Appendix Contents

A. Code and Data Splits for Reproducibility

All code resources needed to reproduce our analysis can be found in our github repo

https://anonymous.4open.science/r/SSL-vs-SSL-benchmark-48B0/README.md

The exact splits of the TissueMNIST, PathMNIST, TMED-2, AIROGS we used will also be provided upon acceptance.

Our codebase builds upon the open-source PyTorch repo by Suzuki [61]. We added many additional algorithms (we added MixMatch, FixMatch, FlexMatch, and CoMatch, as well as all 7 self-supervised methods) and customized the experiments, especially providing a runtime-budgeted hyperparameter tuning strategy as outlined in App. D.

Suzuki's code was intended as a reimplementation in PyTorch of Oliver et al. [54]'s benchmark of semi-supervised learning (while Oliver et al's original repo was in Tensorflow, we prefer PyTorch).

In a way, this makes our repo a "cousin" of the codebase of Su et al. [60]'s fine-grained classification benchmark, because their github repo also credits Suzuki's repo as an ancestor.

B. Dataset Details

B.1. Dataset Selection

We selected PathMNIST and TissueMNIST from 12 candidate datasets in the MedMNIST collections [68, 69] by matching two criteria: (i) contains at least 5 imbalanced classes; (ii) can build a large unlabeled set (at least 50000 images). Prior experiments from dataset creator Yang et al. [69] suggest 28x28 resolution is a reasonable choice. They report that a larger resolution (224x224) does not yield much more accurate classifiers for these two datasets.

B.2. Class Description

TissueMNIST contains images of human kidney cortex cells. The dataset contains 8 classes. See [69] for more details.

Class ID	Abbreviation	Description
0	CD/CT	Collecting Duct, Connecting Tubule
1	DCT	Distal Convoluted Tubule
2	GE	Glomerular endothelial cells
3	IE	Interstitial endothelial cells
4	LEU	Leukocytes
5	POD	Podocytes
6	PT	Proximal Tubule Segments
7	TAL	Thick Ascending Limb

PathMNIST contains patches from colorectal cancer histology slides that comprise of 9 tissue types. See [42, 69] for more details.

Class ID	Abbreviation	Description
0	ADI	adipose
1	BACK	background
2	DEB	debris
3	LYM	lymphocytes
4	MUC	mucus
5	MUS	smooth muscle
6	NORM	normal colon mucosa
7	STR	cancer-associated stroma
8	TUM	colorectal adenocarcinoma epithelium

TMED-2 contains 2D grayscale images captured from routine echocardiogram scans. In this study, we adopt the view classification task from [36]. For more detail please see [36, 37]

Class ID	Abbreviation	Description
0	PLAX	parasternal long axis
1	PSAX	parasternal short axis
2	A2C	apical 2-chamber
3	A4C	apical 4-chamber

AIROGS is a dataset of color image of the retina. The binary classification task is to differentiate between referable glaucoma and no referable glaucoma [22].

Class ID	Abbreviation	Description
0	No Glauc.	no referable glaucoma
1	Glaucoma	referable glaucoma

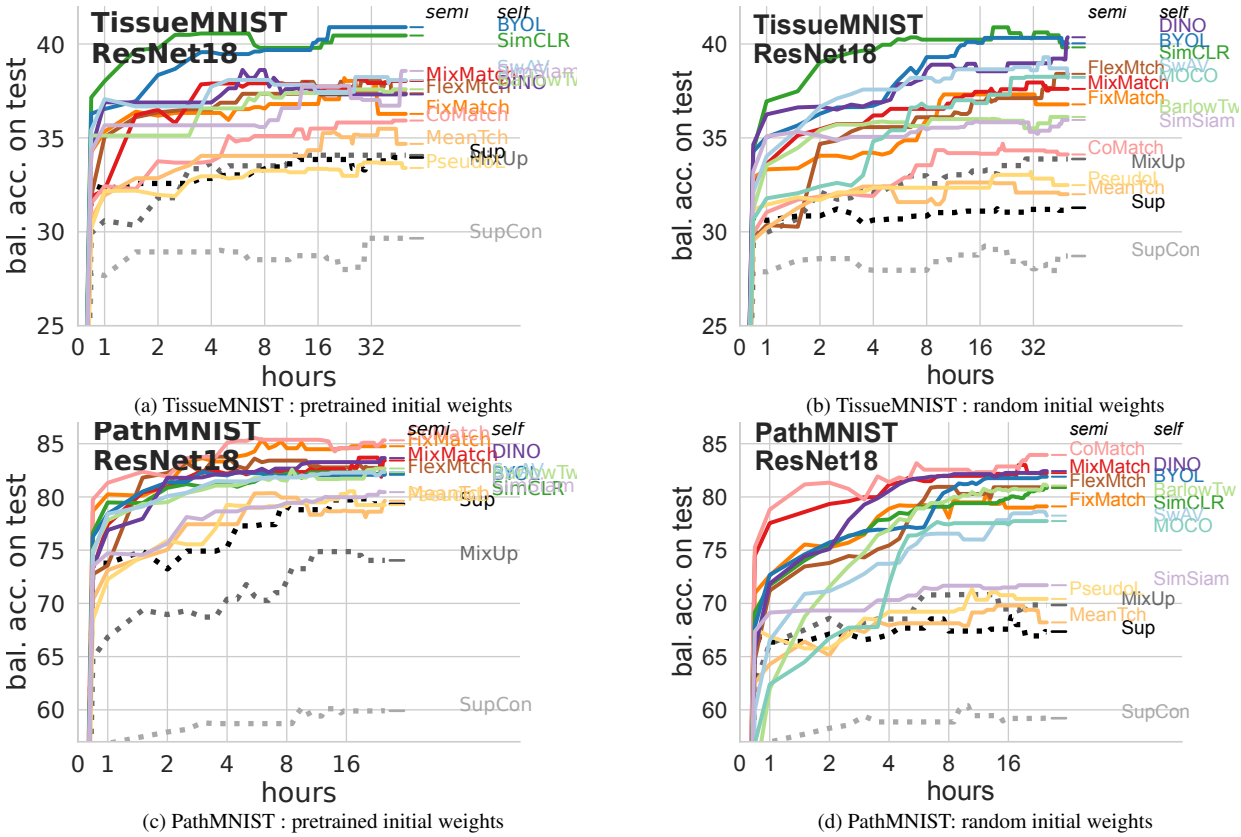
C. Additional Results

C.1. Impact of pretraining on accuracy-over-time profiles

To study the impact of pretraining, we compare the accuracy-over-time profiles of TissueMNIST and PathMNIST based on the two different initialization strategy.

Fig. C.1 left column shows pretraining on ImageNet strategy, right column shows random initialization. On TissueMNIST, SimCLR (green) and BYOL (blue) are the top two methods in both cases. On PathMNIST, semi-supervised methods seem better: FixMatch and CoMatch are best on the pretraining case, with MixMatch and Flexmatch only a few points of balanced accuracy lower. MixMatch and CoMatch are best in the random initialization case.

Across both plots, pretraining does not seem to impact the top-performing methods' ultimate accuracy by much. (e.g. by more than a few points of accuracy). However, with more limited time budgets (e.g. after only a few hours), we do see initialization from pretraining understandably tends to improve some methods. Pretraining time on a source dataset is NOT counted to the runtime reported in x-axis.



(a) TissueMNIST : pretrained initial weights

(b) TissueMNIST : random initial weights

(c) PathMNIST : pretrained initial weights

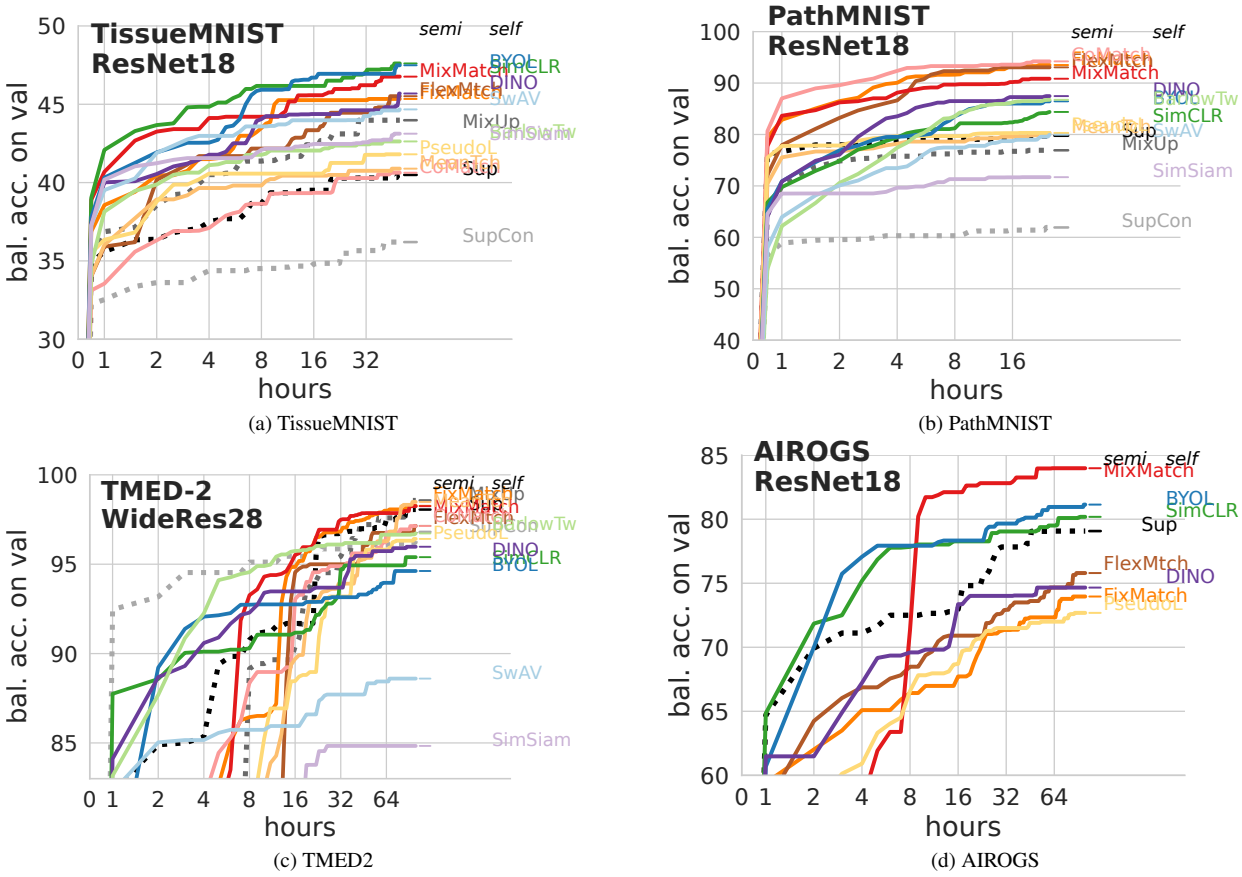
(d) PathMNIST: random initial weights

Figure C.1. Balanced accuracy on test set over time for semi- and self-supervised methods, **with (left) and without (right) initial weight pretraining on ImageNet.** Curves represent mean of each method at each time over 5 trials of Alg. D.1.

C.2. Comparing validation profiles of accuracy-over-time

Fig. C.2 shows profiles of accuracy over time on the validation set, in contrast to the test set performance shown in the main paper's Fig. 1.

All curves here by definition must be monotonically increasing, because our unified algorithm selects new checkpoints only when they improve the validation-set balanced accuracy metric. The important insight our work reveals is that the same model checkpoints selected here, based on validation-set accuracy, also tend to produce improved test-set accuracy over time (in Fig. 1). This helps provide empirical confidence in using ***realistically-sized*** validation sets.



(a) TissueMNIST

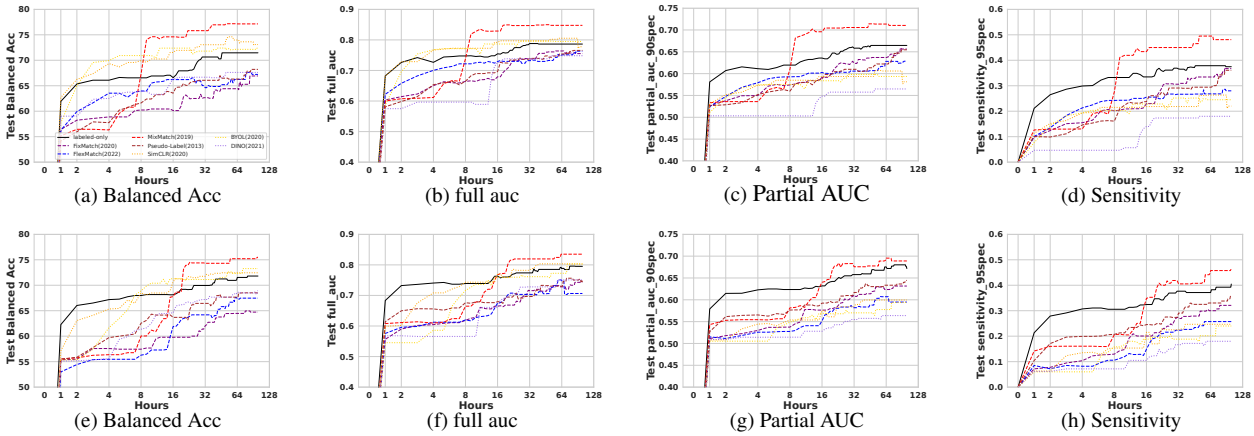
(b) PathMNIST

(c) TMED2

(d) AIROGS

Figure C.2. **VALIDATION** accuracy over time profiles of semi- and self-supervised methods on 4 datasets (panels a-d).All curves here by definition must be monotonically increasing. We see that the increasing validation set performance translate to increasing test set performance in Fig. 1.

C.3. Additional Evaluation Metrics



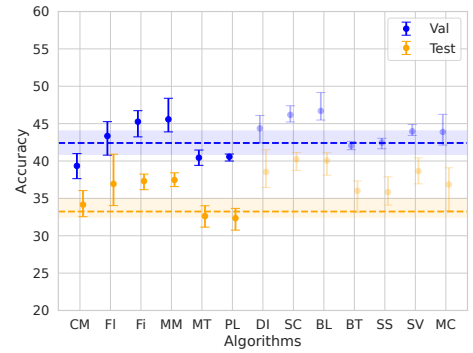
(a) Balanced Acc (b) full auc (c) Partial AUC (d) Sensitivity

(e) Balanced Acc (f) full auc (g) Partial AUC (h) Sensitivity

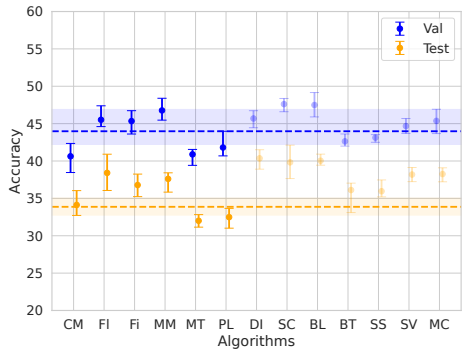
Figure C.3. **Upper: Res18. Lower: Res50.** The figure from left to right are Test Performance over time profiles of semi- and self-supervised methods on AIROGS dataset across **4 metircs: Balanced Accuracy, AUC, Partial AUC for 90% - 100% specificity and Sensitivity at 95% specificity (panels a-c)**. At each time, we report mean of each method over 5 trials of Alg. D.1.

C.4. Variance Across Trials

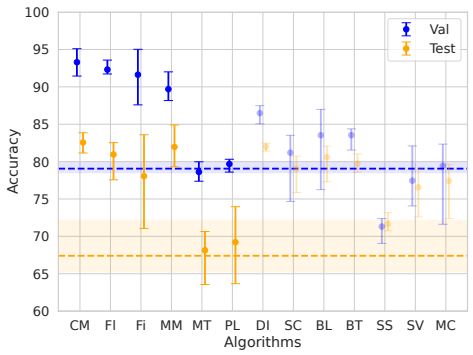
In Fig. C.4 below, we explicitly visualize the variability in performance of each method across the 5 separate trials of Alg. D.1 (most other figures show the mean of these 5 trials for visual clarity).



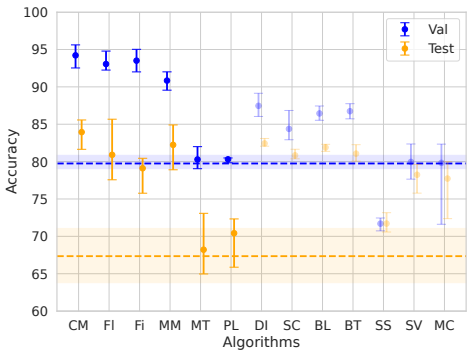
(a) TissueMNIST after 16h



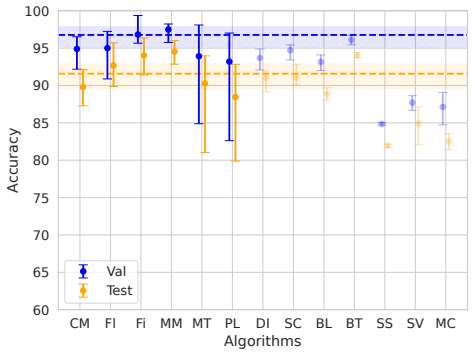
(b) TissueMNIST after 50h



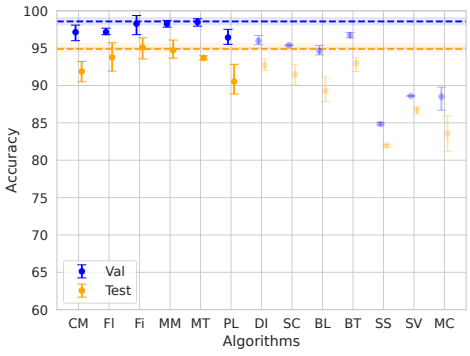
(c) PathMNIST after 8h



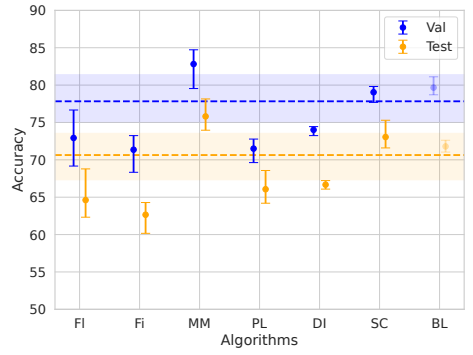
(d) PathMNIST after 25h



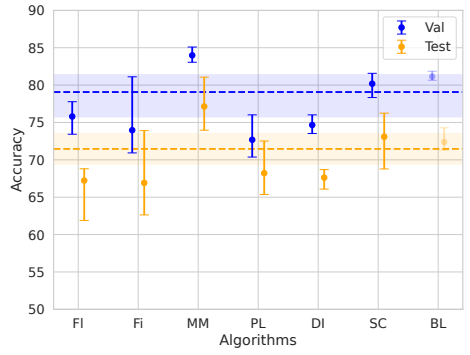
(e) TMED-2 after 32h



(f) TMED-2 after 100h



(g) AIROGS(res18) after 32h



(h) AIROGS(res18) after 100h

Figure C.4. Balanced accuracy of different methods across 2 time budgets (columns) and four datasets (rows). For each method, the interval indicates the low and high performance of 5 separate trials of Alg. D.1, while dot indicates the mean performance. Horizontal lines indicate the best labeled-set-only baseline at that time. Abbreviation: CM, Fl, Fi, MM, MT, PL, DI, SC, BL, BT, SS, SV, MC denote CoMatch, FlexMatch, FixMatch, MixMatch, Mean Teacher, Pseudo Label, DINO, SimCLR, BYOL, Barlow Twins, SimSiam, SwAV, MOCO (v2).

D. Algorithms Details

D.1. Algorithm : Random Hyperparameter search on a budget

Algorithm D.1 outlines our uniform hyperparameter tuning procedures used across all algorithms under comparison. The algorithm requires three sources of data: a labeled training set $\mathcal{L} = \{X, Y\}$, an unlabeled set for training $\mathcal{U} = X^U$, and a separate realistically-sized labeled validation set $\{X^{val}, Y^{val}\}$. We further require some budget restrictions: a common computational budget T (maximum number of hours), and a maximum training epoch per hyperparameter configuration E.

We proceed as follows: We begin by randomly sampling a hyperparameter configuration from a defined range (see Appendix F.1 for details). A model is then initialized and trained using the ADAM optimizer with the sampled hyperparameters. Each configuration is trained for a maximum of E (200) epochs or stopped early if the validation performance does not improve for 20 consecutive epochs. The model's performance on the validation set is measured using balanced accuracy. Upon completion of training for a given hyperparameter configuration (either after reaching maximum epoch E or after early stopping), a new configuration is sampled and the process repeats until the total compute budget T is expended.

We track the best-so-far model performance every 30 minutes, and save the best-so-far model and its corresponding validation and test performance. Semi-supervised algorithms simultaneously train the representation layers v and classifier layer w, while self-supervised algorithms train the representation layers v for each epoch and then fine-tune a linear classifier with weights w anew at the end of each epoch using sklearn logistic regression model [56] with representation parameters v frozen.

Algorithm D.1 Unified Procedure for Training + Hyperparameter selection via random search

Input:
- Train set of features $\mathbf{X}$ paired with labels $\mathbf{Y}$, with extra unlabeled features $\mathbf{U}$
- Validation set of features $\mathbf{X}^{\text{val}}$ and labels $\mathbf{Y}^{\text{val}}$
- Runtime budget T, Max Epoch E

Output: Trained weights $\{v, w\}$, where v is the representation module, w is the classifier layer

while time elapsed $<$ T **do**
  $\lambda \sim$ DRAWHYPERS ▷ Sample hyperparameters for pre-defined range
  $\xi \leftarrow$ CREATEOPTIM(λ) ▷ Initialize stateful optimizer e.g., ADAM
  $\{v, w\} \sim$ INITWEIGHTS ▷ Initialize model weights
  for epoch e in $1, 2, \ldots, E$ **do**
    if self-supervised **then**
      $v \leftarrow$ TRAINONEEPOCH($\mathbf{U}, v, \lambda, \xi$) ▷ Optimize Eq. with $\lambda^L = 0$
      $w \leftarrow$ TRAINCLASSIFIER($\mathbf{Y}, f_v(\mathbf{X})$)
    else if semi-supervised **then**
      $v, w \leftarrow$ TRAINONEEPOCH($\mathbf{X}, \mathbf{Y}, \mathbf{U}, v, w, \lambda, \xi$) ▷ Optimize Eq.
    else
      $v, w \leftarrow$ TRAINONEEPOCH($\mathbf{X}, \mathbf{Y}, v, w, \lambda, \xi$) ▷ Optimize Eq. with $\lambda^U = 0$
    end if
    $m_e \leftarrow$ CALCBALACC($\mathbf{X}^{\text{val}}, \mathbf{Y}^{\text{val}}, v, w$) ▷ Record performance metric on val.
    if first try or $m_e > m_*$ **then**
      $v_*, w_* \leftarrow v, w$; $\lambda_* \leftarrow \lambda$; $m_* \leftarrow m_e$ ▷ Update best config found so far
    end if
    if EARLYSTOP($m_1, m_2, \ldots, m_e$) or time elapsed $> T$ **then**
      break
    end if
  end for
end while
return v_*, w_*, λ_*, m_*

D.2. Hyperparameter transfer strategy

To make the most of limited labeled data, one potential strategy is to use the entire labeled set for training, reserving no validation set at all, and thus relies on pre-established hyperparameters from other dataset/experiments. In this study, we experiment with two scenarios: using pre-determined hyperparameters from 1. CIFAR-10, or 2. TissueMNIST. The CIFAR-10 hyperparameters are sourced from public repositories, we ensure that our CIFAR-10 hyperparameter choices on our re-implementation of the algorithms matches previously reported results in the literature. The TissueMNIST hyperparameters originate from our experiments as depicted in Figure C.2 (a).

D.3. Semi-supervised method details.

Semi-supervised learning trains on the labeled and unlabeled data simultaneously, usually with the total loss being a weighted sum of a labeled loss term and an unlabeled loss term. Different methods mainly differs in how unlabeled data is used to form training signals. Many approaches have been proposed and refined over the past decades. These include co-training, which involves training multiple classifiers on various views of the input data [6, 53]; graph-structure-based models [40, 76]; generative models [46, 47]; consistency regularization-based models that enforce consistent model outputs [4, 49, 62]; pseudo label-based models that impute labels for unlabeled data [11, 50]; and hybrid models that combines several methods [58]. Comprehensive reviews can be found in Chapelle et al. [13], Van Engelen and Hoos [63], Zhu [77].

Among the deep classifier methods following Eq. (1), below we describe each method we selected and how its specific unlabeled loss is constructed.

Pseudo-Labeling uses the current model to assign class probabilities to each sample in the unlabeled batch. If, for an unlabeled sample, the maximum class probability $P(y_i)$ exceeds a certain threshold τ, this sample contributes to the calculation of the unlabeled loss for the current batch. The cross-entropy loss is computed as if the true label of this sample is class i.

Mean-Teacher constructs the unlabeled loss by enforcing consistency between the model's output for a given sample and the output of the same sample from the Exponential Moving Average (EMA) model.

MixMatch uses the MixUp [74] technique on both labeled data (features and labels) and unlabeled data (features and guessed labels) within each batch to produce transformed labeled and unlabeled data. The labeled and unlabeled losses are then calculated using these transformed samples. Specifically, the unlabeled loss is derived from the mean squared error between the model's output for the transformed unlabeled samples and their corresponding transformed guessed labels.

FixMatch generates two augmentation of an unlabeled sample, one with weak augmentation and the other using strong augmentations (e.g., RandAug [20]). The unlabeled loss is then formulated by enforcing the model's output for the strongly augmented sample to closely resemble that of the weakly augmented sample using cross-entropy loss.

FlexMatch builds directly upon FixMatch by incorporating a class-specific threshold on the unlabeled samples during training.

CoMatch marks the first introduction of contrastive learning into semi-supervised learning. The model is equipped with two distinct heads: a classification head, which outputs class probabilities for a given sample, and a projection head, which maps the sample into a low-dimensional embedding. These two components interact in a unique manner. The projection head-derived embeddings inform the similarities between different samples, which are then used to refine the pseudo-labels against which the classification head is trained. Subsequently, these pseudo-labels constitute a pseudo-label graph that trains the embedding graph produced by the projection head.

D.4. Self-supervised method details

In recent years, self-supervised learning algorithms have emerged rapidly and are known as one of the most popular field of machine learning. These include contrastive learning, which involves learning representations by maximizing agreement between differently augmented views of the same data [15, 34]; predictive models that forecast future instances in the data sequence [55]; generative models that learn to generate new data similar to the input [14]; clustering-based approaches that learn representations by grouping similar instances [8, 9]; context-based models that predict a specific part of the data from

other parts [7, 23]; and hybrid models that combine various methods for more robust learning [16]. A more comprehensive review can be found in [41, 78].

Below, we provide for each selected self-supervised method a summary of its internal workings.

SimCLR generates two augmented versions of each image. Then feed these pairs of images into a base encoder network to generate image embeddings. This encoder is followed by a projection head, which is a multilayer neural network, to map these embeddings to a space where contrastive loss can be applied. Next, calculate the contrastive loss. The idea is to make the embeddings of augmented versions of the same image (positive pairs) as similar as possible and to push apart embeddings from different images (negative pairs). The loss function used is NCE loss.

MOCO V2 creates two augmented versions of each image. These pairs are processed by two encoder networks: a query encoder, and a key encoder updated by a moving average of the query encoder. The contrastive loss is computed by comparing a positive pair (the query and corresponding key) against numerous negative pairs drawn from a large queue of keys.

Note on runtime: We notice that the performance on MoCo can be increased when Shuffling BN across multiple GPUs. However, to ensure a fair comparison given our single-GPU setup, we refrained from employing any techniques to simulate multiple GPUs on one, as this would change the encoder's structure.

SwAV begins by creating multiple augmented versions of each image. Then, these versions are input into a deep neural network to generate embeddings. Uses a clustering approach, called online stratified sampling, to predict assignments of each view's prototypes (or cluster centers) to others, encouraging the model to match the representations of different augmentations of the same image.

Note on runtime: We've observed that applying multiple augmentations can enhance the effectiveness of various methods. To prevent the results from being influenced by these augmentations, we've standardized the number of augmentations to two in SwAV, in line with the approach taken by other methods.

BYOL starts by creating two differently augmented versions of each image. These versions are processed through two identical neural networks, known as the target and online networks, which include a backbone and a projection head. The online network is updated through backpropagation, while the target network's weights are updated as a moving average of the online network's weights. The unique aspect of BYOL is that it learns representations without the need for negative samples.

SimSiam creates two differently augmented versions of each image. These versions are passed through two identical networks: one predictor network and one encoder network. The encoder network contains a backbone and a projection head.

DINO utilizes two differently augmented images, processed by a student and a teacher network. The teacher's weights evolve as a moving average of the student's. The key idea is self-distillation, where the student's outputs match the teacher's for one view but differ for the other, without traditional negative samples.

Barlow Twins processes two augmented views of an image through identical networks. The aim is to have similar representations between these networks while minimizing redundancy in their components, sidestepping the need for contrasting positive and negative pairs.

E. Additional Analysis

E.1. Hyperparameter Tuning and Model selection with realistic validation set

E.1.1 Effectivness of Hyperparameter Tuning

While Oliver et al. [54] caution that extensive hyperparameter search may be futile with realistic validation set. Our experiments on the 4 dataset show that the validation set performance for each examined algorithm rise substantially over the course of hyperparameter tuning. This increase in validation set performance further translates to increased test set performance.

This means that for a chosen algorithm on a new dataset, following our hyperparameter tuning protocol (even with limited labeling budget and computation budget), we can obtain better generalization (measured by test set performance).

E.1.2 Differentiating Between Models

Oliver et al. [54] in their Fig 5 and 6 show that on SVHN, between 10 random samples of the validation set across several level of validation set size (1000, 500, 250, 100), the validation accuracy of the trained Pi-model, VAT, Pseudo-labeling and Mean Teacher model has substantial variability and overlap with each other. Thus, they caution that differentiating between models might be infeasible with realistic validation set size.

In this study, we employ a relaxed notion of "realistic validation set", by letting the validation set to be at most as large as the training set. Our experiments cover validation set size 235 (TMED), 400 (Tissue), 450 (Path), 600 (AIROGS); test set size 2019 (TMED2), 47280 (Tissue), 7180 (Path), 6000 (AIROGS). Our experiment shows that within the wide range of methods considered, differentiating between some models are possible. For example, we can see that MixMatch is clearly better than Mean Teacher in TissueMNIST and PathMNIST, in both the validation set and test set, without overlap on the intervals. The field of semi-supervised learning has made significant advancements in recent years. **It is crucial to reevaluate previous conclusions in light of the new developments.**

E.1.3 Theoretical Analysis

Here, we show that the performance gain we observe on the test set are real. We perform the same theoretical analysis using the Hoeffding's inequality Hoeffding [35] as in Oliver et al. [54].

$$\mathbf{P}(|\bar{V} - \mathbb{E}[V]| < p) > 1 - 2\exp(-2np^2) \tag{3}$$

where $\bar{V}$ is the empirical estimate of some model performance metric, $\mathbb{E}[V]$ is its hypothetical true value, p is the desired maximum deviation between our estimate and the true value, and n is the number of examples used.

On TissueMNIST, we have 47280 test samples, we will be more than 99.98% confident that the test accuracy is within 1% of its true value. On Path, we have 7180 test samples, we will be more than 99% confident that the test accuracy is within 2% its true value.

In Fig 1, we see that after hyperparameter tuning, the final test accuracy of each algorithms improves much more than 1% on TissueMNIST and 2% on PathMNIST showing the efficacy of hyperparameter tuning.

Similarly, we can see that the difference between top-performing algorithms (e.g., MixMatch) and worst-performaning alogrithm (e.g., Mean Teacher) is clearly larger then 1% on TissueMNIST, 2% on PathMNIST. Thus we can argue that differentiation between certain methods are viable. Same analysis can also be applied to TMED-2 and AIROGS.

F. Hyperparameter Details

F.1. Search Range

Below we show the search range of each hyperparameter.

.

Shared By All	
Optimizer	Adam
Learning rate schedule	Cosine

Labeled only	
Batch size	64
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5, -2)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-6, -3)$

MixUp	
Batch size	64
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5, -2)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-6, -3)$
Beta shape α	$x, X \sim \text{Uniform}(0.1, 10)$

Sup Contrast	
Batch size	256
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5.5, -1.5)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-7.5, -3.5)$
Temperature	$x, X \sim \text{Uniform}(0.05, 0.15)$

FlexMatch	
Labeled batch size	64
Unlabeled batch size	448
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5, -2)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-6, -3)$
Unlabeled loss coefficient	$10^x, X \sim \text{Uniform}(-1, 1)$
Unlabeled loss warmup schedule	No warmup
Pseudo-label threshold	0.95
Sharpening temperature	1.0

FixMatch	
Labeled batch size	64
Unlabeled batch size	448
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5, -2)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-6, -3)$
Unlabeled loss coefficient	$10^x, X \sim \text{Uniform}(-1, 1)$
Unlabeled loss warmup schedule	No warmup
Pseudo-label threshold	0.95
Sharpening temperature	1.0

CoMatch	
Labeled batch size	64
Unlabeled batch size	448
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5, -2)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-6, -3)$
Unlabeled loss coefficient	$10^x, X \sim \text{Uniform}(-1, 1)$
Unlabeled loss warmup schedule	No warmup
Contrastive loss coefficient	$5 \times 10^x, X \sim \text{Uniform}(-1, 1)$
Pseudo-label threshold	0.95
Sharpening temperature	0.2

For TMED2, unlabeled batch size is set to 320 to reduce GPU memory usage

MixMatch	
Labeled batch size	64
Unlabeled batch size	64
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5, -2)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-6, -3)$
Beta shape α	$x, X \sim \text{Uniform}(0.1, 1)$
Unlabeled loss coefficient	$7.5 \times 10^x, X \sim \text{Uniform}(0, 2)$
Unlabeled loss warmup schedule	linear
Sharpening temperature	0.5

Mean Teacher	
Labeled batch size	64
Unlabeled batch size	64
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5, -2)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-6, -3)$
Unlabeled loss coefficient	$8 \times 10^x, X \sim \text{Uniform}(-1, 1)$
Unlabeled loss warmup schedule	linear

Pseudo-label	
Labeled batch size	64
Unlabeled batch size	64
Learning rate	$3 \times 10^x, X \sim \text{Uniform}(-5, -2)$
Weight decay	$4 \times 10^x, X \sim \text{Uniform}(-6, -3)$
Unlabeled loss coefficient	$10^x, X \sim \text{Uniform}(-1, 1)$
Unlabeled loss warmup schedule	Linear
Pseudo-label threshold	0.95

SwAV	
Batch size	256
Learning rate	$1 \times 10^x, X \sim \text{Uniform}(-4.5, -1.5)$
Weight decay	$1 \times 10^x, X \sim \text{Uniform}(-6.5, -3.5)$
Temperature	$x, X \sim \text{Uniform}(0.07, 0.12)$
number of prototypes	$1 \times 10^x, X \sim \text{Uniform}(1, 3)$

MoCo	
Batch size	256
Learning rate	$1 \times 10^x, X \sim \text{Uniform}(-4.5, -1.5)$
Weight decay	$1 \times 10^x, X \sim \text{Uniform}(-6.5, -3.5)$
Temperature	$x, X \sim \text{Uniform}(0.07, 0.12)$
Momentum	$x, X \sim \text{Uniform}(0.99, 0.9999)$

SimCLR	
Batch size	256
Learning rate	$1 \times 10^x, X \sim \text{Uniform}(-4.5, -1.5)$
Weight decay	$1 \times 10^x, X \sim \text{Uniform}(-6.5, -3.5)$
Temperature	$x, X \sim \text{Uniform}(0.07, 0.12)$

SimSiam	
Batch size	256
Learning rate	$1 \times 10^x, X \sim \text{Uniform}(-4.5, -1.5)$
Weight decay	$1 \times 10^x, X \sim \text{Uniform}(-6.5, -3.5)$

In practice, we round each sampled α value to the nearest tenth decimal place

BYOL	
Batch size	256
Learning rate	$1 \times 10^x, X \sim \text{Uniform}(-4.5, -1.5)$
Weight decay	$1 \times 10^x, X \sim \text{Uniform}(-6.5, -3.5)$
Temperature	$x, X \sim \text{Uniform}(0.07, 0.12)$
Momentum	$x, X \sim \text{Uniform}(0.99, 0.9999)$

DINO	
Batch size	256
Learning rate	$1 \times 10^x, X \sim \text{Uniform}(-4.5, -1.5)$
Weight decay	$1 \times 10^x, X \sim \text{Uniform}(-6.5, -3.5)$
Temperature	$x, X \sim \text{Uniform}(0.07, 0.12)$
Momentum	$x, X \sim \text{Uniform}(0.99, 0.9999)$

Barlow Twins	
Batch size	256
Learning rate	$1 \times 10^x, X \sim \text{Uniform}(-4.5, -1.5)$
Weight decay	$1 \times 10^x, X \sim \text{Uniform}(-6.5, -3.5)$
Temperature	$x, X \sim \text{Uniform}(0.07, 0.12)$
Momentum	$x, X \sim \text{Uniform}(0.99, 0.9999)$

F.2. Chosen Hyperparameters on TissueMNIST Used for Hyperparameter Transfer Experiments

Below we report the chosen hyperparameters on TissueMNIST that are used in the hyperparameter transfer experiments.

FlexMatch					
	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00036	0.00016	0.00016	0.00068	0.00006
Weight decay	0.00259	0.00001	0.00371	0.00023	0.002103
Unlabeled loss coefficient	2.22	0.82	5.00	1.94	6.09

FixMatch					
	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00074	0.00034	0.00392	0.00102	0.00037
Weight decay	0.00045	0.00315	0.00001	0.00005	0.00058
Unlabeled loss coefficient	3.08	6.70	1.85	1.46	0.47

CoMatch					
	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00124	0.00145	0.00061	0.00026	0.00113
Weight decay	0.00042	0.00009	0.00005	0.00009	0.00017
Unlabeled loss coefficient	0.30	1.71	1.26	2.74	0.46
Contrastive loss coefficient	1.26	2.21	3.71	0.56	1.37

MixMatch					
	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00028	0.00003	0.00018	0.00009	0.00005
Weight decay	0.000005	0.00195	0.00005	0.00085	0.00082
Beta shape α	0.2	0.9	0.9	0.8	0.7
Unlabeled loss coefficient	9.13	37.96	8.06	25.16	11.17

Mean Teacher					
	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00062	0.00022	0.00005	0.00128	0.00125
Weight decay	0.00189	0.00001	0.00008	0.00001	0.00001
Unlabeled loss coefficient	67.67	0.87	1.25	7.60	13.56

Pseudo-label

	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00007	0.00021	0.00005	0.00063	0.00060
Weight decay	0.00033	0.00093	0.00383	0.00005	0.00087
Unlabeled loss coefficient	0.19	0.16	8.73	0.82	0.25

SwAV

	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00065	0.00325	0.00012	0.00086	0.00196
Weight decay	0.0001497	0.0000056	0.0000006	0.0000021	0.0000003
Number of prototypes	845	131	36	201	59

MoCo

	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00288	0.00023	0.00043	0.00005	0.02629
Weight decay	0.000002	0.0000008	0.0000003	0.0000005	0.0000004
temperature	0.09331	0.07097	0.10987	0.07414	0.07080
Momentum	0.99242	0.99672	0.99267	0.99950	0.99538

SimCLR

	seed0	seed1	seed2	seed3	seed4
Learning rate	0.00217	0.00131	0.000640	0.00380	0.00136
Weight decay	0.00002	0.00001	0.00001	0.00001	0.00001
temperature	0.11719	0.10426	0.08652	0.07784	0.11478

SimSiam

	seed0	seed1	seed2	seed3	seed4
Learning rate	0.0002	0.00056	0.00013	0.00338	0.00098
Weight decay	0.000066	0.000046	0.000023	0.000001	0.000001

BYOL

	seed0	seed1	seed2	seed3	seed4
Learning rate	0.000245	0.001308	0.000371	0.001653	0.001959
Weight decay	0.0000007	0.0000057	0.0000004	0.000003	0.000001
Momentum	0.9928618	0.996167	0.9988484	0.9940063	0.9934791

DINO

	seed0	seed1	seed2	seed3	seed4
Learning rate	0.000245	0.001308	0.000371	0.001653	0.001959
Weight decay	0.0000007	0.0000057	0.0000004	0.000003	0.000001
Momentum	0.9928618	0.996167	0.9988484	0.9940063	0.9934791

Barlow Twins

	seed0	seed1	seed2	seed3	seed4
Learning rate	0.000245	0.001308	0.000371	0.001653	0.001959
Weight decay	0.0000007	0.0000057	0.0000004	0.000003	0.000001
Momentum	0.9928618	0.996167	0.9988484	0.9940063	0.9934791