CHALMERS
UNIVERSITY OF TECHNOLOGY

An empirical evaluation of deep semi-supervised learning

Downloaded from: https://research.chalmers.se, 2026-09-15 03:46 UTC

Citation for the original published paper (version of record):
Fredriksson, T., Bosch, J., Olsson, H. (2025). An empirical evaluation of deep semi-supervised learning. International Journal of Data Science and Analytics, 20(4): 4127-4148. http://dx.doi.org/10.1007/s41060-024-00713-8

N.B. When citing this work, cite the original published paper.

research.chalmers.se offers the possibility of retrieving research publications produced at Chalmers University of Technology. It covers all kind of research output: articles, dissertations, conference papers, reports etc. since 2004. research.chalmers.se is administrated and maintained by Chalmers Library

(article starts on next page)

International Journal of Data Science and Analytics (2025) 20:4127–4148
https://doi.org/10.1007/s41060-024-00713-8

REGULAR PAPER

An empirical evaluation of deep semi-supervised learning

Teodor Fredriksson[1] · **Jan Bosch**[1] · **Helena H. Olsson**[2]

Received: 26 March 2024 / Accepted: 23 December 2024 / Published online: 21 January 2025
© The Author(s) 2025

Abstract
Obtaining labels for supervised learning is time-consuming, and practitioners seek to minimize manual labeling. Semi-supervised learning allows practitioners to eliminate manual labeling by including unlabeled data in the training process. With many deep semi-supervised algorithms and applications available, practitioners need guidelines to select the optimal labeling algorithm for their problem. The performance of new algorithms is rarely compared against existing algorithms on real-world data. This study empirically evaluates 16 deep semi-supervised learning algorithms to fill the research gap. To investigate whether the algorithms perform differently in different scenarios, the algorithms are run on 15 commonly known datasets of three datatypes (image, text and sound). Since manual data labeling is expensive, practitioners must know how many manually labeled instances are needed to achieve the lowest error rates. Therefore, this study utilizes different configurations for the number of available labels to study the manual effort required for optimal error rate. Additionally, to study how different algorithms perform on real-world datasets, the researchers add noise to the datasets to mirror real-world datasets. The study utilizes the Bradley–Terry model to rank the algorithms based on error rates and the Binomial model to investigate the probability of achieving an error rate lower than 10%. The results demonstrate that utilizing unlabeled data with semi-supervised learning may improve classification accuracy over supervised learning. Based on the results, the authors recommend FreeMatch, SimMatch, and SoftMatch since they provide the lowest error rate and have a high probability of achieving an error rate below 10% on noisy datasets.

Keywords Data labeling · Software engineering · Semi-supervised learning · Bayesian data analysis

1 Introduction

Many industries have recently started implementing machine learning algorithms for various tasks. Among these tasks, practitioners utilize supervised learning algorithms to solve classification tasks such as classifying images and text. For companies to utilize supervised classification, datasets need to be fully labeled. However, datasets are rarely fully labeled in industry, and it is difficult to obtain labels for many reasons [1]. One reason is that the data needs to be manually labeled, and companies need qualified in-house personnel to obtain high-quality labels. Specialists such as data scientists and software engineers are suitable labelers but are busy with more specialized tasks and do not have time for labeling. If specialized personnel are unable label, the other personnel must undergo training to perform labeling. On the other hand, training in-house personnel is expensive in terms of time and resources. A solution to the problems associated with in-house labeling is crowdsourced labeling [2, 3]. Crowdsourcing allows practitioners to obtain labels by outsourcing manual labeling to a group of people through crowdsourcing platforms such as Amazon Mechanical Turk.[1] A problem with crowdsourcing is that it is difficult to guarantee the quality of the labels since the labelers may lack the required expertise and knowledge. In addition, crowdsourc-

Jan Bosch and Helena H. Olsson contributed equally to this work.

✉ Teodor Fredriksson
teodorf@chalmers.se

Jan Bosch
jan.bosch@chalmers.se

Helena H. Olsson
helena.holmstrom.olsson@mau.se

[1] Department of Computer Science and Engineering, Chalmers University of Technology, Hörselgången 5, 41296 Göteborg, Västra Götaland, Sweden

[2] Department of Computer Science and Media Technology, Malmö University, Nordenskiöldsgatan 1, 21119 Malmö, Skåne, Sweden

[1] https://www.mturk.com.

ing is expensive, and companies with confidential data are not allowed share such data through labeling services. These two problems make manual labeling through crowdsourcing unappealing, and companies prefer to implement automated labeling approaches [3, 4].

According to [5], semi-supervised learning is a popular tool to reduce manual labeling. Semi-supervised learning applies to image, video, sound, text, and tabular datasets, but industry practitioners rarely utilize it [5]. Theoretical and empirical studies demonstrate that semi-supervised learning is not guaranteed to outperform supervised learning and may even degrade performance [6–10]. Due to the success of deep learning, the interest in deep semi-supervised learning [11–16] has increased. Development and evaluation of algorithms are time-consuming, so practitioners rely on benchmark studies to choose algorithms [17]. There are four dimensions to consider in semi-supervised learning. *Performance:* Which algorithms have the lowest error rate? *Datatype:* How do the algorithms perform differently depending on the datatype, *Manual Effort:* how many labels are required for an algorithm to reach optimal error rate and, *Robustness:* the capacity of a model to maintain its predictive performance when the input data is subject to noise and perturbations [18]. Recently, Microsoft made the *Unified Semi-Supervised learning Benchmark* (USB) [2] publicly available for practitioners, which lets them experiment and evaluate deep semi-supervised learning algorithms. At the time of this study, USB contains 16 deep semi-supervised learning algorithms evaluated on 15 datasets across three datatypes: image, text and audio. Since the first evaluation of USB [19], two more algorithms, FreeMatch [20] and SoftMatch [21], have been added to USB. All 16 algorithms, including SoftMatch, FreeMatch and supervised learning, are included in this study.

This study reports an empirical evaluation of deep semi-supervised learning algorithms. The study aims to assess the algorithms' performance by measuring the error rate. To access the datatype and manual effort dimensions, the error rate is analyzed separately for different datatypes and different amounts of data, respectively. Last, the study analyses the robustness dimension by simulating how noise in the dataset affects the error rate.

The contribution of this study is multi-fold. First, the study presents the top three highest ranking algorithms with respect to the lowest error rate. Second, it provides the highest-ranking algorithms for each individual datatype. Third, it demonstrates to practitioners how much manual effort is needed to receive sufficient labels for optimal error rate. Fourth, the study presents which algorithms have a high probability of achieving an error rate below 10%. Fifth, it investigates whether noise in the datasets improve the performance of the algorithm and increase the algorithm's probability of reaching an error rate below 10%. The abovementioned aspect is important since many datasets in empirical evaluations are too easy for algorithms to learn. Empirical evaluations that only utilize easy datasets are unreliable. Furthermore, studies [18, 22] demonstrate that adding noise may improve the error rate of algorithms. Last, the study recommends the optimal algorithms for each scenario based on the lowest error rate and their ability to improve or maintain optimal error rate in the presence of noise. The algorithms are ranked with the Bayesian Bradley–Terry model. USB was initially evaluated in [19] by ranking the algorithms according to their *Friedman Ranks* [23]. Friedman ranks only provide point estimates of the ranks, which may lead to misleading conclusions regarding the performance of algorithms. Running each algorithm on each dataset many times utilizing different seeds lead to variations in performance. The Bayesian Bradley–Terry model accounts for that variation by modeling uncertainty [24, 25].

Thanks to the results of this study, practitioners in academia and industry have concrete guidelines on what algorithm is optimal for their labeling scenario. An optimal algorithm is in the top three highest ranking algorithms with respect to the error rate and have an increased probability of achieving an error rate below 10% in the presence of noise. According to the results FreeMatch, SimMatch and SoftMatch are the top three algorithms. None of the algorithms achieve an error rate below 10% without noise. FreeMatch is optimal for all datatypes for small allocation of labels. SimMatch is tied with FreeMatch and is optimal for image datasets and small allocation of labels. Finally, SoftMatch is optimal for text data for both small and large allocations of labels.

The paper is organized in the following manner. Section 2 describes the theory behind semi-supervised learning. Section 3 outlines the research method, how the simulations are set up, what software packages are utilized and how the algorithms are evaluated. Section 4 presents the results, and Sect. 5 discusses the results. Finally, the paper is concluded in Sect. 7. A replication package together with an in-depth analysis of the validity of the Bayesian Data Analysis are found at the online repository.[3]

2 Background

This section discusses related work and presents a theoretical overview of the machine learning and statistical tools utilized in this study.

[2] https://github.com/microsoft/Semi-supervised-learning.

[3] https://github.com/teodorf-bit/Bayesian-Data-Analysis-of-Universal-Semi-Supervised-Benchmark.

2.1 Labeling challenge in software engineering

Many machine learning tasks in the industry are concerned with supervised learning which requires labeled data. Labeling may be time-consuming and 80% of the time spent in a Machine Learning project is allocated to labeling [26]. Since labeling is time-consuming, it is relevant that the right personnel within the company do the labeling. Data scientists and software engineers might have to spend their time on more specialized tasks such as utilizing different programming languages to build machine learning models, perform statistical analysis, and collect data from databases such as SQL [1].

If the company does not have the resources to perform in-house labeling, third-party services such as crowdsourcing [27] are available. Examples of crowdsourcing platforms are *Amazon Mechanical Turk* [28][4] and *Lionbridge AI*.[5] Crowdsourcing encourages different labelers to label data by rewarding them. By utilizing crowdsourcing, companies do not need to develop their labeling infrastructure or hire and train labelers. The issues with this approach are that it is challenging to guarantee high-quality labels and that many companies may not share sensitive data.

A tool that helps reduce the manual labeling is *Active Learning* [29]. Active learning queries what instances to be labeled according to a *query strategy* that selects the instances based on how informative they are. Query strategies ensure that labelers do not waste time labeling random instances that will not reduce the error rate. The training set is updated by adding the newly labeled instances, and the model is retrained utilizing the updated training set. If the model has not reached the desired error rate, then the training set is updated, and the model is retrained until the error rate of the model is sufficient.

2.2 Semi-supervised learning

Machine learning and deep learning algorithms require large amounts of data to achieve low error rate. In the industry many datasets are missing labels either entirely or partially. In order to achieve high-performance classification algorithms without utilizing costly tools such as crowdsourcing and active learning for manual labeling, companies utilize *semi-supervised learning* [30]. Semi-supervised learning algorithms have been designed to learn from unlabeled and labeled data to improve the decision boundary acquired by supervised learning. As unlabeled data is often abundant in industrial settings, it is reasonable to utilize semi-supervised learning to improve the error rate.

There are four main assumptions in semi-supervised learning. The main assumption is that few labeled instances and many unlabeled instances are available. The three other assumptions put constraints on the distribution. These are the *smoothness*, *cluster*, and the *manifold* assumptions [31]. The smoothness assumption says that if two features lie close to each other in a high-density region, their output labels also lie close. The cluster tells us that if two features lie in the same cluster, they likely have the same class label. The manifold assumption, often considered a generalization of the two assumptions above, states that each datapoint lies on a manifold [30].

2.3 Universal semi-supervised benchmark (USB)

USB is an open-source platform for evaluating semi-supervised learning algorithms. It contains algorithms for various Computer Vision (CV), Natural Language Processing (NLP), and Audio-related tasks. The algorithms are evaluated utilizing 15 datasets equally distributed among the three tasks. Initially, the first evaluation of USB [19] utilizes Friedman Ranks [23] to rank a subset of the algorithms included in this study. Furthermore, this study ranks the algorithms utilizing the Bayesian Bradley–Terry model. Bayesian models are more interpretable, do not rely on p-values and have methods to validate results. In addition, this study provides more evidence to prove that the algorithms will work better on real-world datasets. Other studies [32] utilize *Item Response Theory* to illustrate that several datasets are inappropriate for evaluating algorithms. Real-world datasets contain noise and are more complex for algorithms to learn. This study evaluates whether the algorithm will perform well on real-world datasets by adding noise to the benchmark datasets to investigate if there is a change in the algorithm's performance. USB is an extension of its predecessor, TorchSSL, and evaluates the algorithms utilizing fewer labels. In addition, USB introduces pre-trained transformers to speed up training time for several algorithms. The supervised baselines utilized to evaluate USB in this study are: WRN [33], WRN-Var, Resnet [34], ViT [35], BERT [36], and Wave2vec-v2 [19]. The Semi-Supervised vised algorithms are, Pseudo-Labeling [37], Π-model [38], Mean-Teacher [39], VAT [40], MixMatch [41], ReMixMatch [42], UDA [43], FixMatch [44], Dash [45], CoMatch [46], CRMatch [47], FlexMatch [20], AdaMatch [48], SimMatch [49] and SoftMatch [21].

2.4 The Bradley–Terry model

The Bayesian version [50, 51] of the Bradley–Terry model [52, 53] is utilized for ranking and comparison of objects. Each outcome $y_{i,j}$ of the comparisons are binary variables, either taking value 1 with probability $p_{i,j}$ if i beats j or value

[4] https://www.mturk.com.

[5] https://www.lionbridge.com/machine-translation/.

0 with probability $1 - p_{i,j}$ otherwise. Therefore the outcomes $y_{i,j}$ are Bernoulli distributed:

$$y_{i,j} \sim \text{Bernoulli}(p_{i,j}). \tag{1}$$

Furthermore, the Bradley–Terry model assumes that the outcomes are independent. To rank n objects, the first step is to calculate the strength parameter $\mu \in \mathbb{R}$ of each object and then calculate the probability of object i beating object j:

$$p_{i,j} := P(i \text{ over } j) = \text{logit}^{-1}(\mu_i - \mu_j). \tag{2}$$

Next, the algorithms are ranked by strength parameter so that the highest ranking algorithm have highest value of strength parameter. The Bradley–Terry model's ability to calculate the probability of objects beating each other and access the reliability of ranks through uncertainty estimation makes it preferable to other models [50].

2.5 Logit generalized linear mix model for binomial samples

This study utilizes a *Generalized Linear Mixed Model* (GLMM) [54] to account for the random effect on each dataset. The Generalized Linear Mixed Model for Binomial samples [50, 54] calculates the probability of success (an algorithm yields a specific error rate). Let y_i be Bernoulli distributed observations:

$$y_i = \begin{cases} 1 & \text{if success} \\ 0 & \text{if failure} \end{cases}, \tag{3}$$

i.e., $y_i \sim \text{Bernoulli}(p)$.
For n samples $y_1, y_2, ..., y_n$, the sum of all outcomes will be binomial distributed:

$$y = \sum_{i=1}^{n} y_i \sim \text{Binomial}(n, p). \tag{4}$$

Hence, the binomial distribution is utilized as likelihood. The probability of success will be modeled as:

$$p = \text{logit}(P(y = 1)) = a + bx + u, \tag{5}$$

$$u \sim \text{Normal}(0, \sigma^2). \tag{6}$$

where a is the fixed effect, b is the log-odds ratio and u is the random effect.

3 Research method and data analysis

This section describes the datasets utilized, the data collection approach, and the tools utilized to analyze the results.

This simulation study is an empirical evaluation of deep semi-supervised learning algorithms for classification. Benchmark experiments were run as described in [17], but Bayesian data analysis (BDA) was utilized instead of frequentist statistics. BDA is recommended for empirical software engineering [55] and has been utilized for benchmark experiments in similar contexts [50]. In benchmark experiments, a contrived environment is set up to analyze and measure the differences in various techniques. Two common goals of benchmarking studies include *algorithm comparison* and *characterizing algorithms' performance by problem features*. The first goal is to compare the performance of many algorithms to understand the strengths and weaknesses of different algorithms for different types of problems. In the context of this study, the goal is to determine what algorithms achieve the lowest error rate. The second goal is to link features of the problem with the algorithms' performance. In this study, the features are the datatypes, and the objective is to investigate whether some algorithms perform better on a particular dataset and datatype. Lastly, the study investigates how much manual effort is required to achieve a certain error rate. To accomplish these goals, the following research questions were studied.

- *RQ1:* What are the top-3 highest ranking algorithms in terms of lowest error rate.
- *RQ2:* How do the algorithms rank differently according to a specific datatype?
- *RQ3:* How do the algorithms rank differently depending on the number of labeled instances in the dataset?
- *RQ4-a:* What algorithms have high probability of yielding an error rate $\varepsilon \leq 0.1$
- *RQ4-b:* What is the impact of noise in the probability of success of each algorithm's error rate $\varepsilon \leq 0.1$

RQ1 and *RQ2* have previously been answered in [19], but in this study, more algorithms were studied, and Bayesian Bradley–Terry ranks were utilized. Due to the Bayesian nature of Bradley–Terry ranks, they provide a more fair and accurate data analysis [24, 25] than frequentistic Friedman ranks [23].

3.1 Descriptive statistics

The collected data is assumed to be a sample of instances $x_1, \ldots, x_n$, independent and identically distributed (i.i.d) from a random variable (r.v) X. The following descriptive statistics were utilized to describe the data collected from the simulations. The *sample mean* is defined as:

$$\overline{x} = \frac{1}{n} \sum_{i=1}^{n} x_i. \tag{7}$$

The *sample variance* is defined as:

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}(x_i - \overline{x})^2, \tag{8}$$

Furthermore, the mean represents the average value of the sample. For $\alpha \in (0, 1)$, a real number q_α is called the α-*quantile* if:

$$P(X \leq q_\alpha) \geq \alpha. \tag{9}$$

If q_α is the α-quantile of a sample, then $\alpha\%$ of the instances in the sample distribution are greater than q_α. For $\alpha = 0.5$, the quantile $q_{0.50}$ splits the sample dataset into two equal sizes and is called the *median*. The difference between the 95% quantile and the 5% quantile is called the *interquantile* range:

$$\text{Range} = q_{0.95} - q_{0.05}, \tag{10}$$

Moreover, the interquantile range measures the spread of the data.

3.2 Bayesian data analysis

In this study, *Bayesian Data Analysis* (BDA) was utilized due to the many disadvantages of frequentistic statistics that have been reported [24, 25]. BDA is recommended for empirical software engineering due to its ability to mitigate the shortcomings of the frequentistic approach [55], and has previously been utilized to analyze other benchmarks [50, 56].

The *classical* view of statistics expresses probability in terms of random repeatable events. However, many events are not repeatable, so the classical view of viewing probability becomes useless. The existence of non-repeatable events motivates the *Bayesian* viewpoint to express probability as a measurement of uncertainty. This uncertainty is updated through new evidence. Suppose prior information of the parameter θ is available before observing evidence x. This prior information is expressed in a *prior* probability distribution $p(\theta)$. After observing evidence x, the updated information is expressed through the *posterior probability* $p(x|\theta)$ and is calculated with Bayes formula [57]:

$$p(\theta|x) = \frac{p(x|\theta)p(\theta)}{p(x)}, \tag{11}$$

where $p(x)$ is the marginal distribution. Utilizing Bayesian methods for modeling θ is advantageous because posterior distributions consider all values of θ compared to frequentistic statistics where θ is treated as a scalar. *Prior predictive checks* and *posterior predictive checks* are utilized to evaluate the suitability of the prior distribution and the quality of the resulting posterior distribution. Predictive checks are intuitive methods for evaluating results compared to test statistics and p-values [24, 25, 55].

3.3 Algorithms

The deep semi-supervised learning algorithms in USB are *inductive*. Inductive algorithms work just as supervised learning because the algorithms are trained with training and testing sets. However, semi-supervised learning utilizes both labeled and unlabeled data in the training set. The deep semi-supervised learning algorithms in USB were chosen for evaluation due to their popularity [58]. Fourteen different deep semi-supervised learning algorithms were evaluated in this study: Π-Model [38] *(pimodel)*, Mean-Teacher [39]*(meanteacher)*, Pseudo-Label [37]*(pseudolabel)*, VAT [40]*(vat)*, MixMatch [41]*(mixmatch)*, ReMixMatch [42] *(remixmatch)*, UDA [43]*(uda)*, FixMatch [44]*(fixmatch)*, FlexMatch [20]*(flexmatch)*, Dash [45]*(dash)*, AdaMatch [48]*(adamatch)*, CRMatch [47]*(crmatch)*, CoMatch [46] *(comatch)*, SimMatch [49]*(simmatch)*, SoftMatch [21]*(softmatch)* and FreeMatch [59]*(freematch)*. Supervised learning *(supervised)* was included in the evaluation to investigate when unlabeled data reduces the error rate.

3.4 Datasets

The 15 datasets utilized in this study are found in the list below. There are five datasets for each datatype: *image, text* and *audio*.

- *Image data:*
  - *Cifar-100:* [60]. The dataset contains 32 × 32 color images divided into 100 classes. Each class contains 600 images each.
  - *STL-10:* [61] The dataset contains 96 × 96 pixel images divided into ten classes: *airplane, bird, car, cat, deer, dog, horse, monkey, ship, truck*. Each class contains 1300 instances each. The images were collected from ImageNet.
  - *EuroSat:* [62, 63] The dataset contains 64 × 64 pixel images divided into ten classes, *AnnualCrop, Forest, HerbaceousVegetation, Highway, Industrial, Pasture, Permanent Crop, Residental, River and SeaLake*. All classes contain 3000 instances each, except *Permanent Crop* and *River* which contain 2500 instances each.
  - *TissueMNIST:* [64, 65] The dataset contains 32×32×7 gray-scale images of kidney cortex cells. There are eight classes and a total of 236,386 instances.
  - *Semi-Aves:* [66] The dataset contains images of birds divided into 1000 classes of different Aves bird

species. The images are sampled from the iNat-2018 dataset. There are 12,220 images and each class contains 23–250 instances.

- *Text data:*
  - *IMDB:* [67] The dataset contains movie reviews labeled as positive or negative. The dataset is utilized for binary sentiment classification and contains 50,000 instances.
  - *AG News:* [68] The dataset contains news articles collected from 2000 online web sources. There are four classes: *world, sports, business*, and *sci-tech*. Each class contains 31,900 instances for a total of 127,600 instances.
  - *Amazon Review:* [69] This dataset contains reviews from Amazon. It contains 233.1 million instances distributed across 5 classes.
  - *Yahoo! Answers:* [70] The dataset is a sample from the original corpus provided by the Yahoo! Research Alliance Webscope Program. The dataset is a text classification benchmark and contains the ten largest classes from the original dataset: *Security & Culture, Science & Mathematics, Health, Education & Reference, Computers & Internet, Sports, Business & Finance, Entertainment & Music, Family & Relations*, and *Politics & Government*. Each class contains 14,600 instances for a total of 146,000 instances.
  - *Yelp Review:* [71] The dataset contains reviews from Yelp and is divided into five classes: 1,2,3,4,5. There is a total of 10,000 instances [72].
- *Audio*
  - *GTZAN:* The dataset contains 30-second-long audio files. The dataset is divided into ten classes: *blue, classical, country, disco, hip-hop, jazz, metal, pop, reggae,* and *rock*. Each class contains 100 instances for each class.
  - *UrbanSound8K:* [71] The dataset contains 4-second-long audio files in.wav format. There are ten classes: *air conditioner, car horn, children playing, dog bark, drilling, engine idling, gunshot, jackhammer, siren,* and *street music*. The dataset contains a total of 8732 instances.
  - *FZDnoisy18K:* [73] The dataset contains 42.5 h of audio from Freesound. There are 16,860 instances in total, distributed across 20 classes: *Acoustic guitar, Bass guitar, Clapping, Coin, Crash cymbal, Dishes pots and pans, Engine, Fart, Fire, Fireworks, Class, Hi-hat, Piano, Rain, Slam, Sueak, Tearing, Walk footsteps, Wind,* and *Writing*.
  - *Keyword Spotting:* [74] The dataset contains audio files of people saying one-word commands. The dataset contains more than 60,000 audio files distributed across 30 classes.
  - *Esc50:* [75]. The dataset contains 5-second-long environmental audio recordings divided into 50 semantic classes. Each class contains 40 instances.

3.5 Data collection

This study utilized the results from evaluations previously performed by Microsoft. The simulation results are found at the USB GitHub repository[3]. All choices of hyperparameters for the simulations are found in the supplementary material of the original paper [19]. The USB repository contains intervals around the mean error rate for each algorithm on each dataset. The intervals are on the form $(\overline{x} \pm m)$, where x is the error rate. Assuming that $(\overline{x} \pm m)$ is a $(1-\alpha)\%$ CI and that $\overline{x} \sim \text{Normal}(\mu, \sigma)$ the $(1-\alpha)\%$ CI for μ was derived. Data was simulated from the $(1-\alpha)\%$ interval of μ is given by:

$$\overline{x} - \lambda_{\alpha/2}\frac{s}{\sqrt{n}} \leq \mu \leq \overline{x} + \lambda_{\alpha/2}\frac{s}{\sqrt{n}}. \tag{12}$$

If $m = \lambda_{\alpha/2}\frac{s}{\sqrt{n}}$, then $s = m\lambda_{\alpha/2}\sqrt{n}$ where $\lambda_{\alpha/2}$ is the $\alpha/2$-quantile. Therefore the true distribution of μ is $\text{Normal}(\overline{x}, m\lambda_{\alpha/2})$. For this study, the parameters were chosen as $n = 1000$ and $\alpha = 0.05$.

3.6 Experimental setup

The number of available labels was varied to answer questions regarding manual effort. This paper considers the two cases where the training contains a "small" number of labels and a "large" number of available labels. For "small" the number of labels varies depending on the dataset and for "large", there are between 2 and 5 times the "small" number of labels see Table 1. The simulations were computed for ten iterations utilizing different random seeds, and results were saved in a.csv file called the *master dataset*. The pseudocode for creating the master dataset is found in algorithm 1, and a sample of the master dataset is illustrated in Table 2.

Algorithm 1 Creating the dataset

Require: n
1: **for** $d \in$ dataset **do**
2: **for** $a \in$ algorithm **do**
3: **for** $\ell \in$ available labels **do**
4: Draw $(x_1, \cdots, x_n)$ from $\text{Normal}(\overline{x}, m\lambda_{\alpha/2})$.
5: **end for**
6: **end for**
7: **end for**
8: Concatenate into a data frame ε_d with n rows.
9: Concatenate all $\varepsilon_1, \cdots, \varepsilon_d$ into one data frame see Table 2.

Table 1 Summary table for the datasets

Datatype	Dataset	Labels (Small/Large)	Training data	Test data	Classes
Image	Cifar-100	2/4	50,000	10,000	100
	STL-10	2/4	50,000	10,000	100
	EuroSat	4/10	5000/10,000	8000	10
	TissueMNIST	10/50	165,466	47,280	8
	Semi-Aves	15/53	5959/26,640	4000	200
Text	IMDB	10/50	23,000	25,000	2
	Amazon Review	50/200	250,000	65,000	5
	Yelp Review	50/200	250,000	50,000	5
	AG News	10/50	100,000	7600	4
	Yahoo! Answers	50/200	500,000	60000	10
Sound	Keyword Spotting	5/20	18538	2567	10
	ESC-50	5/10	1200	400	50
	UrbanSound8K	10/40	7079	837	10
	FZDnoisy	52/171	1772/15813	947	20
	GTZAN	10//40	7000	1500	10

From left to right, the columns contain the datatype, the name of the dataset, the number of labels utilized for each class, the size of the training dataset, the size of the testing dataset and the number of classes in the dataset

Table 2 Sample of the generated dataset

error_rate	Dataset	Algorithm	Iteration number	Manual effort	Datatype
35.788	fsdnoisy	pimodel	5	Small	Audio
61.232	yahoo_answers	Pimodel	5	Small	Text
37.741	semi_aves	vat	6	Large	Image
34.721	yahoo_answers	vat	5	Large	Text
60.018	amazon_review	fixmatch	4	Small	Text
31.095	yahoo_answers	adamatch	1	Large	Text
48.716	gtzan	vat	8	Large	Audio
31.154	fsdnoisy	freematch	7	Small	Audio
60.009	urbansound8k	dash	9	Small	Audio
22.335	stl	Softmatch	2	Small	Image

3.7 Data analysis

3.7.1 Bradley–Terry model

The Bradley–Terry model described in Sect. 2.4 was utilized with $\mu = a_{\text{alg}} + a_{\text{bm}}$. Here, a_{alg} and a_{bm} are the fixed effects of the algorithms and the benchmarks, respectively. The priors were chosen to have the following distributions.

$$a_{alg,i} \sim \text{Normal}(0, 2), \tag{13}$$
$$a_{bm,i,j} \sim \text{Normal}(0, s), \tag{14}$$
$$s \sim \text{Exponential}(0.1). \tag{15}$$

for all scenarios.

3.7.2 Generalized linear mixed model (GLMM)

The model in Sect. 2.5 was utilized with $a = a_{\text{alg}} + a_{bm}$ and $b = b_{\text{noise}}$. Here, a_{alg} is the fixed effect of the algorithm, and a_{bm} is the fixed effect of each benchmark. The priors were chosen to have the following distributions:

$$a_{\text{alg},i} \sim \text{Normal}(0, d), \tag{16}$$
$$b_{\text{noise},i} \sim \text{Normal}(0, d), \tag{17}$$
$$a_{bm,j} \sim \text{Normal}(0, s), \tag{18}$$
$$s \sim \text{Exponential}(z). \tag{19}$$

where the variables d, z were chosen so that all the MCMC chains converge for each of the scenarios. For aggregated

data $d = 5$, $z = 0.1$, for audio $d = 8.5$, $s = 1.9$, for images, text, small allocation, and large allocation of labels $d = 8.5$, $z = 2.2$.

Table 3 Summary statistics for the error rate (aggregated)

Algorithm	Median	5%	95%	Range
adamatch	26.113	3.275	52.843	49.568
crmatch	28.846	2.467	57.829	55.362
Comatch	29.818	4.458	56.191	51.733
Dash	31.050	3.482	56.297	52.816
Fixmatch	28.684	2.522	56.985	54.463
Flexmatch	29.995	2.674	68.091	65.417
Freematch	26.429	2.874	54.844	51.970
Meanteacher	32.472	5.437	63.772	58.335
mixmatch	44.701	10.334	74.372	64.038
pimodel	41.073	11.866	81.752	69.886
Pseudolabel	34.859	5.084	60.871	55.786
Remixmatch	80.000	8.180	98.000	89.820
Simmatch	27.347	2.658	52.637	49.979
Softmatch	27.924	2.260	59.830	57.569
Supervised	33.863	4.854	59.981	55.127
uda	33.643	7.380	93.393	86.013
vat	33.638	2.853	64.891	62.038

Table 4 Summary statistics for the error rate (image data)

Algorithm	Median	5%	95%	Range
adamatch	21.804	3.626	59.667	56.041
crmatch	24.985	10.170	62.449	52.279
comatch	28.231	3.629	65.146	61.517
Dash	27.005	5.325	58.286	52.960
Fixmatch	29.135	4.584	61.240	56.655
Flexmatch	27.413	4.810	81.700	76.890
Freematch	23.442	3.323	60.397	57.074
Meanteacher	30.730	3.505	60.562	57.057
mixmatch	40.268	21.788	65.746	43.958
pimodel	36.693	10.415	76.959	66.544
Pseudolabel	30.067	3.817	65.154	61.338
Remixmatch	25.862	2.387	63.580	61.194
Simmatch	21.037	4.686	57.470	52.784
Softmatch	23.329	2.976	74.020	71.043
Supervised	33.287	6.411	60.125	53.714
uda	25.866	6.332	62.387	56.054
vat	27.406	8.467	58.261	49.795

4 Results

This section presents the results from the data analysis and summarises the results into guidelines for practitioners. The measurements utilized to describe the results are discussed in section 3.1.

Tables 3, 4, 5, 6, 7, 8, illustrate descriptive statistics for the error rates for each scenario. From left to right, the columns

Table 5 Summary statistics for the error rate (text data)

Algorithm	Median	5%	95%	Range
adamatch	30.827	6.236	52.806	46.570
crmatch	32.674	4.806	56.839	52.033
Comatch	33.274	4.459	54.208	49.749
Dash	35.263	3.627	57.921	54.294
Fixmatch	32.528	3.055	59.410	56.355
Flexmatch	33.466	3.598	53.348	49.751
Freematch	29.669	5.889	52.176	46.286
Meanteacher	38.513	8.153	63.545	55.391
Mixmatch	44.317	8.236	78.126	69.889
Pimodel	50.667	15.160	86.742	71.581
Pseudolabel	42.917	7.708	58.447	50.740
Remixmatch	80.000	50.000	90.000	40.000
Simmatch	32.439	5.377	52.301	46.924
Softmatch	33.866	4.226	51.319	47.092
Supervised	37.000	8.431	63.814	55.384
uda	57.691	21.537	115.187	93.651
vat	38.485	4.520	83.389	78.869

Table 6 Summary statistics for the error rate (audio data)

Algorithm	Median	5%	95%	Range
adamatch	25.213	2.491	39.255	36.764
crmatch	25.369	1.325	54.504	53.180
comatch	25.677	9.192	47.807	38.615
Dash	31.802	1.852	48.570	46.718
Fixmatch	23.316	1.854	49.496	47.641
Flexmatch	29.551	2.222	49.338	47.116
Freematch	26.333	2.368	57.522	55.155
Meanteacher	29.383	5.494	63.724	58.230
Mixmatch	50.611	9.969	75.715	65.746
Pimodel	39.693	19.141	64.029	44.887
Pseudolabel	30.908	4.929	59.690	54.761
Remixmatch	93.246	75.096	105.270	30.173
Simmatch	22.890	2.157	46.629	44.472
Softmatch	26.346	1.366	44.198	42.832
Supervised	32.137	1.828	53.046	51.219
uda	27.960	5.619	52.383	46.764
vat	34.829	2.296	51.224	48.929

Table 7 Summary statistics for the error rate (small allocation of labels)

Algorithm	Median	5%	95%	Range
adamatch	27.062	2.652	54.640	51.988
crmatch	31.654	2.834	60.215	57.381
Comatch	33.542	5.231	62.719	57.488
dash	34.553	4.164	58.762	54.598
Fixmatch	31.501	2.617	55.869	53.252
Flexmatch	31.694	2.659	60.062	57.404
Freematch	29.162	2.935	58.859	55.925
Meanteacher	36.885	5.817	65.507	59.691
Mixmatch	46.660	10.088	74.324	64.235
Pimodel	49.472	20.681	85.018	64.337
Pseudolabel	39.232	6.954	67.484	60.530
Remixmatch	80.000	8.451	98.000	89.549
Simmatch	29.969	2.401	54.586	52.185
Softmatch	29.811	2.692	67.251	64.559
Supervised	38.428	5.384	65.532	60.148
uda	40.277	9.917	86.941	77.025
vat	36.938	2.847	81.862	79.014

Table 8 Summary statistics for the error rate (large allocation of labels)

Algorithm	Median	5%	95%	Range
adamatch	25.202	4.501	49.681	45.180
crmatch	28.056	1.813	47.874	46.061
Comatch	25.651	4.135	48.926	44.791
Dash	28.555	2.470	50.240	47.771
Fixmatch	22.285	2.500	59.410	56.910
Flexmatch	27.987	3.188	84.791	81.602
Freematch	24.992	4.049	48.096	44.046
Meanteacher	30.143	5.206	54.357	49.151
Mixmatch	38.777	11.146	74.636	63.489
Pimodel	35.722	8.892	61.902	53.011
Pseudolabel	30.195	3.895	52.304	48.409
Remixmatch	80.000	9.777	98.000	88.223
Simmatch	24.682	3.272	50.560	47.288
Softmatch	25.487	1.989	44.927	42.938
Supervised	30.293	4.768	50.468	45.700
uda	31.854	5.662	98.290	92.628
vat	29.984	2.987	52.603	49.616

Table 9 Ranking of the algorithms (aggregated data)

Models	Median rank	Variance of the rank
adamatch	2	3.644
simmatch	3	4.649
Freematch	4	4.565
Softmatch	4	4.972
crmatch	5	4.918
Fixmatch	5	5.054
Flexmatch	5	5.080
Dash	8	3.452
comatch	9	3.587
Meanteacher	11	2.763
Supervised	11	2.583
vat	11	2.497
uda	12	2.366
Pseudolabel	14	1.610
pimodel	15	1.140
Mixmatch	16	0.771
Remixmatch	17	0.005

Table 10 Ranking of the algorithms (image data)

Models	Median rank	Variance of the rank
Freematch	3	7.323
Simmatch	3	7.169
adamatch	4	8.649
Remixmatch	4	8.201
Flexmatch	6	10.034
uda	6	10.349
crmatch	7	11.142
Dash	7	10.471
Fixmatch	8	11.620
Softmatch	8	10.730
Comatch	12	9.570
Supervised	12	8.426
Mixmatch	14	6.564
Pimodel	14	6.689
Vat	14	7.911
Pseudolabel	15	5.983
Meanteacher	16	3.951

in the tables illustrate the name of the algorithm, the median, 5% quantile, 95% quantile, and the interquantile range.

Table 3 illustrates the descriptive statistics of the error rate of the aggregated data. The descriptive statistics demonstrate that AdaMatch is the algorithm with the lowest error rate. Tables 4, 5 and 6 demonstrate that the algorithm with the lowest error rate differs when investigating datatype individually. The algorithms with the lowest error rates for image, text and audio are SimMatch, FreeMatch and SimMatch, respectively. Similarly, tables 7 and 8 demonstrate that the algorithm with the lowest error rate is different for small and large allocations of labels. For small allocation of labels, the algorithm with the lowest error rate is AdaMatch, and for large allocation of labels, the algorithm with the lowest error rate is SimMatch. In addition, the error rate decreases as the number of labels increases.

Table 11 Ranking of the algorithms (text data)

Models	Median rank	Variance of the rank
adamatch	3	5.406
crmatch	4	5.990
Softmatch	4	6.509
comatch	5	6.632
Fixmatch	5	6.670
Freematch	5	6.802
Simmatch	5	6.665
Dash	7	6.330
Flexmatch	7	5.657
Meanteacher	10	2.270
Supervised	11	1.719
vat	12	1.795
Mixmatch	14	1.585
Pimodel	14	1.581
Pseudolabel	14	1.490
uda	16	0.220
Remixmatch	17	0.003

Table 12 Ranking of the algorithms (audio data)

Models	Median rank	Variance of the rank
Fixmatch	3	6.201
Softmatch	3	5.346
adamatch	4	6.839
Simmatch	4	6.404
Freematch	6	8.865
uda	6	9.413
crmatch	7	8.640
Flexmatch	7	8.571
Comatch	10	8.254
Dash	11	7.286
Meanteacher	11	8.005
vat	11	7.784
Supervised	12	6.196
Pseudolabel	13	4.673
Pimodel	15	0.729
Mixmatch	16	0.195
Remixmatch	17	0.000

Table 13 Ranking of the algorithms (small allocation of labeled data)

Models	Median rank	Variance of the rank
adamatch	2	3.071
simmatch	3	3.671
Flexmatch	4	4.233
Freematch	4	4.108
Softmatch	4	4.277
crmatch	5	4.484
Fixmatch	6	4.405
Dash	8	3.996
Comatch	9	3.278
Meanteacher	11	2.945
vat	11	3.007
Supervised	12	2.661
uda	13	2.376
Mixmatch	14	1.972
Pseudolabel	15	1.383
Pimodel	16	0.480
Remixmatch	17	0.070

Table 14 Ranking of the algorithms (large allocation of labeled data)

Models	Median rank	Variance of the rank
Fixmatch	2.5	4.899
adamatch	3.0	5.053
Freematch	4.0	7.280
Simmatch	4.0	5.209
Softmatch	5.0	7.281
crmatch	6.0	7.472
Comatch	6.0	8.069
Flexmatch	7.0	8.249
Dash	9.0	7.606
Supervised	11.0	6.400
uda	11.0	7.358
Meanteacher	13.0	4.504
Pimodel	13.0	4.629
vat	13.0	5.253
Pseudolabel	14.0	3.375
Mixmatch	16.0	0.157
Remixmatch	17.0	0.089

4.1 Analysis of the ranks

The master dataset was altered differently to answer each research question before applying the models. To obtain the aggregated results, the column that contains manual effort and datatype was dropped from the master dataset. The manual effort column was ignored to obtain the results based on the datatype. Finally, the column containing the datatype was dropped to analyze the results based on manual effort.

Tables 9, 10, 11, 12, 13, 14 illustrate the rankings of the algorithms calculated utilizing the Bradley-Terry model described in section 3.7.1. The ranks are utilized to answer RQ1-RQ3. From left to right, the columns illustrate the name of the algorithm, the median rank and the sample variance of the ranks. The high variance indicates uncertainty in the esti-

mated ranks and explains why the ranks of many algorithms are tied. Table 15 illustrates which algorithms are always in the top-3 highest-ranking algorithms and table 16 illustrates the algorithms that are tied.

4.2 Analysis of the porbability of success

This research question was answered with respect to aggregated results, datatype, and manual effort. To answer RQ4, the following operations were performed on all three of the datasets that were utilized to answer the previous RQs. First, a copy of the dataset was made. A new column called "SD" (for standard deviation) was added to copied and original variants. In the original dataset, $SD = 0$ to indicate the absence of noise. In the copied dataset, $SD = 3$ to indicate noise in the data. To account for noise we simulated accuracy from a normal distribution with mean y and standard deviation 3. After the operations on the copied dataset were finished, both datasets were concatenated by row into a new dataset. When the GLMM was applied, the odds ratio (OR) of each algorithm's intercept (a_{alg}) and noise (b_{noise}) were computed utilizing the new dataset. OR measures the relative probability of success compared to the probability of failure. An OR > 1 means that the parameter increases the probability of success. If $0 \leq$ OR < 1, the parameter decreases the probability of success and if OR $= 1$, the probability of success is unchanged.

The ORs are located in tables 17, 18, 19, 20, 21, and 22. No algorithm has OR > 1 for the fixed effects in any scenario. A summary of the algorithms that have an OR > 1 for the noise parameters is located in table 23.

4.3 Guidelines for practitioners

The recommended algorithms are presented in table 24 based on the ranks and the probability of success. The recommendations are based on the algorithms in the top-three highest-ranking algorithms and have OR > 1 in the noise parameter. In other words, the algorithms achieve a low error rate and perform well in the presence of noise.

Practitioners are recommended to try FreeMatch because it is among the top three highest-ranking algorithms for each scenario. It has OR > 1 for the noise parameter in each scenario except for a small allocation of labels. Therefore, FreeMatch is recommended for all datatypes but works better with more labels. If utilizing image datasets and a small allocation of labels and practitioners are not satisfied with FreeMatch, they are recommended to try SimMatch. It shares the second-highest ranking algorithm spot with FreeMatch and has OR > 1 for the noise parameter. For text datasets and a small allocation of labels, practitioners are recommended to utilize SoftMatch if they are not satisfied with FreeMatch. SoftMatch is in the top-three highest ranking algorithms and outranks FreeMatch for text datasets. In addition, SoftMatch has OR > 1 for the noise parameter for text and a small allocation of labels.

5 Discussion

Semi-supervised learning is a combination of supervised learning and unsupervised learning where unlabeled data is utilized to improve supervised learning [76]. An unsupervised classifier is said to improve supervised learning well if it helps the classifier predict the correct label and if it provides *fairness*. Fairness means that the model outputs each class label with an equal frequency given that the class distribution in the training data is uniform [76]. Fairness is obtained when the mutual information is maximized [76].

The objective of semi-supervised learning algorithms is to minimize the total loss function, defined as the sum of a supervised loss and an unsupervised loss [37, 58]. The supervised loss involves labeled data, and the unsupervised loss involves unlabeled data. Semi-supervised learning algorithms based on other machine learning methods are available. However, previous empirical and theoretical studies demonstrate that semi-supervised machine learning algorithms may degrade performance [77, 78].

5.1 The quantity-quality tradeoff

In recent years, deep semi-supervised learning has increased in popularity due to the success of the FixMatch algorithm [44]. Previous algorithms like UDA, MixMatch and ReMixMatch precede FixMatch and are all inferior in many scenarios [44]. FixMatch takes many ideas from the previous techniques and simplifies them yet achieves better performance [44].

Many modern deep semi-supervised learning algorithms including FixMatch, utilize *pseudo-labeling* [37], and *consistency regularization* [43, 79]. Pseudo-labeling is a semi-supervised technique that trains a supervised classifier to classify pseudo-labels for unlabeled instances. These pseudo-labels improve the generalization performance by maximizing the conditional log-likelihood and minimizing the entropy of unlabeled data [37].

A problem with semi-supervised learning is that if labeled data is scarce, the supervised classifier will perform poorly and produce low-quality pseudo-labels [30]. To mitigate this, FixMatch utilizes a threshold that makes sure that only the pseudo-labels of sufficient quality are utilized. The consistency regularization aspect of FixMatch is to minimize the cross-entropy between the predictive distributions of the class labels given weakly augmented and strongly augmented instances. Utilizing pseudo-labeling leads to the quantity-quality trade-off [21]. The quantity-quality trade-off states

Table 15 Summary of top-three highest ranking algorithms

	Aggrgated	Image	Text	Audio	Small allocation of labels	Large allocation of labels
Π-model	NO	NO	NO	NO	NO	NO
Mean-Teacher	NO	NO	NO	NO	NO	NO
Pseudo-Label	NO	NO	NO	NO	NO	NO
VAT	NO	NO	NO	NO	NO	NO
MixMatch	NO	NO	NO	NO	NO	NO
ReMixMatch	NO	YES	NO	NO	NO	NO
UDA	NO	YES	NO	YES	NO	NO
FixMatch	NO	NO	YES	YES	NO	YES
FlexMatch	NO	YES	NO	NO	YES	NO
Dash	NO	NO	NO	NO	NO	NO
AdaMatch	YES	YES	YES	YES	YES	YES
CRMatch	NO	NO	YES	NO	NO	NO
CoMatch	NO	NO	NO	NO	NO	NO
SimMatch	YES	YES	YES	YES	YES	YES
SoftMatch	YES	NO	YES	YES	YES	NO
FreeMatch	YES	YES	YES	YES	YES	YES
Supervised	NO	NO	YES	NO	YES	YES

Table 16 Top-three highest ranking algorithms

Scenario	1st	2nd	3rd
Aggregated	AdaMatch	SimMatch	FreeMatch, SoftMatch
Image	FreeMatch, SimMatch	AdaMatch, ReMixMatch	FlexMatch, UDA
Text	AdaMatch	CRMatch, SoftMatch	CoMatch, FixMatch, FreeMatch, SimMatch
Audio	FixMatch, SoftMatch	AdaMatch, SimMatch	FreeMatch, UDA
Small allocation of labels	AdaMatch	SimMatch	FlexMatch, FreeMatch, SoftMatch
Large allocation of labels	FixMatch	AdaMatch	FreeMatch, SimMatch

that higher thresholds lead to fewer pseudo-labels in the training set. Based on the results of this study, the optimal algorithms are FreeMatch, SimMatch and SoftMatch, which all extend upon FixMatch and try to provide both high quality and quantity but in different ways. SimMatch and SoftMatch were both developed in the same year and were originally not compared to each other. FlexMatch was published a year after SoftMatch and SimMatch and consequently was not compared to these either. Therefore, none of the three was explicitly designed to outperform one another.

The pseudo-labels that FixMatch computes are called semantic pseudo-labels. SimMatch extends FixMatch and strives to include more high-quality pseudo-labels by utilizing both semantic and instance pseudo-labels. The instance pseudo-labels are calculated utilizing similarity distributions. Semantic and instance pseudo-labels are then matched to belong to the same class. The instance pseudo-labels are inputs in a third loss function called the *instance loss*. Unlike other semi-supervised learning algorithms, SimMatch is unique in this regard. The previous state-of-the-art algorithm was CoMatch which is based on consistency regularization and contrastive learning [46]. CoMatch utilizes similarity matching through label distribution, but SimMatch is faster, more robust and achieves better performance than CoMatch. The FixMatch and SimMatch algorithms utilize a fixed threshold during training to maintain high-quality pseudo-labels. On the downside, the algorithms discard many labels and reduce quality. Other algorithms, such as Dash and AdaMatch, utilize a dynamically increasing threshold to outperform algorithms that utilize a fixed threshold.

Like FixMatch, SoftMatch minimizes a total loss that is decomposed into a supervised and an unsupervised loss. The inputs of these losses are the same as FixMatch, but the unsupervised loss is the weighted cross-entropy, which requires a sample weight function. The sample weight function in SoftMatch is assumed to have a Gaussian truncated distribution whose mean and variance are estimated utilizing historical predictions of the model's exponential moving

Table 17 Odds ratios for fixed effects and noise parameters (aggregated data)

Parameter	OR Mean	OR HPD low	OR HPD high	OR Range
a_adamatch	0.334	0.067	2.334	2.268
a_crmatch	0.206	0.041	1.436	1.395
a_comatch	0.183	0.036	1.249	1.213
a_dash	0.178	0.035	1.296	1.261
a_fixmatch	0.240	0.047	1.683	1.637
a_flexmatch	0.297	0.058	2.119	2.061
a_freematch	0.175	0.034	1.240	1.206
a_meanteacher	0.092	0.018	0.635	0.617
a_mixmatch	0.021	0.004	0.154	0.150
a_pimodel	0.016	0.003	0.113	0.110
a_pseudolabel	0.098	0.019	0.680	0.660
a_remixmatch	0.034	0.006	0.251	0.245
a_simmatch	0.261	0.051	1.859	1.807
a_softmatch	0.334	0.065	2.353	2.288
a_supervised	0.057	0.011	0.411	0.399
a_uda	0.052	0.010	0.370	0.360
a_vat	0.110	0.022	0.771	0.749
b_adamatch	0.929	0.800	1.078	0.278
b_crmatch	1.106	0.946	1.289	0.343
b_comatch	0.956	0.814	1.120	0.305
b_dash	0.970	0.828	1.139	0.311
b_fixmatch	0.976	0.838	1.139	0.301
b_flexmatch	0.976	0.835	1.139	0.304
b_freematch	1.033	0.879	1.216	0.337
b_meanteacher	0.980	0.829	1.158	0.329
b_mixmatch	1.087	0.864	1.363	0.499
b_pimodel	1.031	0.794	1.333	0.539
b_pseudolabel	1.000	0.852	1.177	0.325
b_remixmatch	0.999	0.815	1.233	0.418
b_simmatch	1.006	0.856	1.180	0.325
b_softmatch	0.973	0.836	1.134	0.298
b_supervised	1.032	0.864	1.238	0.375
b_uda	0.982	0.806	1.193	0.387
b_vat	0.998	0.848	1.170	0.322

average. Therefore, the threshold varies at each time stamp, and theoretical arguments demonstrate that SoftMatch provides better quantity and quality over UDA, FixMatch and FlexMatch [21].

The FreeMatch algorithm utilizes supervised and unsupervised loss as FixMatch, but also calculates the self-adaptive threshold (SAT) to balance the quantity-quality trade-off by automatically adjusting the threshold during training. The threshold is low at the start of training, but as training continues, it becomes more confident and increases. The SAT is calculated by combining two other thresholds, known as global and local thresholds. The global threshold represents the model's confidence in unlabeled data and is computed utilizing EMA. The local threshold is class-specific and, therefore, considers that different class labels are easier than others to predict. Finally, fairness is included to make the label predictions fair. From simulations, it is observed that FreeMatch achieves superior performance on various benchmarks. ReMixMatch and UDA may outperform FreeMatch due to the MixUp property [80]. The results of [59] demonstrate that FreeMatch has a lower threshold in the early learning process than FlexMatch and FixMatch and, therefore, utilizes more data than these [59].

Table 18 Odds ratios for fixed effects and noise parameters (image data)

Parameter	OR mean	OR HPD low	OR HPD high	OR range
a_adamatch	0.152	0.030	0.940	0.911
a_crmatch	0.011	0.002	0.079	0.077
a_comatch	0.140	0.027	0.867	0.840
a_dash	0.070	0.014	0.437	0.423
a_fixmatch	0.064	0.012	0.413	0.401
a_flexmatch	0.071	0.014	0.465	0.451
a_freematch	0.068	0.014	0.442	0.428
a_meanteacher	0.034	0.006	0.223	0.217
a_mixmatch	0.002	0.000	0.018	0.018
a_pimodel	0.014	0.002	0.095	0.092
a_pseudolabel	0.041	0.008	0.257	0.249
a_remixmatch	0.093	0.018	0.596	0.578
a_simmatch	0.063	0.012	0.413	0.401
a_softmatch	0.153	0.030	0.983	0.952
a_supervised	0.017	0.003	0.117	0.114
a_uda	0.033	0.006	0.219	0.213
a_vat	0.026	0.005	0.170	0.165
b_adamatch	0.952	0.763	1.182	0.420
b_crmatch	1.923	1.394	2.765	1.371
b_comatch	0.924	0.732	1.162	0.430
b_dash	0.936	0.737	1.192	0.455
b_fixmatch	0.995	0.773	1.275	0.502
b_flexmatch	1.031	0.815	1.302	0.487
b_freematch	1.032	0.813	1.309	0.496
b_meanteacher	0.864	0.650	1.141	0.490
b_mixmatch	0.951	0.354	2.450	2.096
b_pimodel	1.002	0.693	1.450	0.758
b_pseudolabel	0.999	0.786	1.273	0.487
b_remixmatch	0.997	0.789	1.256	0.467
b_simmatch	1.086	0.845	1.387	0.542
b_softmatch	0.968	0.770	1.220	0.450
b_supervised	1.055	0.769	1.447	0.678
b_uda	1.020	0.776	1.350	0.574
b_vat	0.999	0.756	1.320	0.564

5.2 Benchmarking

Due to the rapid development and publication of deep semi-supervised learning algorithms, continuous benchmarking is necessary. Furthermore, benchmarking is only able to evaluate methods implemented in a current release of the software. New releases of a method may differ in accuracy and runtime, which is why permanent benchmarking efforts are necessary. In addition, datasets utilized to evaluate algorithms need to be updated due to the fact that many datasets are too easy for the algorithms to learn. In [32], *Item Response Theory* demonstrates that many datasets utilized to evaluate graph-based semi-supervised learning algorithms are too easy to learn. Therefore, many algorithms will achieve low error rate on benchmark datasets but may perform differently on real-world datasets because they are more difficult to learn. In particular, only four out of 15 benchmark datasets are suitable for evaluating graph-based semi-supervised learning [32]. Similarly, [81] demonstrates that many supervised machine learning algorithms suffer from the same problem. Furthermore, USB utilizes many benchmark datasets that have been proven too easy for supervised learning and graph-based semi-supervised learning. Therefore, determining whether these datasets are suitable for evaluating the algorithms contained in USB is essential. Datasets must be updated or discarded with time due to their ability to evaluate

Table 19 Odds ratios for fixed effects and noise parameters (text data)

Parameter	OR mean	OR HPD low	OR HPD high	OR range
a_adamatch	0.054	0.008	0.439	0.430
a_crmatch	0.127	0.019	1.098	1.079
a_comatch	0.055	0.008	0.455	0.447
a_dash	0.058	0.009	0.491	0.482
a_fixmatch	0.082	0.013	0.706	0.693
a_flexmatch	0.088	0.013	0.744	0.731
a_freematch	0.042	0.006	0.354	0.347
a_meanteacher	0.012	0.002	0.106	0.104
a_mixmatch	0.015	0.002	0.129	0.126
a_pimodel	0.006	0.001	0.053	0.052
a_pseudolabel	0.015	0.002	0.125	0.123
a_remixmatch	0.000	0.000	0.001	0.001
a_simmatch	0.060	0.009	0.493	0.484
a_softmatch	0.035	0.005	0.304	0.299
a_supervised	0.011	0.002	0.099	0.097
a_uda	0.001	0.000	0.013	0.013
a_vat	0.024	0.003	0.198	0.195
b_adamatch	0.893	0.688	1.155	0.466
b_crmatch	0.846	0.657	1.089	0.432
b_comatch	1.018	0.791	1.306	0.515
b_dash	0.975	0.753	1.269	0.516
b_fixmatch	0.946	0.734	1.213	0.479
b_flexmatch	0.906	0.694	1.190	0.496
b_freematch	1.011	0.791	1.298	0.507
b_meanteacher	1.172	0.852	1.618	0.766
b_mixmatch	1.066	0.781	1.459	0.678
b_pimodel	0.991	0.645	1.520	0.875
b_pseudolabel	0.996	0.726	1.366	0.639
b_remixmatch	0.003	0.000	3.455	3.455
b_simmatch	0.958	0.732	1.250	0.518
b_softmatch	1.065	0.813	1.391	0.578
b_supervised	1.059	0.768	1.469	0.702
b_uda	0.956	0.375	2.394	2.019
b_vat	1.024	0.774	1.358	0.584

the algorithms [32] and it is important to include real-world datasets in evaluations. Furthermore, benchmarking is essential for the industry as they must know what algorithms to utilize on their real-world datasets.

5.3 Robustness

This study does not utilize real-world datasets in the evaluation. Instead, noise is added to the benchmark datasets to make them more representative of real-world data and make the results more generalizable. When constructing machine learning algorithms, it is essential to consider accuracy and robustness. Many machine learning algorithms rely on assumptions on the data. Both supervised and semi-supervised learning rely on empirical risk minimization [82], which means that all instances in the training set is from the same distribution. Therefore there is no guarantee that the trained algorithm generalizes well on data that is out of distribution (OOD). There are many different measurements for evaluating the robustness of an algorithm, such as *robustness measure* [83] and *coefficient of variation* [22]. The results of this study demonstrate that noise may improve the performance of semi-supervised algorithms. More specifically, noise increases the probability of achieving error rate below 10% for all datatypes. Noise has previously been demonstrated to improve the performance of algorithms in other

Table 20 Odds ratios for fixed effects and noise parameters (audio data)

Parameter	OR mean	OR HPD low	OR HPD high	OR range
a_adamatch	0.469	0.077	3.310	3.233
a_crmatch	0.376	0.063	2.696	2.633
a_comatch	0.008	0.001	0.059	0.057
a_dash	0.039	0.006	0.292	0.285
a_fixmatch	0.206	0.033	1.490	1.457
a_flexmatch	0.585	0.100	4.030	3.929
a_freematch	0.097	0.015	0.727	0.711
a_meanteacher	0.110	0.017	0.825	0.808
a_mixmatch	0.004	0.001	0.027	0.026
a_pimodel	0.001	0.000	0.009	0.009
a_pseudolabel	0.079	0.013	0.586	0.573
a_remixmatch	0.000	0.000	0.001	0.001
a_simmatch	0.700	0.124	4.653	4.529
a_softmatch	0.862	0.156	5.896	5.741
a_supervised	0.034	0.006	0.235	0.229
a_uda	0.036	0.006	0.255	0.249
a_vat	0.210	0.035	1.614	1.579
b_adamatch	0.914	0.618	1.344	0.726
b_crmatch	0.996	0.670	1.491	0.821
b_comatch	0.919	0.636	1.323	0.688
b_dash	1.025	0.659	1.601	0.942
b_fixmatch	0.987	0.634	1.538	0.903
b_flexmatch	0.999	0.698	1.430	0.731
b_freematch	1.126	0.703	1.795	1.093
b_meanteacher	0.894	0.573	1.372	0.798
b_mixmatch	1.178	0.799	1.743	0.944
b_pimodel	1.166	0.664	2.096	1.432
b_pseudolabel	0.993	0.654	1.505	0.852
b_remixmatch	0.003	0.000	3.586	3.586
b_simmatch	0.932	0.660	1.308	0.648
b_softmatch	0.877	0.625	1.220	0.595
b_supervised	1.019	0.688	1.506	0.818
b_uda	0.887	0.589	1.350	0.760
b_vat	0.974	0.627	1.520	0.893

studies [18, 22, 84]. In [22], three deep learning algorithms are evaluated across image datasets and their performance is compared between clean data and perturbed data. The results demonstrate that perturbed data improves robustness, and error rate in many cases [22]. Similarly, [84] compares the error rate of machine learning algorithms evaluated on text datasets where the amount of noise added to the samples varies between 0–100%. The results demonstrate that adding noise up to 40% will leave the error rate unchanged, and adding noise up to 70% will only increase the error rate slightly. Since noise may degrade or increase performance, it is necessary to investigate the impact of noise on the probability of successfully obtaining an error rate of 10% or less by simulating accuracies as described in Sect. 3. The results of this study demonstrate that noise increases the probability of successfully achieving error rate below 10% for all datatypes, which is confirmed by [18, 22].

5.4 Comparison with the original evaluation

This paper evaluates two additional algorithms SoftMatch and FreeMatch that have been added to USB since the original paper evaluation [19]. This paper utilizes Bayesian modeling, which has many advantages and provides a more fair evaluation due to the many benefits of Bayesian analysis.

Table 21 Odds ratios for fixed effects and noise parameters (small allocation of labels)

Parameter	OR mean	OR HPD low	OR HPD high	OR range
a_adamatch	0.047	0.012	0.196	0.185
a_crmatch	0.036	0.009	0.151	0.142
a_comatch	0.028	0.007	0.121	0.114
a_dash	0.024	0.006	0.104	0.098
a_fixmatch	0.034	0.008	0.142	0.133
a_flexmatch	0.036	0.009	0.146	0.138
a_freematch	0.031	0.008	0.130	0.123
a_meanteacher	0.017	0.004	0.073	0.069
a_mixmatch	0.004	0.001	0.017	0.016
a_pimodel	0.000	0.000	0.000	0.000
a_pseudolabel	0.012	0.003	0.049	0.046
a_remixmatch	0.005	0.001	0.024	0.023
a_simmatch	0.052	0.013	0.223	0.210
a_softmatch	0.061	0.015	0.255	0.240
a_supervised	0.010	0.002	0.044	0.042
a_uda	0.004	0.001	0.021	0.020
a_vat	0.019	0.004	0.079	0.074
b_adamatch	0.912	0.715	1.167	0.452
b_crmatch	0.961	0.749	1.230	0.481
b_comatch	0.940	0.727	1.212	0.485
b_dash	0.970	0.744	1.261	0.517
b_fixmatch	0.965	0.751	1.233	0.482
b_flexmatch	0.995	0.773	1.288	0.515
b_freematch	0.995	0.771	1.281	0.510
b_meanteacher	1.045	0.802	1.367	0.564
b_mixmatch	1.001	0.693	1.459	0.766
b_pimodel	4.007	0.669	53.703	53.034
b_pseudolabel	0.995	0.748	1.331	0.582
b_remixmatch	0.930	0.655	1.312	0.657
b_simmatch	1.044	0.824	1.330	0.507
b_softmatch	1.008	0.794	1.270	0.476
b_supervised	1.031	0.765	1.403	0.638
b_uda	0.994	0.700	1.408	0.709
b_vat	0.897	0.679	1.186	0.507

The Bayesian Bradley–Terry ranks obtained in this study differ from the Friedman Ranks obtained in the original paper [19]. The Bayesian Bradley–Terry model accounts for the uncertainty that is associated with the variation the error rate. Therefore, this study provides less misleading ranks than [19]. The original paper does not compare aggregated data or consider the number of available labels [19]. It only considers datatypes and does not give practitioners an idea of how many labels are required to achieve the lowest possible error rate.

The Friedman ranks obtained from the original study [19] are located in Table 25. The original study had CRMatch in the top three highest ranking algorithms for images, but UDA replaced it in this study. The results of this study are the same as the original for text and audio. However, the rank order is different and more algorithms are top-ranked in this study because the Bradley–Terry ranks incorporate uncertainty. Therefore there are ties in this study. Friedman ranks do not incorporate uncertainty, so there are no ties in [19].

6 Threats to validity

This section discusses four types of threats to validity for simulation studies described in [85].

Table 22 Odds ratios for fixed effects and noise parameters (large allocation of labels)

Parameter	OR mean	OR HPD low	OR HPD high	OR range
a_adamatch	0.103	0.027	0.390	0.363
a_crmatch	0.090	0.023	0.339	0.316
a_comatch	0.042	0.011	0.162	0.151
a_dash	0.054	0.014	0.205	0.191
a_fixmatch	0.093	0.025	0.342	0.317
a_flexmatch	0.149	0.039	0.556	0.517
a_freematch	0.048	0.013	0.183	0.170
a_meanteacher	0.019	0.005	0.074	0.069
a_mixmatch	0.005	0.001	0.020	0.019
a_pimodel	0.009	0.002	0.034	0.031
a_pseudolabel	0.035	0.009	0.134	0.125
a_remixmatch	0.006	0.001	0.023	0.022
a_simmatch	0.073	0.020	0.274	0.254
a_softmatch	0.093	0.025	0.350	0.326
a_supervised	0.013	0.003	0.052	0.049
a_uda	0.018	0.005	0.068	0.064
a_vat	0.031	0.008	0.115	0.107
b_adamatch	0.922	0.735	1.158	0.423
b_crmatch	1.053	0.828	1.337	0.510
b_comatch	0.972	0.776	1.217	0.441
b_dash	0.962	0.770	1.205	0.435
b_fixmatch	0.983	0.790	1.227	0.437
b_flexmatch	0.949	0.752	1.201	0.449
b_freematch	1.072	0.858	1.341	0.483
b_meanteacher	0.956	0.746	1.219	0.473
b_mixmatch	1.163	0.853	1.599	0.746
b_pimodel	1.043	0.783	1.393	0.610
b_pseudolabel	1.008	0.814	1.241	0.426
b_remixmatch	1.050	0.775	1.420	0.645
b_simmatch	0.981	0.779	1.229	0.450
b_softmatch	0.959	0.765	1.200	0.435
b_supervised	1.034	0.809	1.320	0.510
b_uda	0.936	0.720	1.217	0.496
b_vat	1.069	0.856	1.330	0.474

First is *Construct Validity*, which refers to how appropriate the statistical model is for answering the research questions. The RQs of this paper are concerned with ranking semi-supervised learning algorithms according to the lowest error rate. The Bayesian Bradley–Terry model was created for ranking and is appropriate for answering our RQs. The Bayesian Linear Regression model is utilized to calculate the probability of exceeding a certain threshold and is therefore appropriate to calculate the probability of an algorithm to achieve an error rate below 10%.

Second is *External Validity*, which refers to how generalizable the results are to other situations. In [19], the algorithms are evaluated on the same datasets. The algorithms have previously been evaluated in a similar way but with Friedman ranks in [19]. Thanks to the replication package, the results may be replicated and there is external validity.

Third is *Internal Validity*, which refers to whether the independent variables cause the outcome because simplification was made in the machine learning model or because some factors were not accounted for. In this study, no factors were omitted or any simplification in the models were made. Thus internal validity is ensured.

The final threat is *Conclusion Validity*, which refers to whether the results were evaluated utilizing appropriate statistical tests. In the study, posterior predictive checks are performed and the number of efficient examples and

Table 23 Summary of algorithms with odds ration higher than one

	Aggrgated	Image	Text	Audio	Small allocation of labels	Large allocation of labels
Π-model	YES	YES	NO	YES	YES	YES
Mean-Teacher	NO	NO	YES	NO	YES	NO
Pseudo-Label	YES	NO	NO	NO	NO	YES
VAT	NO	NO	YES	NO	NO	NO
MixMatch	YES	NO	NO	YES	YES	YES
ReMixMatch	NO	NO	NO	NO	NO	YES
UDA	NO	YES	NO	NO	NO	NO
FixMatch	PN	NO	NO	NO	NO	NO
FlexMatch	NO	YES	NO	NO	NO	NO
Dash	NO	NO	NO	YES	NO	NO
AdaMatch	NO	NO	NO	NO	NO	NO
CRMatch	YES	YES	NO	NO	NO	YES
CoMatch	NO	N0	YES	NO	NO	NO
SimMatch	YES	YES	NO	NO	YES	NO
SoftMatch	NO	NO	YES	NO	YES	NO
FreeMatch	YES	YES	YES	YES	NO	YES
Supervised	YES	YES	YES	NO	YES	YES

Table 24 Recommended Algorithms

Algorithm	Datatype	Comment
FreeMatch	All, large allocation of labels	In the top-3 highest ranking algorithms for every scenario
		Has OR $>$ 1 in the presence of noise for every scenario except for small allocation of labels
SimMatch	Images, small allocation of labels	Ties spot as highest ranking algorithm with FreeMatch
		Has OR $>$ 1 in the presence of noise for images and small allocation of labels
SoftMatch	Text	Second-highest ranking algorithm
		Has OR $>$ 1 in the presence of noise for text and small allocation of labels

Table 25 Top-three highest ranking algorithms for the original study [19]

Scenario	1st	2nd	3rd
Image	ReMixMatch	CRMatch	AdaMatch
Text	SimMatch	CRMatch	CoMatch
Audio	AdaMatch	SimMatch	FixMatch

Gelman-Rubin potential scale reduction are interpreted [86] to evaluate the results.

7 Conclusion

This study analyzes deep semi-supervised learning algorithms and presents a framework for what algorithms to utilize in industrial situations to obtain a certain error rate. The study provides an updated evaluation of USB that includes more algorithms added since the first evaluation [19]. In addition, this study investigates the impact of noise to understand how the algorithms will perform on real-world datasets.

According to the results, none of the algorithms have an error rate below 10%. The original simulations [19] were run utilizing different baseline deep learning models, and different hyperparameters were utilized for each task and varied across algorithms. It is possible to achieve a lower error rate if different supervised baselines are utilized and the hyperparameters are appropriately set for a given dataset. A takeaway is that many of the semi-supervised learning algorithms outperform supervised learning, and therefore, utilizing unlabeled data is relevant to improving the error rate. In addition, the results also demonstrate that the more labels are available, the better the error rate.

Generally, practitioners are recommended to investigate three algorithms: FreeMatch, SimMatch and SoftMatch. FreeMatch is recommended on all datatypes and for large

allocations of labels. SimMatch is recommended for utilize on image data and a small allocation of labels. Finally, it is recommended that SoftMatch be utilized for text data types and small allocation of labels. The algorithms are recommended for these scenarios since the results demonstrate the algorithms are the highest-ranking and perform well on real-world data.

The results of this study help machine learning specialists in industry and academia determine which algorithm will have the lowest error rate.

For future simulation studies, it is relevant to examine these algorithms utilizing other statistical models such as Bayesian regression [50] to answer related RQs and evaluate other types of semi-supervised learning algorithms not included in USB. Another interesting study is to utilize the item response theory to investigate if the 15 datasets are appropriate for evaluating USB.

Acknowledgements This work was partially supported by the Wallenberg AI Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation and by the Software Center.

Author Contributions Teodor collected all the data and perform the analysis and wrote the paper. Jan and Helena are the academic supervisor of PhD student Teodor and inspected the results and the paper to ensure high quality.

Funding Open access funding provided by Chalmers University of Technology.

Data availability The implementation and datasets is available at a publicly available repository https://github.com/teodorf-bit/Bayesian-Data-Analysis-of-Universal-Semi-Supervised-Benchmark.

Declarations

Conflict of interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Fredriksson, T., Mattos, D.I., Bosch, J., Olsson, H.H.: Data labeling: An empirical investigation into industrial challenges and mitigation strategies. In: International conference on product-focused software process improvement, pp. 202–216 (2020). Springer
2. Alenezi, H.S., Faisal, M.H.: Utilizing crowdsourcing and machine learning in education: Literature review. Educ. Inf. Technol. **25**(4), 2971–2986 (2020)
3. Chang, J.C., Amershi, S., Kamar, E.: Revolt: Collaborative crowdsourcing for labeling machine learning datasets. In: Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, pp. 2334–2346 (2017)
4. Zhang, J., Sheng, V.S., Li, T., Wu, X.: Improving crowdsourced label quality using noise correction. IEEE Trans. Neural Netw. Learn. Syst. **29**(5), 1675–1688 (2017)
5. Fredriksson., T., Bosch., J., Olsson., H.H.: machine learning models for automatic labeling: a systematic literature review. In: Proceedings of the 15th international conference on software technologies—volume 1: ICSOFT,, pp. 552–561. SciTePress, (2020). https://doi.org/10.5220/0009972705520561 . INSTICC
6. Shahshahani, B.M., Landgrebe, D.A.: The effect of unlabeled samples in reducing the small sample size problem and mitigating the hughes phenomenon. IEEE Trans. Geosci. Remote Sens. **32**(5), 1087–1095 (1994)
7. Nigam, K., McCallum, A.K., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using em. Mach. Learn. **39**, 103–134 (2000)
8. Baluja, S.: Probabilistic modeling for face orientation discrimination: Learning from labeled and unlabeled data. Advances in Neural Information Processing Systems **11** (1998)
9. Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Mach. Learn. **29**, 131–163 (1997)
10. Bruce, R.: Semi-supervised learning using prior probabilities and em. In: International joint conference on artificial intelligence, workshop on text learning: beyond supervision (2001)
11. Mahajan, D., Girshick, R., Ramanathan, V., He, K., Paluri, M., Li, Y., Bharambe, A., Van Der Maaten, L.: Exploring the limits of weakly supervised pretraining. In: Proceedings of the European Conference on Computer Vision (ECCV), pp. 181–196 (2018)
12. Hestness, J., Narang, S., Ardalani, N., Diamos, G., Jun, H., Kianinejad, H., Patwary, M.M.A., Yang, Y., Zhou, Y.: Deep learning scaling is predictable. Empirically **1712**, 2 (2017)
13. Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P.J.: Exploring the limits of transfer learning with a unified text-to-text transformer. J. Mach. Learn. Res. **21**(140), 1–67 (2020)
14. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al.: Language models are unsupervised multitask learners. OpenAI blog **1**(8), 9 (2019)
15. Xie, Q., Luong, M.-T., Hovy, E., Le, Q.V.: Self-training with noisy student improves imagenet classification. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 10687–10698 (2020)
16. Jozefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., Wu, Y.: Exploring the limits of language modeling. arXiv preprint arXiv:1602.02410 (2016)
17. Bartz-Beielstein, T., Doerr, C., Berg, D.v.d., Bossek, J., Chandrasekaran, S., Eftimov, T., Fischbach, A., Kerschke, P., La Cava, W., Lopez-Ibanez, M., et al.: Benchmarking in optimization: Best practice and open issues. arXiv preprint arXiv:2007.03488 (2020)
18. Braiek, H.B., Khomh, F.: Machine learning robustness: A primer. arXiv preprint arXiv:2404.00897 (2024)
19. Wang, Y., Chen, H., Fan, Y., Sun, W., Tao, R., Hou, W., Wang, R., Yang, L., Zhou, Z., Guo, L.-Z., et al.: Usb: a unified semi-supervised learning benchmark for classification. Adv. Neural. Inf. Process. Syst. **35**, 3938–3961 (2022)
20. Zhang, B., Wang, Y., Hou, W., Wu, H., Wang, J., Okumura, M., Shinozaki, T.: Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling. Adv. Neural. Inf. Process. Syst. **34**, 18408–18419 (2021)

21. Chen, H., Tao, R., Fan, Y., Wang, Y., Wang, J., Schiele, B., Xie, X., Raj, B., Savvides, M.: Softmatch: Addressing the quantity-quality trade-off in semi-supervised learning. arXiv preprint arXiv:2301.10921 (2023)
22. Dai, W., Berleant, D.: Benchmarking robustness of deep learning classifiers using two-factor perturbation. In: 2021 IEEE international conference on big Data (Big Data), pp. 5085–5094 (2021). IEEE
23. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. J. Am. Stat. Assoc. **32**(200), 675–701 (1937)
24. Cohen, J.: The earth is round (p< .05). Am. Psychol. **49**(12), 997 (1994)
25. Ioannidis, J.P.: Why most published research findings are false. PLoS Med. **2**(8), 124 (2005)
26. CloudFactory.com: The Ultimate Guide to Data Labeling for Machine Learning. (2019). https://www.cloudfactory.com/data-labeling-guide
27. Roh, Y., Heo, G., Whang, S.E.: A survey on data collection for machine learning: a big data-ai integration perspective. IEEE Trans. Knowledge Data Eng. **33**(4), 1328 (2019)
28. Crowston, K.: Amazon mechanical turk: A research tool for organizations and information systems scholars. In: Shaping the future of ICT research. Methods and approaches: IFIP WG 8.2, Working Conference, Tampa, FL, USA, December 13-14, 2012. Proceedings, pp. 210–221 (2012). Springer
29. Settles, B.: Active learning. morgan claypool. Synthesis Lectures on AI and ML (2012)
30. Zhu, X.J.: Semi-supervised learning literature survey. Technical report, University of Wisconsin-Madison Department of Computer Sciences (2005)
31. Subramanya, A., Talukdar, P.P.: Graph-based semi-supervised learning. Synthesis Lectures on Artificial Intelligence and Machine Learning **8**(4), 1–125 (2014)
32. Fredriksson, T., Mattos, D.I., Bosch, J., Olsson, H.H.: Assessing the suitability of semi-supervised learning datasets using item response theory. In: 2021 47th Euromicro conference on software engineering and advanced applications (SEAA), pp. 326–333 (2021). IEEE
33. Zagoruyko, S., Komodakis, N.: Wide residual networks. arXiv preprint arXiv:1605.07146 (2016)
34. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
35. Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: transformers for image recognition at scale. arXiv preprint arXiv:2010.11929 (2020)
36. Hsu, W.-N., Bolte, B., Tsai, Y.-H.H., Lakhotia, K., Salakhutdinov, R., Mohamed, A.: Hubert: Self-supervised speech representation learning by masked prediction of hidden units. IEEE/ACM Trans. Audio Speech Lang. Process. **29**, 3451–3460 (2021)
37. Lee, D.-H., et al.: Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In: Workshop on Challenges in Representation Learning, ICML, vol. 3, p. 896 (2013). Atlanta
38. Rasmus, A., Berglund, M., Honkala, M., Valpola, H., Raiko, T.: Semi-supervised learning with ladder networks. Advances in neural information processing systems **28** (2015)
39. Tarvainen, A., Valpola, H.: Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. Advances in neural information processing systems **30** (2017)
40. Miyato, T., Maeda, S.-I., Koyama, M., Ishii, S.: Virtual adversarial training: a regularization method for supervised and semi-supervised learning. IEEE Trans. Pattern Anal. Mach. Intell. **41**(8), 1979–1993 (2018)
41. Berthelot, D., Carlini, N., Goodfellow, I., Papernot, N., Oliver, A., Raffel, C.A.: Mixmatch: A holistic approach to semi-supervised learning. Advances in neural information processing systems **32** (2019)
42. Berthelot, D., Carlini, N., Cubuk, E.D., Kurakin, A., Sohn, K., Zhang, H., Raffel, C.: Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring. arXiv preprint arXiv:1911.09785 (2019)
43. Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., Le, Q.V.: Unsupervised data augmentation for consistency training. arxiv 2019. arXiv preprint arXiv:1904.12848 (1904)
44. Sohn, K., Berthelot, D., Carlini, N., Zhang, Z., Zhang, H., Raffel, C.A., Cubuk, E.D., Kurakin, A., Li, C.-L.: Fixmatch: Simplifying semi-supervised learning with consistency and confidence. Adv. Neural. Inf. Process. Syst. **33**, 596–608 (2020)
45. Xu, Y., Shang, L., Ye, J., Qian, Q., Li, Y.-F., Sun, B., Li, H., Jin, R.: Dash: Semi-supervised learning with dynamic thresholding. In: International conference on machine learning, pp. 11525–11536 (2021). PMLR
46. Li, J., Xiong, C., Hoi, S.C.: Comatch: Semi-supervised learning with contrastive graph regularization. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 9475–9484 (2021)
47. Fan, Y., Kukleva, A., Dai, D., Schiele, B.: Revisiting consistency regularization for semi-supervised learning. Int. J. Comput. Vision **131**(3), 626–643 (2023)
48. Berthelot, D., Roelofs, R., Sohn, K., Carlini, N., Kurakin, A.: Adamatch: A unified approach to semi-supervised learning and domain adaptation. arXiv preprint arXiv:2106.04732 (2021)
49. Zheng, M., You, S., Huang, L., Wang, F., Qian, C., Xu, C.: Simmatch: Semi-supervised learning with similarity matching. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 14471–14481 (2022)
50. Mattos, D.I., Bosch, J., Olsson, H.H.: Statistical models for the analysis of optimization algorithms with benchmark functions. IEEE Trans. Evol. Comput. **25**(6), 1163 (2021)
51. Issa Mattos, D., Martins Silva Ramos, É.: Bayesian paired comparison with the bpcs package. Behavior Research Methods. 1–21 (2022)
52. Bradley, R.A., Terry, M.E.: Rank analysis of incomplete block designs: I: the method of paired comparisons. Biometrika **39**(3/4), 324–345 (1952)
53. Cattelan, M.: Models for paired comparison data: a review with emphasis on dependent data. Stat. Sci. **27**, 412–433 (2012)
54. Agresti, A.: Categorical Data Analysis. Wiley (2003)
55. Furia, C.A., Feldt, R., Torkar, R.: Bayesian data analysis in empirical software engineering research. IEEE Trans. Software Eng. **47**(9), 1786–1810 (2019)
56. Fredriksson, T., Mattos, D.I., Bosch, J., Olsson, H.H.: An empirical evaluation of algorithms for data labeling. In: 2021 IEEE 45th annual computers, software, and applications conference (COMPSAC), pp. 201–209 (2021). IEEE
57. Robert, C.P., et al.: The Bayesian Choice: from Decision-theoretic Foundations to Computational Implementation. Springer (2007)
58. Yang, X., Song, Z., King, I., Xu, Z.: A survey on deep semi-supervised learning. IEEE Trans. Knowl. Data Eng. **35**(9), 8934 (2022)
59. Wang, Y., Chen, H., Heng, Q., Hou, W., Fan, Y., Wu, Z., Wang, J., Savvides, M., Shinozaki, T., Raj, B., et al.: Freematch: Self-adaptive thresholding for semi-supervised learning. arXiv preprint arXiv:2205.07246 (2022)
60. Krizhevsky, A., Hinton, G., et al.: Learning multiple layers of features from tiny images (2009)
61. Coates, A., Ng, A., Lee, H.: An analysis of single-layer networks in unsupervised feature learning. In: Proceedings of the fourteenth

international conference on artificial intelligence and statistics. pp. 215–223 (2011). JMLR Workshop and Conference Proceedings
62. Helber, P., Bischke, B., Dengel, A., Borth, D.: Eurosat: A novel dataset and deep learning benchmark for land use and land cover classification. IEEE J. Sel. Topics Appl. Earth Observ. Remote Sens. **12**(7), 2217–2226 (2019)
63. Helber, P., Bischke, B., Dengel, A., Borth, D.: 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, pp. 204–207 (2018). IEEE
64. Yang, J., Shi, R., Ni, B.: Medmnist classification decathlon: A lightweight automl benchmark for medical image analysis. In: 2021 IEEE 18th international symposium on biomedical imaging (ISBI), pp. 191–195 (2021). IEEE
65. Yang, J., Shi, R., Wei, D., Liu, Z., Zhao, L., Ke, B., Pfister, H., Ni, B.: Medmnist v2-a large-scale lightweight benchmark for 2d and 3d biomedical image classification. Sci. Data **10**(1), 41 (2023)
66. Su, J.-C., Maji, S.: The semi-supervised inaturalist-aves challenge at fgvc7 workshop. arXiv preprint arXiv:2103.06937 (2021)
67. Maas, A., Daly, R.E., Pham, P.T., Huang, D., Ng, A.Y., Potts, C.: Learning word vectors for sentiment analysis. In: Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies, pp. 142–150 (2011)
68. Zhang, X., Zhao, J., LeCun, Y.: Character-level convolutional networks for text classification. Advances in neural information processing systems **28** (2015)
69. McAuley, J., Leskovec, J.: Hidden factors and hidden topics: understanding rating dimensions with review text. In: Proceedings of the 7th ACM conference on recommender systems, pp. 165–172 (2013)
70. Chang, M.-W., Ratinov, L.-A., Roth, D., Srikumar, V.: Importance of semantic representation: dataless classification. Aaai **2**, 830–835 (2008)
71. Salamon, J., Jacoby, C., Bello, J.P.: A dataset and taxonomy for urban sound research. In: Proceedings of the 22nd ACM international conference on multimedia, pp. 1041–1044 (2014)
72. Asghar, N.: Yelp dataset challenge: review rating prediction. arXiv preprint arXiv:1605.05362 (2016)
73. Fonseca, E., Plakal, M., Ellis, D.P., Font, F., Favory, X., Serra, X.: Learning sound event classifiers from web audio with noisy labels. In: ICASSP 2019-2019 IEEE international conference on acoustics, speech and signal processing (ICASSP), pp. 21–25 (2019). IEEE
74. Yang, S.-w., Chi, P.-H., Chuang, Y.-S., Lai, C.-I.J., Lakhotia, K., Lin, Y.Y., Liu, A.T., Shi, J., Chang, X., Lin, G.-T., et al.: Superb: Speech processing universal performance benchmark. arXiv preprint arXiv:2105.01051 (2021)
75. Piczak, K.J.: Esc: Dataset for environmental sound classification. In: Proceedings of the 23rd ACM international conference on multimedia, pp. 1015–1018 (2015)
76. Bridle, J., Heading, A., MacKay, D.: Unsupervised classifiers, mutual information and'phantom targets. Advances in neural information processing systems **4** (1991)
77. Elworthy, D.: Does baum-welch re-estimation help taggers? arXiv preprint cmp-lg/9410012 (1994)
78. Cozman, F.G., Cohen, I., Cirelo, M.C., et al.: Semi-supervised learning of mixture models. In: ICML, vol. 4, p. 24 (2003)
79. Sajjadi, M., Javanmardi, M., Tasdizen, T.: Regularization with stochastic transformations and perturbations for deep semi-supervised learning. Advances in neural information processing systems **29** (2016)
80. Zhang, H.: mixup: Beyond empirical risk minimization. arXiv preprint arXiv:1710.09412 (2017)
81. Cardoso, L.F., Santos, V.C., Francês, R.S.K., Prudêncio, R.B., Alves, R.C.: Decoding machine learning benchmarks. In: Brazilian conference on intelligent systems, pp. 412–425 (2020). Springer
82. Shalev-Shwartz, S., Ben-David, S.: Understanding Machine Learning: From Theory to Algorithms. Cambridge University Press (2014)
83. Laugros, A., Caplier, A., Ospici, M.: Are adversarial robustness and common perturbation robustness independant attributes? In: Proceedings of the IEEE/CVF international conference on computer vision workshops, pp. 0–0 (2019)
84. Agarwal, S., Godbole, S., Punjani, D., Roy, S.: How much noise is too much: A study in automatic text classification. In: Seventh IEEE international conference on data mining (ICDM 2007), pp. 3–12 (2007). IEEE
85. França, B.B.N., Travassos, G.: Simulation based studies in software engineering: A matter of validity. In: CIbSE, pp. 308–321 (2014)
86. Mattos, D.I., Bosch, J., Olsson, H.H.: Statistical models for the analysis of optimization algorithms with benchmark functions. arXiv preprint arXiv:2010.03783 (2020)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.