training performs the worst among different backbones. The reason can be that ViT is data hungry if trained from scratch [34, 72, 73]. However, after appropriate pre-training, ViT performs the best among all the backbones. In addition, we provide the T-SNE visualization of the features in Figure 3, where the pretrained ViT model demonstrates the most separable feature space after training. In a word, pre-trained ViT makes the training more efficient and improves the generalization performance of SSL algorithms. For NLP tasks, we observe similar results, yet the improvement can be relatively less significant since pre-training is the de-facto fashion in the field.



(a) WRN-28-8 from scratch. (b) Pre-trained WRN-28-8. (c) Pre-trained ViT-S-P2-32.

(d) WRN-28-8 from scratch. (e) Pre-trained WRN-28-8. (f) Pre-trained ViT-S-P2-32.

Figure 3: T-SNE visualization of FixMatch features on training data (first row) and testing data (second row) of CIFAR-100 (400 labels). Different colors refer to labeled data with different classes while unlabeled data is indicated by gray color.

Robustness SSL sometimes hurts the generalization performance due to the large differences between the number of labeled data and the number of unlabeled data as shown in Table 9. We refer to an SSL algorithm as a robust SSL algorithm if it is consistently better than the supervised training setting. SSL algorithms cannot always outperform supervised training especially when labeled data is scarce. We find that CRMatch, AdaMatch and SimMatch are relatively robust SSL algorithms in USB. Although previous work has done some research towards robust SSL when using support vector machine [74, 75], we hope that our finding can serve as the motivation to delve into deep learning based robust SSL methods.

6 Codebase Structure of USB

In this section, we provide an overview of the codebase structure of USB, where four abstract layers are adopted. The layers include the core layer, algorithm layer, extension layer, and API layer in the bottom up direction as shown in Fig. 4.

Core Layer. In the core layer, we implement the commonly used core functions for training SSL algorithms. Besides, the code regarding datasets, data loaders, and models used in USB is also provided in the core layer. For flexible training, we implement common training hooks similar to MMCV [76], which can be modified and extended in the upper layers.

Algorithm Layer. In the algorithm layer, we first implement the base class for SSL algorithms, where we initialize the datasets, data loaders, and models from the core layer. Instead of implementing SSL algorithms independently as in TorchSSL [21], we further abstract the SSL algorithms, enabling better code reuse and making it easier to implement new algorithms. Except for the standalone implementation of loss functions used in SSL algorithms and algorithm-specific configurations, we further provide algorithm hooks according to the algorithm components summarized in Table 4. The algorithm hooks not only highlight the common part of different algorithms but also allows for a very easy and flexible combination of different components to resemble a new algorithm or conduct an ablation study. Based on this, we support 14 core SSL algorithms in USB, with two extra supervised learning variants. More algorithms are expected to be added through continued extension of USB.

Extension Layer. The extension layer is where we further extend the core SSL algorithms to different applications. Continuted effort are made on the extension of core SSL algorithms to imbalanced SSL algorithms [77, 78, 79, 80, 81, 82, 83, 84] and open-set SSL algorithms [85, 86, 87, 88, 89].

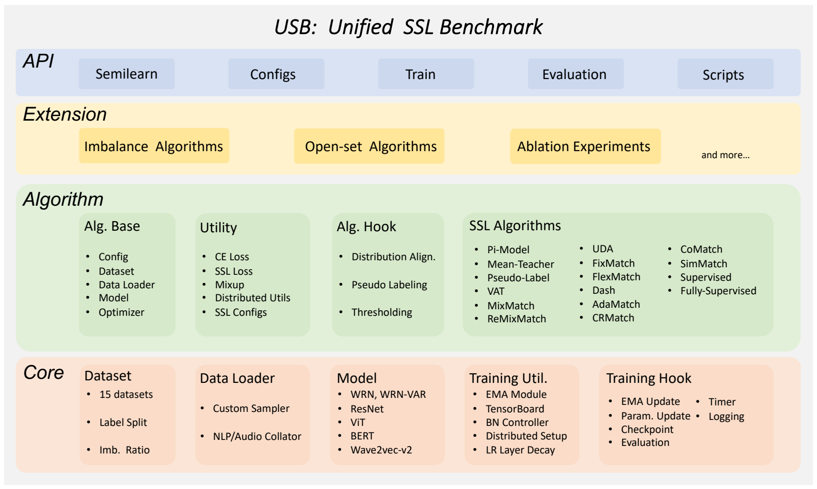


Figure 4: Structure of USB Codebase, consisting of 4 layers. The core layer provides the common functions, datasets, and models for SSL algorithms. The algorithm layer mainly implements the related SSL algorithms, with a high abstract level of algorithm components. Upon the algorithm layer, we use an extension layer for easy and flexible extension of core SSL algorithms. The top API layer supports a public python package SEMILEARN: `pip install semilearn`.

Systematic ablation study can also be conducted in the extension layer by inheriting either the core components and algorithms from the core layer or the algorithm layer.

API Layer. We wrap the core functions and algorithms in USB in the API layer as a public python package SEMILEARN. SEMILEARN is friendly for users from different backgrounds who want to employ SSL algorithms in new applications. Training and inference can be done in only a few lines of code with SEMILEARN. In addition, we provide the configuration files of all algorithms supported in USB with detailed parameter settings, which allows for reproduction of the results present in USB.

7 Limitation

Our primary focus is on semi-supervised classification in this paper. However, there are other SSL tasks that the SSL community should not ignore. USB currently does not include SSL tasks such as imbalanced semi-supervised learning [77, 79, 80, 81, 82, 83, 84], open-set semi-supervised learning [85, 86, 87, 88, 89], semi-supervised sequence modeling [90, 91, 92, 93, 26, 94], semi-supervised text generation [95, 96, 97], semi-supervised regression [98, 99, 100, 101, 102], semi-supervised object detection [103, 104, 105, 106, 107, 108], semi-supervised clustering [109, 110, 111, 112], etc. In addition, we do not implement generative adversarial networks based SSL algorithms [113, 64, 114, 65] and graph neural network based SSL algorithms [7, 115, 116, 117, 118] in USB, which are also important to the SSL community. Moreover, it is of great importance to extend current SSL to distributional shift settings, such as domain adaptation [119, 120] and out-of-distribution generalization [121], as well as time series anaysis [122]. We plan to evolve the benchmark in the future iterations over time by extending with more tasks.

8 Conclusion

We constructed USB, a unified SSL benchmark for classification that aims to enable consistent evaluation over multiple datasets from multiple domains and reduce the training cost to make the evaluation of SSL more affordable. With USB, we evaluate 14 SSL algorithms on 15 tasks across domains. We find that (1) although the performance of SSL algorithms is roughly close across domains, introducing diverse tasks from multiple domains is still necessary in the SSL scenario because the performance of SSL algorithms are not exactly steady across domains; (2) pre-training techniques can be helpful in the SSL scenario because it can not only accelerate the training but also improve the generalization performance; (3) unlabeled data sometimes hurts the performance especially when labeled data is extremely scarce. USB is a project for open extension and we plan to extend USB with more challenging tasks other than classification and introduce new algorithms.

Acknowledgments

We would like to thank the anonymous reviewers for their insightful comments and suggestions to help improve the paper. The computing resources of this study were mainly supported by Microsoft Asia and partially supported by High-Flyer AI.

References

[1] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[2] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[3] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[5] Dong Yu and Li Deng. *Automatic speech recognition*, volume 1. Springer, 2016.

[6] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *InterSpeech*, 2020.

[7] Vikas Verma, Meng Qu, Kenji Kawaguchi, Alex Lamb, Yoshua Bengio, Juho Kannala, and Jian Tang. Graphmix: Improved training of gnns for semi-supervised learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 10024–10032, 2021.

[8] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015.

[9] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*, 2019.

[10] YCAP Reddy, P Viswanath, and B Eswara Reddy. Semi-supervised learning: A brief review. *Int. J. Eng. Technol*, 7(1.8):81, 2018.

[11] Xiaojin Zhu. Semi-supervised learning literature survey. *world*, 10:10, 2005.

[12] Xiaojin Zhu and Andrew B Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

[13] Jesper E Van Engelen and Holger H Hoos. A survey on semi-supervised learning. *Machine Learning*, 109(2):373–440, 2020.

[14] Yassine Ouali, Céline Hudelot, and Myriam Tami. An overview of deep semi-supervised learning. *arXiv preprint arXiv:2006.05278*, 2020.

[15] Guo-Jun Qi and Jiebo Luo. Small data challenges in big data era: A survey of recent progress on unsupervised and semi-supervised methods. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020.

[16] Xiaohua Zhai, Avital Oliver, Alexander Kolesnikov, and Lucas Beyer. S4l: Self-supervised semi-supervised learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1476–1485, 2019.

[17] Junnan Li, Richard Socher, and Steven CH Hoi. Dividemix: Learning with noisy labels as semi-supervised learning. *arXiv preprint arXiv:2002.07394*, 2020.

[18] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[19] Hieu Pham, Zihang Dai, Qizhe Xie, and Quoc V Le. Meta pseudo labels. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11557–11568, 2021.

[20] 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.

[21] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. *Advances in Neural Information Processing Systems*, 34, 2021.

[22] Avital Oliver, Augustus Odena, Colin A Raffel, Ekin Dogus Cubuk, and Ian Goodfellow. Realistic evaluation of deep semi-supervised learning algorithms. *Advances in neural information processing systems*, 31, 2018.

[23] 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*, 2019.

[24] Yi Xu, Lei Shang, Jinxing Ye, Qi Qian, Yu-Feng Li, Baigui Sun, Hao Li, and Rong Jin. Dash: Semi-supervised learning with dynamic thresholding. In *International Conference on Machine Learning*, pages 11525–11536. PMLR, 2021.

[25] Jiaao Chen, Zichao Yang, and Diyi Yang. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. In *ACL*, 2020.

[26] Murali Karthick Baskar, Shinji Watanabe, Ramon Astudillo, Takaaki Hori, Lukáš Burget, and Jan Černockỳ. Semi-supervised sequence-to-sequence asr using unpaired speech and text. *arXiv preprint arXiv:1905.01152*, 2019.

[27] Léo Cances, Etienne Labbé, and Thomas Pellegrini. Comparison of semi-supervised deep learning algorithms for audio classification. *EURASIP Journal on Audio, Speech, and Music Processing*, 2022(1):1–16, 2022.

[28] 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, 2019.

[29] Qizhe Xie, Zihang Dai, Eduard Hovy, Thang Luong, and Quoc Le. Unsupervised data augmentation for consistency training. *Advances in Neural Information Processing Systems*, 33:6256–6268, 2020.

[30] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina N. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018.

[31] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

[32] Wei-Ning Hsu, Benjamin Bolte, Yao-Hung Hubert Tsai, Kushal Lakhotia, Ruslan Salakhutdinov, and Abdelrahman Mohamed. Hubert: Self-supervised speech representation learning by masked prediction of hidden units. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3451–3460, 2021.

[33] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[34] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[35] Antti Rasmus, Mathias Berglund, Mikko Honkala, Harri Valpola, and Tapani Raiko. Semi-supervised learning with ladder networks. *Advances in Neural Information Processing Systems*, 28:3546–3554, 2015.

[36] 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.

[37] 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.

[38] Ben Athiwaratkun, Marc Finzi, Pavel Izmailov, and Andrew Gordon Wilson. There are many consistent explanations of unlabeled data: Why you should average. In *International Conference on Learning Representations*, 2019.

[39] Alex Krizhevsky et al. Learning multiple layers of features from tiny images. 2009.

[40] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[41] Jiancheng Yang, Rui Shi, and Bingbing Ni. Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In *IEEE 18th International Symposium on Biomedical Imaging (ISBI)*, pages 191–195, 2021.

[42] Jiancheng Yang, Rui Shi, Donglai Wei, Zequan Liu, Lin Zhao, Bilian Ke, Hanspeter Pfister, and Bingbing Ni. Medmnist v2: A large-scale lightweight benchmark for 2d and 3d biomedical image classification. *arXiv preprint arXiv:2110.14795*, 2021.

[43] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 2019.

[44] Patrick Helber, Benjamin Bischke, Andreas Dengel, and Damian Borth. Introducing eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. In *IGARSS 2018-2018 IEEE International Geoscience and Remote Sensing Symposium*, pages 204–207. IEEE, 2018.

[45] Jong-Chyi Su and Subhransu Maji. The semi-supervised inaturalist-aves challenge at fgvc7 workshop. *arXiv preprint arXiv:2103.06937*, 2021.

[46] Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y Ng. Reading digits in natural images with unsupervised feature learning. 2011.

[47] Mingkai Zheng, Shan You, Lang Huang, Fei Wang, Chen Qian, and Chang Xu. Simmatch: Semi-supervised learning with similarity matching. *arXiv preprint arXiv:2203.06915*, 2022.

[48] Changchun Li, Ximing Li, and Jihong Ouyang. Semi-supervised text classification with balanced deep representation distributions. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5044–5053, 2021.

[49] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[50] Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

[51] Ming-Wei Chang, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar. Importance of semantic representation: Dataless classification. In *Aaai*, volume 2, pages 830–835, 2008.

[52] Julian McAuley and Jure Leskovec. Hidden factors and hidden topics: understanding rating dimensions with review text. In *Proceedings of the 7th ACM conference on Recommender systems*, pages 165–172, 2013.

[53] Yelp dataset: http://www.yelp.com/dataset_challenge.

[54] Justin Salamon, Christopher Jacoby, and Juan Pablo Bello. A dataset and taxonomy for urban sound research. In *Proceedings of the 22nd ACM international conference on Multimedia*, pages 1041–1044, 2014.

[55] Karol J. Piczak. ESC: Dataset for Environmental Sound Classification. In *Proceedings of the 23rd Annual ACM Conference on Multimedia*, pages 1015–1018. ACM Press.

[56] Eduardo Fonseca, Manoj Plakal, Daniel PW Ellis, Frederic Font, Xavier Favory, and Xavier Serra. Learning sound event classifiers from web audio with noisy labels. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 21–25. IEEE, 2019.

[57] Shu-wen Yang, Po-Han Chi, Yung-Sung Chuang, Cheng-I Jeff Lai, Kushal Lakhotia, Yist Y Lin, Andy T Liu, Jiatong Shi, Xuankai Chang, Guan-Ting Lin, et al. Superb: Speech processing universal performance benchmark. *arXiv preprint arXiv:2105.01051*, 2021.

[58] Gtzan dataset - music genre classification.

[59] 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, page 896, 2013.

[60] Junnan Li, Caiming Xiong, and Steven CH Hoi. Comatch: Semi-supervised learning with contrastive graph regularization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9475–9484, 2021.

[61] Yue Fan, Anna Kukleva, and Bernt Schiele. Revisiting consistency regularization for semi-supervised learning. In *DAGM German Conference on Pattern Recognition*, pages 63–78. Springer, 2021.

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

[63] Xiangli Yang, Zixing Song, Irwin King, and Zenglin Xu. A survey on deep semi-supervised learning. *arXiv preprint arXiv:2103.00550*, 2021.

[64] Jost Tobias Springenberg. Unsupervised and semi-supervised learning with categorical generative adversarial networks. *arXiv preprint arXiv:1511.06390*, 2015.

[65] Emily Denton, Sam Gross, and Rob Fergus. Semi-supervised learning with context-conditional generative adversarial networks. *arXiv preprint arXiv:1611.06430*, 2016.

[66] Zihang Dai, Zhilin Yang, Fan Yang, William W Cohen, and Russ R Salakhutdinov. Good semi-supervised learning that requires a bad gan. *Advances in neural information processing systems*, 30, 2017.

[67] Abhishek Kumar, Prasanna Sattigeri, and Tom Fletcher. Semi-supervised learning with gans: Manifold invariance with improved inference. *Advances in neural information processing systems*, 30, 2017.

[68] Olivier Chapelle, Bernhard Scholkopf, and Alexander Zien. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews]. *IEEE Transactions on Neural Networks*, 20(3):542–542, 2009.

[69] Milton Friedman. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association*, 32(200):675–701, 1937.

[70] Milton Friedman. A comparison of alternative tests of significance for the problem of m rankings. *The Annals of Mathematical Statistics*, 11(1):86–92, 1940.

[71] Alexei Baevski, Yuhao Zhou, Abdelrahman Mohamed, and Michael Auli. wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in Neural Information Processing Systems*, 33:12449–12460, 2020.

[72] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE transactions on pattern analysis and machine intelligence*, 2022.

[73] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International Conference on Machine Learning*, pages 10347–10357. PMLR, 2021.

[74] Yu-Feng Li and Zhi-Hua Zhou. Towards making unlabeled data never hurt. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(1):175–188, 2015.

[75] William S Noble. What is a support vector machine? *Nature biotechnology*, 24(12):1565–1567, 2006.

[76] MMCV Contributors. MMCV: OpenMMLab computer vision foundation. `https://github.com/open-mmlab/mmcv`, 2018.

[77] Jaehyung Kim, Youngbum Hur, Sejun Park, Eunho Yang, Sung Ju Hwang, and Jinwoo Shin. Distribution aligning refinery of pseudo-label for imbalanced semi-supervised learning. *Advances in Neural Information Processing Systems*, 33:14567–14579, 2020.

[78] Yidong Wang, Bowen Zhang, Wenxin Hou, Zhen Wu, Jindong Wang, and Takahiro Shinozaki. Margin calibration for long-tailed visual recognition. In *The 14th Asian Conference on Machine Learning*.

[79] Shoushan Li, Zhongqing Wang, Guodong Zhou, and Sophia Yat Mei Lee. Semi-supervised learning for imbalanced sentiment classification. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[80] Minsung Hyun, Jisoo Jeong, and Nojun Kwak. Class-imbalanced semi-supervised learning. *arXiv preprint arXiv:2002.06815*, 2020.

[81] Chen Wei, Kihyuk Sohn, Clayton Mellina, Alan Yuille, and Fan Yang. Crest: A class-rebalancing self-training framework for imbalanced semi-supervised learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10857–10866, 2021.

[82] Yuzhe Yang and Zhi Xu. Rethinking the value of labels for improving class-imbalanced learning. *Advances in Neural Information Processing Systems*, 33:19290–19301, 2020.

[83] Yue Fan, Dengxin Dai, and Bernt Schiele. Cossl: Co-learning of representation and classifier for imbalanced semi-supervised learning. *arXiv preprint arXiv:2112.04564*, 2021.

[84] Youngtaek Oh, Dong-Jin Kim, and In So Kweon. Distribution-aware semantics-oriented pseudo-label for imbalanced semi-supervised learning. *arXiv preprint arXiv:2106.05682*, 2021.

[85] Kuniaki Saito, Donghyun Kim, and Kate Saenko. Openmatch: Open-set consistency regularization for semi-supervised learning with outliers. *arXiv preprint arXiv:2105.14148*, 2021.

[86] Lan-Zhe Guo, Zhen-Yu Zhang, Yuan Jiang, Yu-Feng Li, and Zhi-Hua Zhou. Safe deep semi-supervised learning for unseen-class unlabeled data. In *International Conference on Machine Learning*, pages 3897–3906. PMLR, 2020.

[87] Qing Yu, Daiki Ikami, Go Irie, and Kiyoharu Aizawa. Multi-task curriculum framework for open-set semi-supervised learning. In *European Conference on Computer Vision*, pages 438–454. Springer, 2020.

[88] Huixiang Luo, Hao Cheng, Yuting Gao, Ke Li, Mengdan Zhang, Fanxu Meng, Xiaowei Guo, Feiyue Huang, and Xing Sun. On the consistency training for open-set semi-supervised learning. *arXiv preprint arXiv:2101.08237*, 3(6), 2021.

[89] Zhuo Huang, Chao Xue, Bo Han, Jian Yang, and Chen Gong. Universal semi-supervised learning. *Advances in Neural Information Processing Systems*, 34, 2021.

[90] Kevin Clark, Minh-Thang Luong, Christopher D Manning, and Quoc Le. Semi-supervised sequence modeling with cross-view training. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1914–1925, 2018.

[91] Luoxin Chen, Weitong Ruan, Xinyue Liu, and Jianhua Lu. Seqvat: Virtual adversarial training for semi-supervised sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8801–8811, 2020.

[92] Wei Li and Andrew McCallum. Semi-supervised sequence modeling with syntactic topic models. In *AAAI*, volume 5, pages 813–818, 2005.

[93] Andrew M Dai and Quoc V Le. Semi-supervised sequence learning. *Advances in neural information processing systems*, 28, 2015.

[94] Yidong Wang, Hao Wu, Ao Liu, Wenxin Hou, Zhen Wu, Jindong Wang, Takahiro Shinozaki, Manabu Okumura, and Yue Zhang. Exploiting unlabeled data for target-oriented opinion words extraction. In *Proceedings of the 29th International Conference on Computational Linguistics*, 2022.

[95] Ao Liu, An Wang, and Naoaki Okazaki. Semi-supervised formality style transfer with consistency training. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4689–4701, 2022.

[96] Junxian He, Jiatao Gu, Jiajun Shen, and Marc'Aurelio Ranzato. Revisiting self-training for neural sequence generation. In *International Conference on Learning Representations*, 2019.

[97] Jiaao Chen and Diyi Yang. Simple conversational data augmentation for semi-supervised abstractive dialogue summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6605–6616, 2021.

[98] Larry Wasserman and John Lafferty. Statistical analysis of semi-supervised regression. *Advances in Neural Information Processing Systems*, 20, 2007.

[99] Neal Jean, Sang Michael Xie, and Stefano Ermon. Semi-supervised deep kernel learning: Regression with unlabeled data by minimizing predictive variance. *Advances in Neural Information Processing Systems*, 31, 2018.

[100] Georgios Kostopoulos, Stamatis Karlos, Sotiris Kotsiantis, and Omiros Ragos. Semi-supervised regression: A recent review. *Journal of Intelligent & Fuzzy Systems*, 35(2):1483–1500, 2018.

[101] Zhi-Hua Zhou, Ming Li, et al. Semi-supervised regression with co-training. In *IJCAI*, volume 5, pages 908–913, 2005.

[102] Yu-Feng Li, Han-Wen Zha, and Zhi-Hua Zhou. Learning safe prediction for semi-supervised regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 31, 2017.

[103] Yuxing Tang, Josiah Wang, Boyang Gao, Emmanuel Dellandréa, Robert Gaizauskas, and Liming Chen. Large scale semi-supervised object detection using visual and semantic knowledge transfer. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2119–2128, 2016.

[104] Peng Tang, Chetan Ramaiah, Yan Wang, Ran Xu, and Caiming Xiong. Proposal learning for semi-supervised object detection. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 2291–2301, 2021.

[105] Mengde Xu, Zheng Zhang, Han Hu, Jianfeng Wang, Lijuan Wang, Fangyun Wei, Xiang Bai, and Zicheng Liu. End-to-end semi-supervised object detection with soft teacher. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3060–3069, 2021.

[106] Yihe Tang, Weifeng Chen, Yijun Luo, and Yuting Zhang. Humble teachers teach better students for semi-supervised object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3132–3141, 2021.

[107] Jiyang Gao, Jiang Wang, Shengyang Dai, Li-Jia Li, and Ram Nevatia. Note-rcnn: Noise tolerant ensemble rcnn for semi-supervised object detection. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9508–9517, 2019.

[108] Yen-Cheng Liu, Chih-Yao Ma, Zijian He, Chia-Wen Kuo, Kan Chen, Peizhao Zhang, Bichen Wu, Zsolt Kira, and Peter Vajda. Unbiased teacher for semi-supervised object detection. In *International Conference on Learning Representations*, 2020.

[109] Sugato Basu, Arindam Banerjee, and Raymond Mooney. Semi-supervised clustering by seeding. In *In Proceedings of 19th International Conference on Machine Learning (ICML-2002*. Citeseer, 2002.

[110] Eric Bair. Semi-supervised clustering methods. *Wiley Interdisciplinary Reviews: Computational Statistics*, 5(5):349–361, 2013.

[111] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Unsupervised and semi-supervised clustering: a brief survey. *A review of machine learning techniques for processing multimedia content*, 1:9–16, 2004.

[112] Sugato Basu, Mikhail Bilenko, and Raymond J Mooney. A probabilistic framework for semi-supervised clustering. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 59–68, 2004.

[113] Durk P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. Semi-supervised learning with deep generative models. *Advances in neural information processing systems*, 27, 2014.

[114] Augustus Odena. Semi-supervised learning with generative adversarial networks. *arXiv preprint arXiv:1606.01583*, 2016.

[115] Wenzheng Feng, Jie Zhang, Yuxiao Dong, Yu Han, Huanbo Luan, Qian Xu, Qiang Yang, Evgeny Kharlamov, and Jie Tang. Graph random neural networks for semi-supervised learning on graphs. *Advances in neural information processing systems*, 33:22092–22103, 2020.

[116] Aravind Sankar, Xinyang Zhang, and Kevin Chen-Chuan Chang. Meta-gnn: Metagraph neural network for semi-supervised learning in attributed heterogeneous information networks. In *Proceedings of the 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining*, pages 137–144, 2019.

[117] Maoguo Gong, Hui Zhou, AK Qin, Wenfeng Liu, and Zhongying Zhao. Self-paced co-training of graph neural networks for semi-supervised node classification. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

[118] Xujiang Zhao, Feng Chen, Shu Hu, and Jin-Hee Cho. Uncertainty aware semi-supervised learning on graph data. *Advances in Neural Information Processing Systems*, 33:12827–12836, 2020.

[119] Jindong Wang, Wenjie Feng, Yiqiang Chen, Han Yu, Meiyu Huang, and Philip S Yu. Visual domain adaptation with manifold embedded distribution alignment. In *Proceedings of the 26th ACM international conference on Multimedia*, pages 402–410, 2018.

[120] Jindong Wang, Yiqiang Chen, Shuji Hao, Wenjie Feng, and Zhiqi Shen. Balanced distribution adaptation for transfer learning. In *2017 IEEE international conference on data mining (ICDM)*, pages 1129–1134. IEEE, 2017.

[121] Jindong Wang, Cuiling Lan, Chang Liu, Yidong Ouyang, Tao Qin, Wang Lu, Yiqiang Chen, Wenjun Zeng, and Philip Yu. Generalizing to unseen domains: A survey on domain generalization. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[122] Yuntao Du, Jindong Wang, Wenjie Feng, Sinno Pan, Tao Qin, Renjun Xu, and Chongjun Wang. Adarnn: Adaptive learning and forecasting of time series. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 402–411, 2021.

[123] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2021.

[124] Xiaohua Zhai, Joan Puigcerver, Alexander Kolesnikov, Pierre Ruyssen, Carlos Riquelme, Mario Lucic, Josip Djolonga, Andre Susano Pinto, Maxim Neumann, Alexey Dosovitskiy, et al. A large-scale study of representation learning with the visual task adaptation benchmark. *arXiv preprint arXiv:1910.04867*, 2019.

[125] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018.

[126] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 702–703, 2020.

[127] Yizhou Wang, Shixiang Tang, Feng Zhu, Lei Bai, Rui Zhao, Donglian Qi, and Wanli Ouyang. Revisiting the transferability of supervised pretraining: an mlp perspective. *arXiv preprint arXiv:2112.00496*, 2021.

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

A Details of Datasets and in TorchSSL

We provide the details of datasets of TorchSSL in Table 10.

Table 10: Details of CV datasets and #labels used in TorchSSL. #Label per class represents the number of chosen labeled data per class from the training data. The test data is kept unchanged except for ImageNet where we use the validation dataset as the test dataset.

Dataset	#Label per class	#Training data	#Test data	#Class
CIFAR-10	4 / 25 / 100	50,000	10,000	10
CIFAR-100	4 / 25 / 100	50,000	10,000	100
SVHN	4 / 25 /100	604,388	26,032	10
STL-10	4 / 25 /100	100,000	10,000	10
ImageNet	100	1,281,167	50,000	1,000

B Correlation between TorchSSL and USB

Here we show the correlation between the mean error rates on TorchSSL and USB CV tasks. We take the 14 algorithms considered in the main paper and show their mean performance on TorchSSL versus that on USB CV tasks in Figure 5. Despite the fact that the Pearson correlation coefficient is 0.85, the final rank SSL algorithms is not consistent, which shows different adaptability of different methods when using pre-trained ViTs. For example, ReMixMatch shows the best mean performance on USB while AdaMatch and SimMatch have the best mean performance on TorchSSL. Please refer to Table 8 for more detailed rankings on CV, NLP, and Audio.

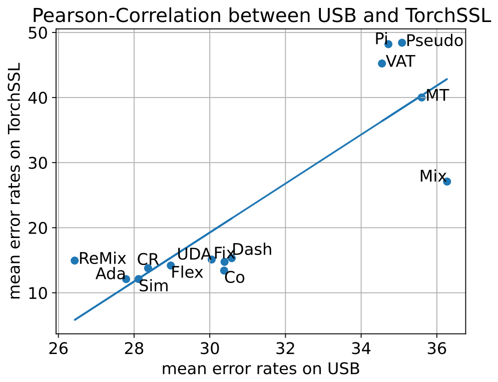


Figure 5: Correlation between TorchSSL and USB.

C Performance Results on ImageNet

Although we have excluded ImageNet from USB, we provide an evaluation on ImageNet of MAE pre-trained ViT-B, using UDA [29], FixMatch [20], FlexMatch [21], CoMatch [60], and SimMatch [47]. We train these algorithms using 10 labels per class and 100 labels per-class, i.e., a total of 10,000 labels and 100,000 labels respectively, corresponding to roughly 1% and 10% of the total labeled data in ImageNet. For learning rate and weight decay, we follow the fine-tuning protocol in MAE [33], where we use AdamW with a learning rate of 1e-3 and weight decay of 0.05. We use 16 A100 to train each algorithm and set the batch size to 256 for both labeled and unlabeled data. Other algorithmic hyper-parameters stay the same as their original implementations.

We present the results on ImageNet in Table 11. UDA and Fixmatch are near the bottom, similar to USB. SimMatch is still marked as one of the tops. Surprisingly, CoMatch does so well on ImageNet when it ranked only 9th on the USB benchmark. Also, while FlexMatch is the best on USB, it's pretty firmly behind CoMatch and SimMatch on ImageNet.

Table 11: ImageNet **accuracy** results. We use MAE pre-trained ViT-B.

Method	1w Labels	10w Labels	Rank
UDA	38.62	62.37	5
FixMatch	37.93	62.88	4
FlexMatch	39.13	63.09	3
CoMatch	44.32	65.80	2
SimMatch	46.48	67.61	1

Table 12: Swin-Transformer results on EuroSAT and Semi-AVES.

Dataset	EuroSAT		Semi-Aves
# Label	20	40	5,959
Supervised	$44.32_{\pm1.10}$	$34.40_{\pm1.44}$	$38.76_{\pm0.21}$
Fully-Supervised	$1.86_{\pm0.10}$		-
Π-Model	$42.49_{\pm3.21}$	$30.54_{\pm1.37}$	$38.74_{\pm0.60}$
Pseudo-Labeling	$42.49_{\pm3.21}$	$30.54_{\pm1.37}$	$38.74_{\pm0.60}$
Mean Teacher	$35.85_{\pm1.95}$	$19.62_{\pm3.28}$	$33.37_{\pm0.06}$
VAT	$40.63_{\pm2.68}$	$29.94_{\pm1.87}$	$35.84_{\pm0.36}$
UDA	$18.15_{\pm5.70}$	$12.09_{\pm1.26}$	$29.28_{\pm0.20}$
FixMatch	$17.19_{\pm3.46}$	$12.57_{\pm1.28}$	$28.88_{\pm0.22}$
Dash	$18.04_{\pm1.21}$	$12.98_{\pm1.27}$	$28.69_{\pm0.39}$
CoMatch	$13.65_{\pm1.42}$	$10.17_{\pm0.68}$	$37.71_{\pm0.31}$
CRMatch	$30.28_{\pm1.64}$	$22.39_{\pm1.41}$	$29.22_{\pm0.21}$
FlexMatch	$10.46_{\pm1.20}$	$9.06_{\pm1.80}$	$30.19_{\pm0.51}$
SimMatch	$11.19_{\pm1.01}$	$10.65_{\pm1.64}$	$28.55_{\pm0.13}$

D Results with Different Pre-trained Backbones

In this section, we verify USB with different pre-trained backbones. Different pre-trained backbones do affect the performance of SSL algorithms, which makes it important to report results with multiple backbones. We will continuously update results with different backbones at `https://github.com/microsoft/Semi-supervised-learning`. Here we report several results in Table 12, Table 13, and Table 14. Across the tasks, there is a pretty clear distinction between the performance of algorithms in the first half of the ranking list and the second half of the ranking list. While switching out backbones does not change the membership of these two halves, it does seem like the relative orderings within the top half can indeed vary a bit.

To compare different backbones on CV tasks, we fine-tune pre-trained public Swin-Transformer [123] with USB. We keep all hyper-parameters the same as in Table 15, and mainly evaluate on EuroSAT (32) and Semi-Aves (224). For EuroSAT, we change the input image size of the pre-trained Swin-S from 224 to 32, and the window size from 7 to 4 to accommodate the adapted input image size. For Semi-Aves, we adopt the original Swin-S. From the results in Table 12, one can observe, that on EuroSAT (32), as we adopt 224 pre-trained Swin-S and change its input and window size, the results are inferior to ViT-32 reported in the paper, whereas on Semi-Aves (224), the results are better than ViT-S. An interesting finding is that CoMatch performs relatively better with Swin-S while CrMatch performs worse. This also shows the importance of constantly updating the backbone in the future development of USB.

For NLP tasks, we additionally experiment with RoBERTa [31]. We train RoBERTa using the same hyper-parameters reported in Table 16. RoBerta generally performs better than Bert as expected. The performance difference is both very close when using RoBerta or Bert.

Due to the fact that the audio tasks setting in the current version of USB being built upon raw waveforms, there are not many pre-trained models available to use. We report the results of HuBert [32] and Wave2Vecv2.0 [71] for audio tasks to compare different backbones. The difference between these two backbones selected mainly lies in pre-training data. Wave2Vecv2.0 is pre-trained using raw human voice data and HuBert is an improved model with a discrete clustering target. Thus we can observe from the results, that on human voice tasks Superb-KS, Wave2Vecv2.0 has better performance, whereas, on other tasks, HuBert is more robust and outperforms Wave2Vecv2.0.

Table 13: RoBERTa results on Yelp.

Dataset	Yelp	
# Labels	250	1000
Supervised	42.56±1.15	39.00±0.16
Fully-Supervised	29.15±0.12	
Pseudo-Label	48.26±0.02	40.56±0.16
MeanTeacher	49.41±0.03	44.36±1.04
Π-Model	49.16±2.04	42.93±0.88
VAT	43.04±0.02	39.24±0.06
AdaMatch	38.24±0.02	35.64±0.06
UDA	40.13±0.15	38.98±0.03
FixMatch	39.82±0.95	37.42±0.30
FlexMatch	39.11±0.02	36.84±0.01
Dash	39.86±1.01	36.23±0.21
CRMatch	40.08±1.28	35.85±0.38
CoMatch	39.95±0.86	36.89±0.22
SimMatch	38.76±0.68	36.39±0.34

Table 14: HuBert results on keyword Spotting and Wave2Vec2.0 results on FSDnoisy.

Dataset	keyword Spotting		FSDnoisy
# Label	50	400	1,772
Supervised	8.95±1.62	6.31±0.46	33.54±1.65
Fully-Supervised	2.41±0.15		-
Π-Model	87.86±2.88	72.89±3.23	35.97±0.84
Pseudo-Labeling	25.59±2.88	13.02±2.47	35.23±0.78
Mean Teacher	89.79±0.30	90.01±0.02	40.13±1.70
VAT	2.27±0.07	2.43±0.02	34.21±0.31
UDA	11.76±0.06	2.23±0.16	33.09±1.03
FixMatch	11.63±0.24	8.93±2.04	33.09±0.64
Dash	11.88±0.15	8.25±4.22	33.02±1.39
CoMatch	15.96±1.02	10.34±1.52	30.24±0.55
CRMatch	5.85±1.19	3.66±0.33	30.48±0.65
FlexMatch	10.22±1.10	5.10±3.70	32.66±4.09
SimMatch	9.43±0.63	5.47±2.72	29.57±0.52

E Details of Datasets in USB

E.1 CV Tasks

CIFAR-100 The CIFAR-100 [39] dataset is a natural image (32×32 pixels) recognition dataset consisting 100 classes. There are 500 training samples and 100 test samples per class.

STL-10 The STL-10 [40] dataset is a natural color image (96×96 pixels) recognition dataset consisting 10 classes. Particularly, each class has 500 training samples and 800 test samples. Apart from the labeled samples, STL-10 also provides 100,000 unlabeled samples. Note that the unlabeled samples contain other classes in addition to the ones in the labeled data.

EuroSat EuroSAT [43, 44] dataset is based on Sentinel-2 satellite images covering 13 spectral bands and consisting of 10 classes with 27,000 labeled and geo-referenced samples. Following [124], we use the dataset with the optical R, G, B frequency bands, thus each image is of size $64 \times 64 \times 3$. We take the first 60% images from each class as training set; the next 20% as val set, and the last 20% as test set.

TissueMNIST TissueMNIST [41, 42] is a medical dataset of human kidney cortex cells, segmented from 3 reference tissue specimens and organized into 8 categories. The total 236,386 training samples are split with a ratio of $7:1:2$ into training (165,466 images), validation (23,640 images) and test set (47,280 images). Each gray-scale image is 28×28 pixels.

Semi-Aves Semi-Aves [45] is a dataset of Aves (birds) classification, where 5,959 images of 200 bird species are labeled and 26,640 images are unlabeled. As class distribution mismatch hurts the performance [85], we do not use out-of-class unlabeled data. This dataset is challenging as it is naturally imbalanced. The validation and test set contain 10 and 20 images respectively for each of the 200 categories in the labeled set.

E.2 NLP Tasks

IMDB The IMDB [49] dataset is a binary sentiment classification dataset. There are 25,000 reviews for training and 25,000 for test. IMDB is class balanced which means the positive and negative reviews have the same number both for training and test. For USB, we draw 12,500 samples and 1,000 samples per class from training samples to form the training dataset and validation dataset respectively. The test dataset is unchanged.

Amazon Review The Amazon Review [52] dataset is a sentiment classification dataset. There are 5 classes (scores). Each class (score) contains 600,000 training samples and 130,000 test samples. For USB, we draw 50,000 samples and 5,000 samples per class from training samples to form the training dataset and validation dataset respectively. The test dataset is unchanged.

Yelp Review The Yelp Review [53] sentiment classification dataset has 5 classes (scores). Each class (score) contains 130,000 training samples and 10,000 test samples. For USB, we draw 50,000 samples and 5,000 samples per class from training samples to form the training dataset and validation dataset respectively. The test dataset is unchanged.

AG News The AG News [50] dataset is a news topic classification dataset containing 4 classes. Each class contains 30,000 training samples and 1,900 test samples. For USB, we draw 25,000 samples and 2,500 samples per class from training samples to form the training dataset and validation dataset respectively. The test dataset is unchanged.

Yahoo! Answer The Yahoo! Answer [51] topic classification dataset has 10 categories. Each class contains 140,000 training samples and 6,000 test samples. For USB, we draw 50,000 samples and 5,000 samples per class from training samples to form the training dataset and validation dataset respectively. The test dataset is unchanged.

E.3 Audio Tasks

GTZAN The GTZAN dataset is collected for music genre classification of 10 classes and 100 audio recordings for each class. The maximum length of the recordings is 30 seconds and the original sampling rate is 22,100 Hz. We split 7,000 samples for training, 1,500 for validation, and 1,500 for testing. All recordings are re-sampled at 16,000 Hz.

UrbanSound8k The UrbanSound8k dataset [54] contains 8,732 labeled sound events of urban sounds of 10 classes, with the maximum length of 4 seconds. The original sampling rate of the audio recordings is 44,100 and we re-sample it to 16,000. It is originally divided into 10 folds, where we use the first 8 folds of 7,079 samples as training set, and the last two folds as validation set of size 816 and testing set of size 837 respectively.

FSDNoisy18k The FSDNoisy18 dataset [56] is a sound event classification dataset across 20 classes. It consists of a small amount of manually labeled data - 1,772 and a large amount of noisy data - 15,813 which is treated as unlabeled data in our paper. The original sample rate is 44,100 Hz, and the length of the recordings lies between 3 seconds and 30 seconds. We use the testing set provided for evaluation, which contains 947 samples.

Keyword Spotting (Superb-KS) The Keyword spotting dataset is one of the tasks in Superb [57] for classifying the keywords. It contains speech utterances of a maximum length of 1 second and the sampling rate of 16,000. The training, validation, and testing set contain 18,538; 2,577; 2,567 recordings, respectively. For pre-processing, we remove the silence and unknown labels from the dataset.

ESC-50 The ESC-50 [55] is a dataset containing 2,000 environmental audio recordings for 50 sound classes. The maximum length of the recordings is 5 seconds and the original sampling rate is 44,100. We split 1,200 samples as training data, 400 as validation data, and 400 as testing data. We also re-sample the audio recordings to 16,000 Hz during pre-processing.

F Details of Implemented SSL algorithms in USB

Π **model [35]** is a simple SSL algorithm that forces the output probability of perturbed versions of unlabeled data be the same. Π model uses Mean Squared Error (MSE) for optimization.

Pseudo Labeling [59] turns the output probability of unlabeled data into the 'one-hot' hard one and makes the same unlabeled data to learn the pseudo 'one-hot' label. Unlike Π model, Pseudo Labeling uses CE for optimization.

Mean Teacher [36] takes the exponential moving average (EMA) of the neural model as the teacher model. With Mean Teacher, the neural model forces itself to output a similar probability to the EMA teacher. Though the later SSL algorithms will not always choose the EMA model as the teacher, they often use the EMA model for validation/test cause it decreases the risk of neural models falling into the local optima.

VAT [37] enhances the robustness of the conditional predicted label distribution around each unlabeled data against an adversarial perturbation. In other words, VAT forces the neural model to give similar predictions on unlabeled data even facing a strong adversarial perturbation.

MixMatch [28] first introduces Mixup [125] into SSL by taking the input as the mixture of labeled and unlabeled data and the output as the mixture of labels and model predictions on unlabeled data. Note that MixMatch also utilizes MSE as the unsupervised loss.

ReMixMatch [23] can be seen as the upgraded version of MixMatch. ReMixMatch improves MixMatch by (1) proposing stronger augmentation (i.e., Control Theory Augmentation (CTAugment) [23]) for unlabeled data; (2) using Augmentation Anchoring to force the model to output similar predictions to weakly augmented unlabeled data when fed strongly augmented data; (3) utilizing Distribution Alignment to encourage the marginal distribution of predictions on unlabeled data to be similar to the marginal distribution of labeled data.

UDA [29] also introduces strong augmentation (i.e., RandAugment [126]) for unlabeled data. The core idea of UDA is similar to Augmentation Anchoring [23], which forces the predictions of neural models on the strongly-augmented unlabeled data to be close to those of weakly-augmented unlabeled data. Instead of turning predictions into hard 'one-hot' pseudo-labels, UDA sharpens the prediction on unlabeled data. Thresholding technique is used to mask out unconfident unlabeled samples that are considered noise here.

FixMatch [20] is the upgraded version of Pseudo Labeling. FixMatch turns the predictions on weakly-augmented unlabeled data into hard 'one-hot' pseudo-labels and then further uses them as the learning signal of strongly-augmented unlabeled data. FixMatch finds that using a high threshold (e.g., 0.95) to filter noisy unlabeled predictions and take the rest as the pseudo-label can achieve very good performance.

Dash [24] improves the FixMatch by using a gradually increased threshold instead of a fixed threshold, which allows more unlabeled data to participate in the training at the early stage. Moreover, Dash theoretically establishes the convergence rate from the view of non-convex optimization.

CoMatch [60] firstly introduces contrastive learning into SSL. Except for consistency regularizing on the class probabilities, it is also exploited on graph-based feature representations, which impose smooth constraints on pseudo-labels generated.

CRMatch [61] proposed an improved consistency regularization framework which impose consistency and equivariance on the classification probability and the feature level.

FlexMatch [21] firstly introduces the class-specific thresholds into SSL by considering the different learning difficulties of different classes. Specifically, the hard-to-learn classes should have a low threshold to speed up convergence while the easy-to-learn classes should have a high threshold to avoid confirmation bias.

AdaMatch [62] is proposed mainly for domain adaption, but can also adapted to SSL. It is characterized by Relative Threshold and Distribution Alignment, where the relative threshold is adaptively estimated from EMA of the confidence on labeled data.

SimMatch [47] extends CoMatch [60] by considering semantic-level and instance-level consistency regularization. Similar similarity relationship of different augmented versions on the same data with respect to other instances is encouraged during training. In addition, a memory buffer consisting of predictions on labeled data is adopted to connect the two-level regularization.

G Experiment Details in USB

G.1 Setup for CV Tasks in USB

Table 15: Hyper-parameters of CV tasks in USB.

Dataset	CIFAR-100	STL-10	Euro-SAT	TissueMNIST	Semi-Aves
Image Size	32	96	32	32	224
Model	ViT-S-P4-32	ViT-B-P16-96	ViT-S-P4-32	ViT-T-P4-32	ViT-S-P16-224
Weight Decay	5e-4				
Labeled Batch size	16				
Unlabeled Batch size	16				
Learning Rate	5e-4	1e-4	5e-5	5e-5	1e-3
Layer Decay Rate	0.5	0.95	1.0	0.95	0.65
Scheduler	$\eta = \eta_0 \cos(\frac{7\pi k}{16K})$				
Model EMA Momentum	0.0				
Prediction EMA Momentum	0.999				
Weak Augmentation	Random Crop, Random Horizontal Flip				
Strong Augmentation	RandAugment [126]				

For CV tasks in USB, we use ViT models [34]. We find that directly using released ViT models leads to overfitting and one needs to fix the image resolution as the pre-trained resolution, as demonstrated in Paragraph 5.4. Instead, we pre-train our own ViT models on ImageNet-1K [8]. To match the number of parameters as the CNN models used in the classic setting, we use ViT-Tiny and ViT-Small with a patch size of 2 and image size of 32 for TissueMNIST, CIFAR-100 and EuraSAT, respectively; ViT-Small with a patch size of 16 and image size of 224 for Semi-Aves. For better transfer performance, we adopt an MLP before the final classifier during pre-training, as in [127]. For supervised pre-training on ImageNet-1K, we use Lamb optimizer with a learning rate of 0.05, and a weight decay of 0.03 for ViT-Tiny and a weight decay of 0.05 for ViT-Small. We adopt a large batch size of 4096 and train the networks for 300 epochs, with a linear learning rate warmup for the first 20 epochs. After the warmup, cosine scheduler is utilized. For augmentation, we use RandAugment [126], along with Mixup [125] and CutMix [128]. We also use label smoothing of 0.1 during pre-training. Since STL10 is a subset of ImageNet, we adopt unsupervised pre-training MAE [33] of ViT-Small with image size of 96 to avoid cheating.

For USB CV tasks, we adopt layer-wise learning rate decay as in [123]. We tune the learning rate and layer decay rate on different datasets using FixMatch, and use the best configuration to train all SSL algorithms [7]. The cosine annealing scheduler is similar to the classic setting but with total steps of $204,800$ and a warm-up of $5,120$ steps. The labeled and unlabeled batch size is both set to 16. Other algorithm-related hyper-parameters stay the same as in the original papers.

G.2 Setup for NLP Tasks in USB

We use pre-trained BERT-Base [30] for all NLP tasks in USB. We set the batch size of labeled data and unlabeled data to 4 for reducing the training time and GPU memory requirement. To fine-tune the BERT-Base under USB, we adopt AdamW optimizer with weight decay of $1\mathrm{e}{-4}$. Similarly, we conduct a grid search of the learning rate and layer decay on different datasets using FixMatch and pick the best configuration to fine-tune other SSL algorithms. We utilize the same cosine learning rate

[7]We present the full tuning results in: `https://github.com/microsoft/Semi-supervised-learning`.

scheduler as in the classic setting with the total training steps of $102,400$ and a warm-up of $5,120$ steps. We use the fine-tuned model without parameter momentum to conduct evaluations. For all datasets, we cut the long sentence to satisfy the input length requirement of BERT-Base. For data augmentation, we adopt back-translation as the strong augmentation [29, 25]. Specifically, we use De-En and Ru-En translation with WMT19.

Table 16: Hyper-parameters of NLP tasks in USB.

Dataset	AG News	Yahoo! Answer	IMDb	Amazom-5	Yelp-5
Max Length	512				
Model	Bert-Base				
Weight Decay	1e-4				
Labeled Batch size	4				
Unlabeled Batch size	4				
Learning Rate	5e-5	1e-4	5e-5	1e-5	5e-5
Layer Decay Rate	0.65	0.65	0.75	0.75	0.75
Scheduler	$\eta = \eta_0 \cos(\frac{7\pi k}{16K})$				
Model EMA Momentum	0.0				
Prediction EMA Momentum	0.999				
Weak Augmentation	None				
Strong Augmentation	Back-Translation [29]				

G.3 Setup for Audio Tasks in USB

For Audio tasks, we adopt Wav2Vec 2.0 [71] and HuBert [32] as the pre-trained model. The batch size of labeled data and unlabeled data is set to 8. We keep the sampling rate of audios as $16,000$. We adopt AdamW optimizer with a weight decay of $5\text{e}{-4}$, and search the learning rate and layer decay as before. Other hyper-parameter settings are the same as NLP tasks. Mimicking RandAugment, for strong augmentation in audio tasks, we random sample 2 augmentations from the augmentation pool and random set the augmentation magnitude during training.

Table 17: Hyper-parameters of Audio tasks in USB.

Dataset	GTZAN	Keyword Spotting	UrbanSound8k	FSDNoisy	ESC-50
Sampling Rate	16,000				
Max Length	3.0	1.0	4.0	5.0	5.0
Model	Wav2Vecv2-Base		HuBERT-Base		
Weight Decay	5e-4				
Labeled Batch size	8				
Unlabeled Batch size	8				
Learning Rate	2e-5	5e-5	5e-5	5e-4	1e-4
Layer Decay Rate	1.0	0.75	0.75	0.75	0.85
Scheduler	$\eta = \eta_0 \cos(\frac{7\pi k}{16K})$				
Model EMA Momentum	0.0				
Prediction EMA Momentum	0.999				
Weak Augmentation	Random Sub-sample				
Strong Augmentation	Random Sub-sample, Random Gain, Random Pitch, Random Speed				