Smart Agricultural Technology 9 (2024) 100613

Contents lists available at ScienceDirect

Smart Agricultural Technology

journal homepage: www.journals.elsevier.com/smart-agricultural-technology

Consistency regularization based semi-supervised plant disease recognition

Murat Ilsever *, Ipek Baz

Department of Electrical and Electronics Engineering, Engineering Faculty, Yeditepe University, Istanbul, Turkey

ARTICLE INFO

Keywords:
Deep learning
Convolutional neural networks
Plant disease recognition
Fine-grained classification
Semi-supervised learning

ABSTRACT

Plant diseases in agriculture are one of the most important factors affecting the quality and efficiency of production. Autonomous early detection of diseases in plants allows the necessary pesticide work to be carried out on time. Such an autonomous system can minimize production losses and improve product quality. This study highlights the data collection and labeling challenges in the field of plant diseases and proposes to solve the plant disease recognition problem with semi-supervised learning. To this end, we first evaluate three open datasets in the literature and settle on one. Then, numerous experiments are conducted to show that the supervised learning solution to the plant disease recognition problem is ineffective with limited labels. With these experiments, the effects of factors are measured, including limited labeled data, dataset balance, batch size, different fine-tuning strategies, and different levels of input augmentation in supervised learning. Finally, semi-supervised learning experiments showed an improvement in accuracy of up to 5.52 % over supervised learning.

1. Introduction

Due to the increasing growth of the world's population, scientists and agricultural producers have shifted their attention to supplying human demands for food. Based on the report of United Nations Food and Agriculture Organization, world's population will reach 9.1 billion by 2050. Therefore, the rate of food growth should be increased to meet the nutritional needs of people [1]. Agriculture requires the development of new and environmentally beneficial solutions. These solutions require automation in agriculture.

In agriculture, plant diseases are one of the most crucial concerns. They cause a decrease in both the quality and yield of agricultural products. Detecting diseases with the naked eye is problematic and costly when scanning large fields, since it requires expertise and a lot of time. Some plant diseases on various plant leaves are presented in Fig. 1. Identification of plant diseases and deficiencies is also not as accurate as automatic identification [2]. Furthermore, early detection of crop diseases is crucial since it increases food production rates, while also protecting farmers from costly spray processes [1]. Manual solutions are prohibitively expensive in developing nations; thus, researchers must propose effective low-cost procedures that are accessible to all farmers.

When plants are under abiotic stress or suffer from disease, visual changes can be seen in morphological, physiological, and biochemical ways. This is used by image-based analysis techniques to detect plant nutrient deficiencies and diseases. The study in [3] has shown that phenotypic changes in leaf such as color, leaf texture, and rate of light reflection can be detected using different imaging systems. Studies in the literature show that, sensors used in plant imaging can be examined in five different classes: (1) RGB imaging [4], (2) spectroscopy imaging [5], (3) fluorescence imaging [6], (4) thermal imaging [7], and (5) three-dimensional imaging [8].

Recent advances in plant disease recognition have seen a shift toward leveraging computer vision techniques and deep learning methodologies, particularly Convolutional Neural Networks (CNNs), to expedite and enhance accuracy in identifying various plant diseases. Mohanty et al. [9] trained a CNN on the Plant Village dataset of healthy and diseased plant leave images acquired in controlled environments achieving over 99 % accuracy. Sladojevic et al. [10] trained a model which is able to recognize 13 different types of plant diseases out of healthy leaves. They described all steps required for implementing the recognition model including data collection, data preprocessing and augmentation, and network training, Proposed model achieved an average of 96.3 % accuracy. Goncharov et al. [11] aimed to identify an optimal deep learning architecture by evaluating various models on the Plant Village dataset. The authors identified the negative impact of the synthetic nature of the Plant Village dataset images on the performance of the trained model on real-world images. To address this limitation, they created a specialized database focusing on grape leaves, comprising four image sets. They developed a deep Siamese convolutional network specifically to tackle the challenge of small data size. Kerkech et al. [12]

* Corresponding author.
E-mail addresses: murat.ilsever@std.yeditepe.edu.tr (M. Ilsever), ipek.baz@yeditepe.edu.tr (I. Baz).

https://doi.org/10.1016/j.atech.2024.100613
Received 16 July 2024; Received in revised form 18 October 2024; Accepted 20 October 2024
Available online 21 October 2024

Fig. 1. Three different plant diseases on the leaves of three plant species. Each image is from a different dataset. Left: Frog eye leaf spot on apple leaf (Plant Pathology 2021 dataset), Middle: Leaf blight on corn leaf (Plant Doc dataset), Right: Bacterial spot on tomato leaf (Plant Village dataset).

proposed a method to identify diseased areas in a vineyard. Their method employs a deep segmentation approach on Unmanned Aerial Vehicle (UAV) images, combining visible and infrared data from different sensors. Trained model classified pixels into categories such as shadow, ground, healthy, and symptomatic, achieving 92 % accuracy at the grapevine level and 87 % accuracy at leaf level. Although these studies have collectively highlighted the potential and effectiveness of deep learning techniques, they have not addressed the fact that model training requires a large amount of labeled data. In this paper, we investigate how to improve model performance using limited labeled data.

Plant disease recognition is qualified as a fine-grained classification task due to its focus on discerning subtle visual discrepancies between various diseases. The field of fine-grained visual classification in computer vision, particularly in the context of deep learning, has been a subject of significant interest. Two primary approaches have emerged: part-based and end-to-end feature encoding methods.

Earlier part-based methods utilized traditional object detectors and segmentation models thus heavily relied on additional part annotations to locate semantic key parts of the objects [13–15]. However, the need for part annotations has become an obstacle that limits the scalability and practicality of the fine-grained applications. Consequently, there emerged a need to accurately locate fine-grained parts using only image-level labels. These approaches, known as “weakly-supervised”, relied on unsupervised approaches to extract semantic groups that correspond to different parts of the object [16–18].

Another line of work focused on improving representational capacity of the model to aid fine-grained classification through end-to-end feature coding. One subgroup of this line concentrated on high-order feature interactions. Bilinear CNN [19] achieved good results by integrating the covariance matrix-based representation with deep descriptors. Gao et al. [20] proposed a compact bilinear pooling method to approximate second-order statistics of the original bilinear pooling operation while simultaneously reducing the feature dimensions. Other researchers [21–23] proposed methods to reduce the output size and the computation time of bilinear pooling. Another subgroup of end-to-end feature learning methods has attempted to build specific loss functions that contribute to the solution of the fine-grained classification problem. In fine-grained classification the similarities between classes are large, so it makes sense to position the classifier to produce high entropy outputs (i.e. not too confident predictions). Motivated by this idea, Dubey et al. [24] proposed to train their networks in such a way as to maximize the entropy of the output probability distribution. Similarly, Dubey et al. [25] employed a bidirectional confusion optimization approach to address overfitting and mitigate sample-specific artifacts in fine-grained recognition. Their method involves approximating different class conditional probability distributions and inducing confusion in the deep network.

The challenge in fine-grained classification lies in effectively discriminating between classes with subtle differences while accommodating high intra-class variance. This implies that fine-grained models need to be trained with large amounts of labeled data. In this study, instead of directly applying a fine-grained classification method, we will investigate how effectively our model can solve the fine-grained classification problem under semi-supervised learning (SSL) conditions (i.e. small amount of labeled data).

Due to the complexity, diversity, and variability of plant diseases, building a high-quality plant disease dataset is difficult: (1) building a dataset requires consulting many experts in different fields for annotation, (2) collecting plant disease images is extremely limited by the time of year and location of the crop [26]. In these respects, plant disease recognition is particularly suitable for SSL due to its capacity to diminish the data dependency within the training process. Deep SSL methods, which integrate SSL techniques into deep neural networks, fall into three primary categories: generative methods, consistency regularization methods, and pseudo-labeling methods. Generative methods employ techniques such as modifying generative adversarial network (GAN) frameworks, introducing new objective functions, or integrating inference networks to handle unlabeled data and enhance semi-supervised classification [27–29]. Consistency regularization methods aim to optimize model consistency between predictions on original and augmented samples, leveraging manifold or smoothness assumptions in SSL [30–32]. Pseudo-labeling methods like co-training frameworks and self-training rely on confident model predictions to label unlabeled data for further model training, expanding the dataset, and improving the performance in SSL scenarios [33–35].

Image recognition can be categorized into three main groups depending on the level of supervision provided during the learning process: supervised, semi-supervised, and unsupervised. Semi-supervised learning offers a powerful approach for plant disease recognition, bridging the gap between supervised and unsupervised methods. Unlike supervised learning, which demands large amounts of labeled data, semi-supervised learning leverages a small set of labeled images alongside a larger set of unlabeled ones, significantly reducing the need for expert annotation. This is particularly beneficial in plant disease detection, where obtaining labeled data is challenging. On the other hand, unsupervised learning approaches are ill-suited to this problem, as disease type discovery could overcomplicate the problem by generating less interpretable or less accurate groupings. Schmarje et al. [36], in their study on semi-, self- and unsupervised learning methods, highlight the superiority of semi-supervised methods for datasets with a few and fixed number of classes, which is in line with the dataset and results of our study. They also investigate the degree of supervision needed to achieve comparable results to fully supervised learning, and report that state-of-the-art semi-supervised methods leave a performance gap of less than 5 % compared to full supervision.

Recent research on plant leaf disease detection has predominantly employed fully supervised learning approaches. Sharma et al. [37] provide a comprehensive review of those methods. Table 1 presents studies that utilize deep semi-supervised learning approaches, highlighting the position of our research within the broader literature. Notably, two recent studies focus on object detection for foliar diseases in maize, aiming to detect lesion location and disease type. Given that the noisy student method falls under pseudo-labeling, it becomes

Table 1
Recent studies on semi-supervised plant leaf disease recognition.

References	Semi-supervised Method	Task	Dataset
Liu et al. [38] (2024)	Mean-teacher	Object Detection	Plant Village (Maize) + Self-collected
Chen et al. [39] (2024)	Mean-teacher	Object detection	Plant Village (Maize)
Sharma et al. [37] (2023)	Pseudo-labeling	Classification	Plant Village
Tao et al. [40] (2022)	GAN	Classification	Plant Village (Tomato)
Li et al. [41] (2021)	Pseudo-labeling	Classification	Plant Village
Keh [42] (2020)	Noisy Student	Classification	Plant Pathology 2020
Our study	Mean-teacher	Classification	Plant Pathology 2021

evident that pseudo-labeling is a widely adopted semi-supervised approach for classification tasks. In our study, we explore the mean-teacher method as a semi-supervised alternative. While most research leverages the Plant Village dataset, our decision to use the Plant Pathology 2021 dataset, instead of Plant Village, is detailed in Section 2.3. Plant Pathology 2020 dataset, included in Keh's study, is a more limited version of the 2021 dataset. Furthermore, unlike other studies, our work explicitly addresses the performance degradation of deep learning models when trained on limited data. We demonstrate how semi-supervised learning, by leveraging unlabeled data, mitigates this performance decline.

Mean-teacher [31], a widely used consistency regularization method, is adopted as the semi-supervised learning method. The proposed work diverges from the original mean-teacher paper in terms of (1) use of limited and unbalanced data, (2) the plant disease detection problem being a fine-grained classification problem, and (3) use of semi-supervised training during fine-tuning of pretrained networks instead of scratch training. Details and comparative evaluation of three relatively mature datasets from the literature are provided. Three strategies that are used in fine-tuning our backbone network are discussed, and the impact of these strategies in the final performance is shown. The supervised training performance of our backbone, considering several factors such as data size, dataset balance, batch size, and fine-tuning strategy, is intensively investigated to build up a baseline for semi-supervised experiments. In addition to those factors, the performance under two different input augmentation regimes, namely weak and strong, is investigated as input noise is a major factor in consistency regularization. The negative impact of limited data on supervised learning is demonstrated by training five widely accepted deep architectures. The superiority of semi-supervised learning over supervised learning for plant disease recognition is demonstrated over different training set sizes. Moreover, comparative experiments with alternative semi-supervised learning methods are conducted to establish the position of the mean-teacher method within the set of semi-supervised learning approaches. Finally, we explore whether zero-shot learning can serve as an alternative to semi-supervised learning for plant disease recognition.

In summary, the proposed work has the following main contributions:

(1) The proposed work shows that the problem of recognition of plant disease can be effectively solved by semi-supervised learning instead of the prevailing supervised learning solution.
(2) Our solution is justified by citing the scarcity of data and the high cost of labeling in the recognition of plant disease. In connection with this, relatively mature datasets in the literature were reviewed and numerous experiments were conducted.
(3) To demonstrate that semi-supervised learning is an effective solution, a systematic review of various factors affecting supervised learning, particularly the scarcity of data, was presented.

The remainder of this study is organized as follows. The 'proposed work' presents the plant disease recognition method based on the mean-teacher model, reviews various plant disease datasets and introduces model fine-tuning strategies. Then, 'experimental results' presents the performance of the proposed work. Finally, the study is concluded, future research is addressed in 'conclusion'.

2. Proposed work

This work proposes a semi-supervised learning solution to plant disease recognition. Mean-teacher, a widely used consistency regularization method, is implemented and applied to address the problem of plant disease recognition. Section 2.1 introduces the mean-teacher model. Section 2.2 discusses the application of the mean-teacher model to plant leaf images. Section 2.3 introduces three mature plant disease datasets and provides a comparative evaluation. Section 2.4 introduces the fine-tuning strategies used in our work.

2.1. Mean teacher

It's been established that employing an ensemble of multiple neural networks typically results in superior predictions compared to a single network within the ensemble. This effect has also been indirectly utilized during the training of a single network using techniques like dropout [43] or drop connect [44]. These techniques concentrate training efforts on specific subsets of the network, effectively creating an implicit ensemble of trained sub-networks within the complete network.

Ensemble models seek consistency around the training points, either at the input level or at the intermediate representation level. Both depend on the classification loss computed on the labeled data. Since the classification loss is undefined for unlabeled samples in semi-supervised learning, to build an ensemble model, a consistency loss is defined that can be computed over the unlabeled data. Consistency loss forces the network to classify the same for the same input under various noise factors, such as (1) augmentations or added noise to the input, or (2) noise added to the network itself or turning off part of it.

Mean-teacher model develops on temporal ensembling [30] model, but there are clear differences between them. Temporal ensembling builds up an ensemble prediction from the network predictions and computes the consistency loss by comparing this ensemble prediction with the network's response to the input. Mean-teacher, on the other hand, uses a separate teacher model. The teacher model cannot be trained; it is just the exponential moving average (EMA) of the student models seen so far. The consistency loss is computed from the response of the student and teacher networks to the same input. One advantage of mean-teacher is that the teacher model evolves with each training iteration (weight update) in contrast to temporal ensembling in which it takes one epoch to fully update its ensemble prediction. This property of the mean-teacher also becomes valuable in our work, since it will be on pre-trained networks with high learning capacity, such as ResNET [45].

One training pass of the mean-teacher model is depicted in Fig. 2. A minibatch consists of a specific amount of labeled and unlabeled samples. The supervised loss (classification loss in Fig. 2) is the cross-entropy distance between the output of the student model and the one-hot label vector, computed only for the labeled samples of the minibatch as follows.

$$\mathscr{L}_{sup}(y,\hat{y}) = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C} y_{n,i}\log\left(\hat{y}_{n,i}\right)$$

N : number of images in the minibatch
C : number of classes
$y_{n,i}$: true label for class i for image n (one-hot encoded labels)
$\hat{y}_{n,i}$: student model prediction for class i for image n

Consistency loss, on the other hand, is the mean squared error (MSE) or Kullback-Leibler divergence (KL-div) between the outputs of the student and teacher models. Unsupervised loss (consistency loss) is computed, as follows, for all samples (labeled and unlabeled) of the minibatch, as it does not require labels.

$$\mathscr{L}_{unsup} = \mathscr{L}_{MSE} \text{ or } \mathscr{L}_{KL}$$

$$\mathscr{L}_{MSE}(\hat{y},\hat{y}') = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{C}\sum_{i=1}^{C}\left(\hat{y}_{n,i}-\hat{y}'_{n,i}\right)^2$$

$$\mathscr{L}_{KL}(\hat{y},\hat{y}') = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{C}\hat{y}_{n,i}\log\left(\frac{\hat{y}_{n,i}}{\hat{y}'_{n,i}}\right)$$

N : number of images in the minibatch
C : number of classes
$\hat{y}_{n,i}$: student model prediction for class i for image n
$\hat{y}'_{n,i}$: teacher model prediction for class i for image n

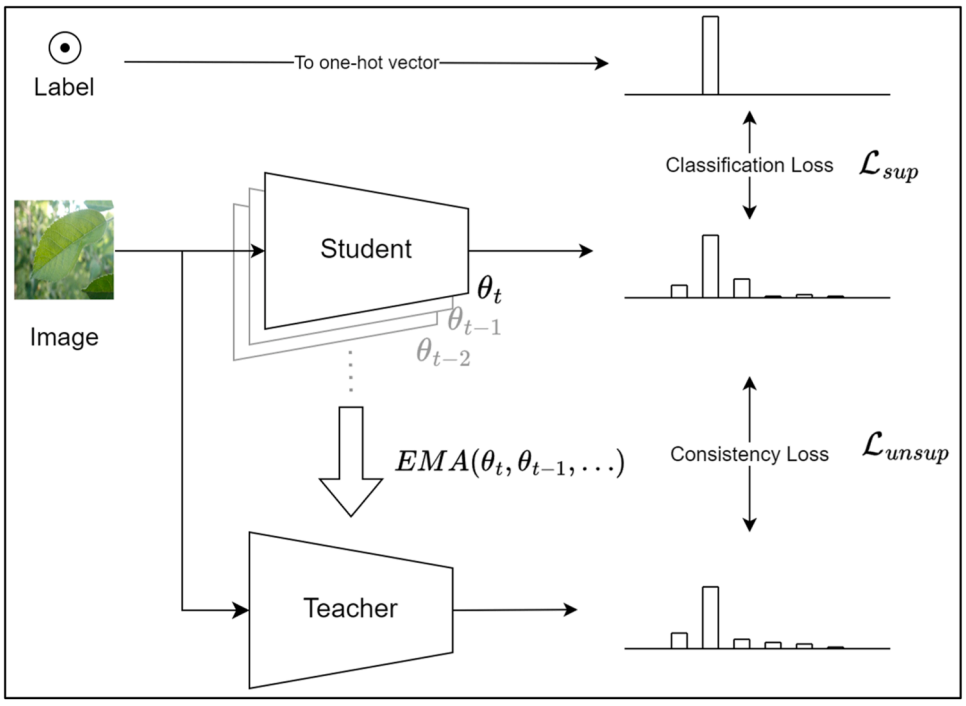

Fig. 2. One training pass of the mean-teacher model. Student model can be any deep architecture. ResNET-50 is adopted in our work. Teacher model parameters are exponential moving average (EMA) of Student model parameters (θ_t). Consistency loss, calculated at every training iteration, is the distance between the prediction distributions of Student and Teacher models.

The total loss is equal to the weighted sum of the supervised and unsupervised terms.

$$\mathscr{L}_{tot} = \mathscr{L}_{sup} + w(t) * \mathscr{L}_{unsup}$$

where, w(t) schedules the weight of the unsupervised loss. Student model parameters are updated with gradient descent applied to the total loss function. Teacher model parameters are the exponential moving average of the student model parameters seen earlier.

2.2. Application of the mean teacher model to plant leaf images

The plant pathology dataset used in this study will be discussed in the next section (Section 2.3). The preparation of the data for the experiments is discussed in Section 3.1. In this section, we describe in general how plant leaf images are used with the proposed method.

We followed the methodology outlined in Section 2.1 to train the mean-teacher model on plant leaf images. Fig. 3 illustrates the proposed approach in its simplest form. Each training iteration includes a mix of labeled and unlabeled leaf images. The blue arrows represent labeled

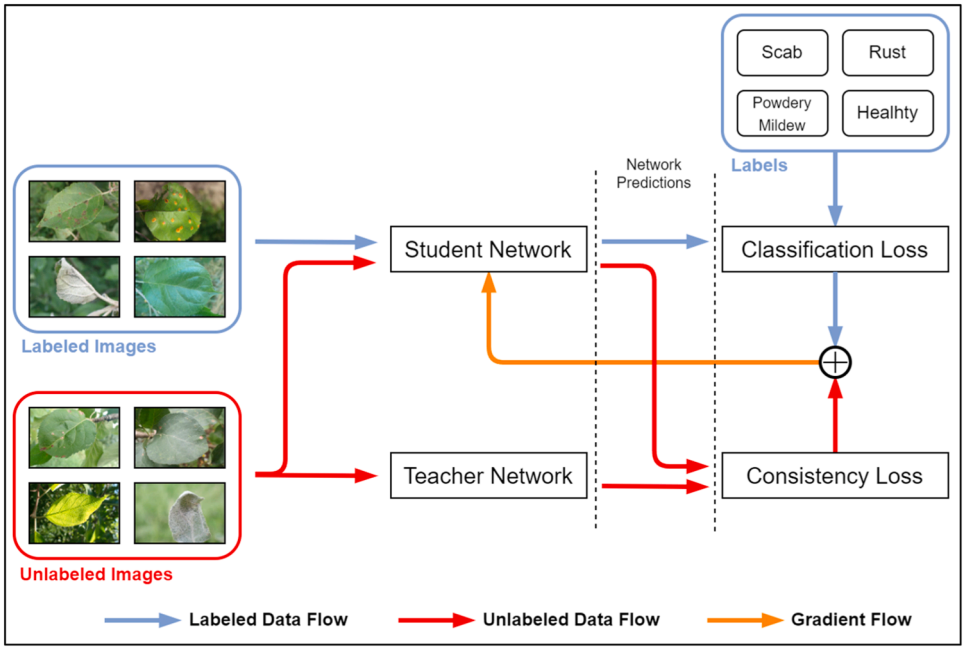

Fig. 3. Application of the mean-teacher model to plant leaf images.

data flow, and the red arrows represent unlabeled data flow. For labeled images, the classification loss is computed by comparing the student model's predictions to the ground truth labels. Simultaneously, for the unlabeled images, the consistency loss is calculated by comparing the student model's predictions to those of the teacher model. The combined loss from these two components is then backpropagated to update the student model's parameters (illustrated by the orange flow). In the same iteration, the teacher model is updated with the current student model, aiming to become an average of the recent student models.

Fig. 4 illustrates the training and test processes, alongside the detailed network architecture. The backbone of the network is the ResNet-50 model, comprising 25.5 million parameters. The output feature vector from this backbone is passed to a fully connected network (FCN), also referred to as the classifier head, whose details are provided in Section 2.4. The FCN produces logits, which are converted into prediction probabilities using the softmax function. These probabilities are then used to compute both classification and consistency losses, as previously discussed in Fig. 3.

During training (Fig. 4 top), only the student network is trainable. The student network is trained using the gradient signal from the total loss, which is back-propagated through the network. The training diagram also shows that image augmentation is applied to both labeled and unlabeled images. The student network's optimal parameters, learned during training, are carried over to the testing phase. During test (Fig. 4 bottom), labeled images are passed through the student network without augmentation. The class with the highest prediction probability is selected as the predicted label, and by comparing it to the ground truth, the network's accuracy on the test set is computed.

2.3. Plant disease datasets

Few datasets exist in the field of plant disease recognition, and the datasets found are often imbalanced and have few samples [46]. For this reason, it is an understudied area. Scarcity of data makes it a relevant application for semi-supervised learning. Research was conducted on comparing existing datasets in plant disease recognition and determining the dataset in which unlabeled data can be used most effectively. Below we investigate the 3 most mature datasets.

2.3.1. Plant village dataset

Plant Village contains 54,309 images. The images span 14 plant species and 26 plant diseases. Healthy leaf images for 12 plant species with no disease detected are also available. A total of 38 healthy and unhealthy classes exists. Table 2 shows the data distribution of the Plant Village dataset [47].

The dataset curators evaluate the applicability of deep convolutional

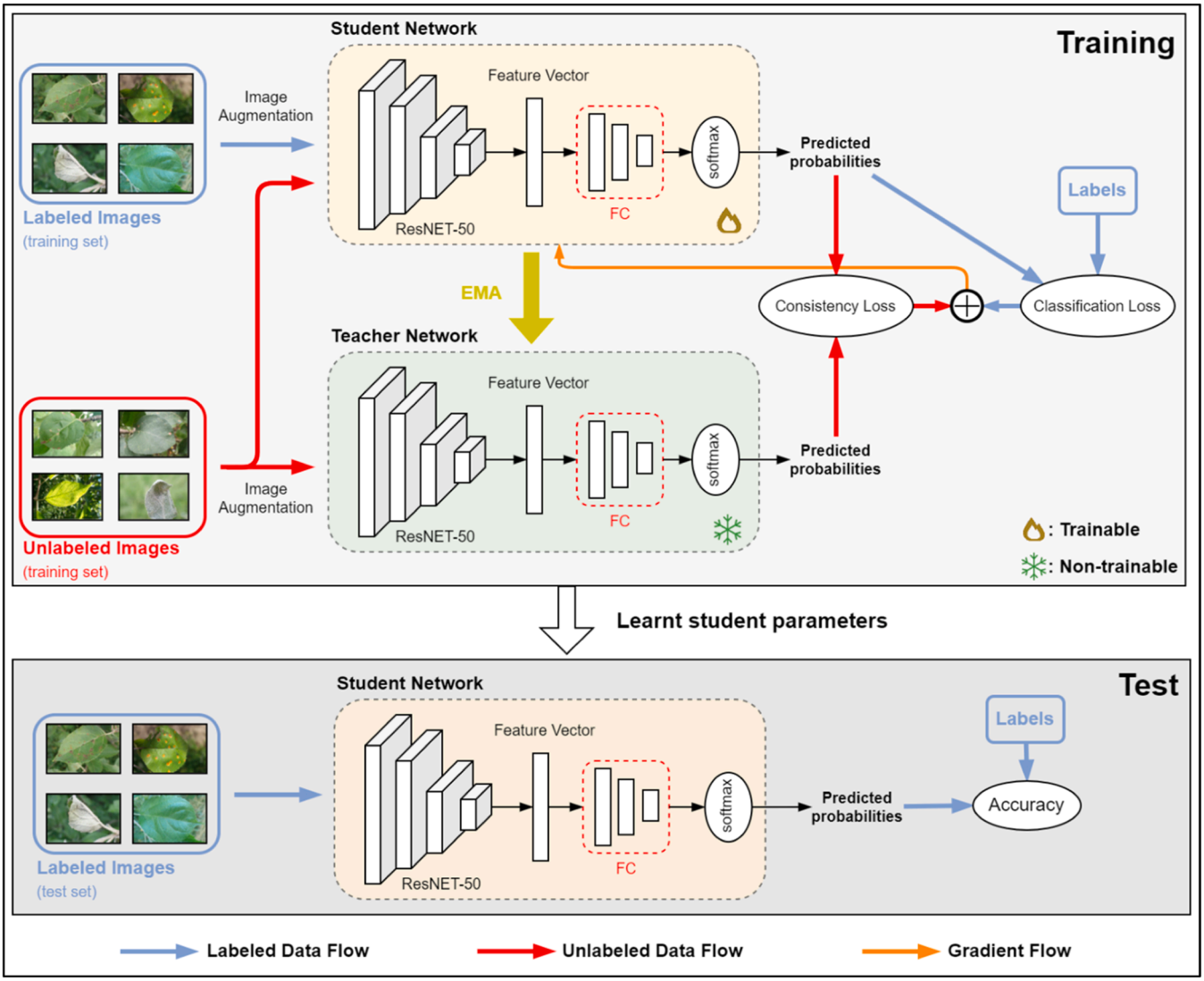

Fig. 4. Detailed network structure and training/test procedures. FC indicates fully connected layers.

Table 2
Plant Village data distribution.

Apple	**Orange**	**Squash**
Scab - 630	Huanglongbing - 5507	Powdery Mildew - 1835
Black Rot - 621	**Peach**	**Strawberry**
Cedar Apple Rust - 275	Bacterial Spot - 2297	Leaf Scorch - 1109
Healthy - 1645	Healthy - 360	Healthy - 456
Blueberry	**Bell Pepper**	**Tomato**
Healthy - 1502	Bacterial Spot - 997	Bacterial Spot - 2127
Cherry	Healthy - 1478	Early Blight - 1000
Powdery Mildew - 1502	**Potato**	Late Blight - 1909
Healthy - 854	Early Blight - 1000	Leaf Mold - 952
Corn	Late Blight - 1000	Septoria Leaf Spot - 1771
Gray Leaf Spot - 513	Healthy - 152	Spider Mites - 1676
Common Rust - 1192	**Raspberry**	Target Spots - 1404
Northern Leaf Blight - 985	Healthy - 371	Yellow Leaf Curl Virus - 5357
Healthy – 1162	**Soybean**	Mosaic Virus - 373
Grape	Healthy - 5090	Healthy - 1591
Black Rot - 1180		
Esca (Black measles) - 1383		
Leaf Blight - 1076		
Healthy - 423		

neural networks for plant disease classification [9]. They trained GoogLeNet and AlexNet, achieving an accuracy of 99.35 %. However, the images in the Plant Village dataset were captured in laboratory settings and are far from being representative of real cultivation field conditions (Fig. 5), thus the real-world performance of the models trained on this dataset is likely to be poor. In our experiments, the ResNET-50 architecture was fine-tuned using the HeadOnly strategy described in Section 2.4, and achieved 98.4 % accuracy in 90 epochs. It was surprising that such a high accuracy was achieved on such a large dataset with our simplest fine-tuning strategy. In Section 3.2, it will be shown that this is not the case with the Plant Pathology 2021 dataset. These results lead us to consider the capture bias inherent to the Plant Village dataset.

2.3.2. Plant Doc dataset

Plant Doc contains 2598 images across 13 plant species and 27 classes (17–10, disease-healthy). Table 3 shows the data distribution of the Plant Doc dataset [46]. Dataset curators claim that Plant Village data set will not be successful in real environments since it was prepared in a controlled environment. They collected plant leaf and disease images from the internet, then eliminated inappropriate images according to some rules.

There are two problems with the Plant Doc dataset. While the claim by the curators was that the images they collected are from real environments, a fair amount of images with unrealistic backgrounds, especially from the healthy classes, can still be found when the samples were examined (Fig. 6). The second problem is that the visual scope of the images is not uniform and shows a great deal of variation due samples are downloaded from the internet.

2.3.3. Plant pathology 2021 (PP2021)

PP2021 contains 18,632 training images for Apple leaf from 12 categories [48,49]. Test set of the Plant Pathology 2021 Challenge

Fig. 5. Samples from the Plant Village dataset. (Left: Apple, Middle: Cherry, Right: Orange).

Table 3
Plant Doc data distribution.

Apple	**Potato**
Rust - 89	Early Blight - 117
Scab - 87	Late Blight - 105
Healthy - 91	**Raspberry**
Bell Pepper	Healthy - 116
Leaf Spot - 71	**Soybean**
Healthy – 61	Healthy - 65
Blueberry	**Squash**
Healthy - 116	Powdery Mildew - 129
Cherry	**Strawberry**
Healthy - 57	Healthy - 96
Corn	**Tomato**
Gray Leaf Spot - 68	Early Blight - 83
Leaf Rust - 116	Late Blight - 111
Leaf Blight - 192	Leaf Mold - 91
Grape	Bacterial Leaf Spot - 107
Black Rot - 64	Septoria leaf spot - 148
Healthy - 69	Yellow Leaf Virus - 75
Peach	Mosaic Virus - 54
Healthy - 112	Healthy - 62

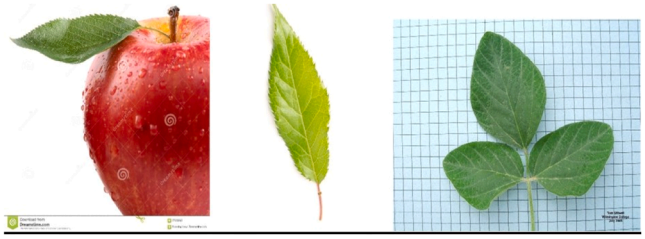

Fig. 6. Samples with unrealistic background from Plant Doc dataset. (Left: Apple leaf, Middle: Cherry leaf, Right: Soyabean leaf).

contains 5000 images but was hidden until end of the challenge and was provided only to the participants. The challenge targeted two classification tasks: Multi-Class classification and Multi-Label classification. Since our task is semi-supervised multi-class classification, we only give the classes and the data distribution for the multi-class case (Table 4). Complex class is explained in the challenge page as follows: "Unhealthy leaves with too many diseases to classify visually will have the complex class and may also have a subset of the diseases identified."

PP2021 is built on Plant Pathology 2020 dataset; increased the dataset size from 3651 to 23,249 and number of classes from 4 to 12. Thapa et al. [49], gives the details of image capturing process and dataset formation for PP2020; "We captured high-quality, real-life RGB images of multiple apple foliar disease symptoms during the 2019 growing season from commercially grown cultivars in an unsprayed apple orchard at Cornell AgriTech (Geneva, New York, USA). Photos were taken using a Canon Rebel T5i DSLR (Canon Inc., Tokyo, Japan) and smartphones under various illumination, angle, surface, and noise conditions. The complexities of the data set were increased by including (1) an imbalanced data set of different disease categories, (2) non-homogeneous image backgrounds, (3) images taken at different times of day, (4) images from plants at different maturity stages, (5)

Table 4
Plant Pathology 2021 data distribution for multi-class classification.

Foliar diseases	Total images	Training dataset	Test dataset
Scab	6034	4827	1207
Frogeye leaf spot	3979	3183	796
Ceder apple rust	2329	1863	466
Powdery mildew	1513	1210	303
Complex	3610	2888	722
Healthy	5784	4631	1158
Total	23,249	18,602	4652

images displaying multiple diseases, and (6) images taken using different focus settings. A majority of the pictures taken were of apple scab, cedar apple rust, Alternaria leaf spot, frogeye leaf spot, and healthy leaves." Fig. 7 shows samples from PP2021 dataset.

We evaluate 3 datasets based on the following criteria: being realistic, number of samples, number of classes, and sample/class balance. Plant Village dataset is not close to being realistic. It's the largest dataset among all, and it has a good sample/class balance (54,309/38); with this feature it's the closest to representing 38 classes best. However, capture bias, inherent to it, allows deep architectures learn simply discriminate between those 38 classes. Plant Doc dataset has the smallest number of samples and has a poor sample/class balance (2598/27). While it contains realistic images and has a large capture variation, whether this variation will translate into an autonomous system to be used in cultivation fields remains a question mark since the images are downloaded from the internet. Moreover, it cannot be said that this dataset is 100 % realistic. PP2021 dataset has the smallest number of classes (6), and it is a single crop (apple) dataset focusing on only foliar disease. Its sample/class balance is very good (18,602/6), having at least 1200 training samples per class. All samples are realistic, and visual scope of the captures are uniform (e.g., all images are taken from similar distance).

We would like to make our choice as a dataset with the best features of these 3 datasets, but we have no such luck. Considering the deep architecture to be used in our experiments (Resnet-50), the capture variation (image complexity) is important, and thus, the most important criterion is that the dataset be realistic. For this reason, the PP2021 dataset, which is regarded as the most realistic and has a good sample/class balance, is chosen. Conversely, the experiments with PP2021 will be conducted with a limited number of classes. We hope that our research will shed light on future research with a realistic dataset that supports more classes.

2.4. Fine-tuning strategies

Several pretrained deep architectures were fine-tuned on the PP2021 dataset. To fine-tune a pretrained model on a new dataset, the classifier head of the model is replaced with a new head compatible with the number of classes of the new dataset. Fig. 8 shows two different ways of this surgery applied to a deep model pretrained on ImageNET. Since the model is pretrained on ImageNET, initially it has 1000 units at its classifier head. This linear classifier can be replaced directly with a new linear layer having units equal to the number of classes of the new dataset. We will refer to this as "Softmax" head (Fig. 8 top). One other approach is replacing the classifier head with a Fully Connected Network (FCN) having the last linear layer as the softmax layer. We will refer to this as "FCN" head (Fig. 8 bottom). This is the preferred approach if the difference between the number of units of the softmax layer and the layer before it is large. In the case of ResNET-50 model, an input image is encoded into a 2048-dimensional feature vector. A single linear layer connects this feature vector to a 1000-unit softmax classifier. When the 1000-unit classifier is replaced with a 6-unit classifier (number of classes of PP2021 dataset), there appears a huge difference between the unit counts of 2048 and 6. So instead 1000-unit classifier is replaced with an FCN as shown in Fig. 8 bottom. Table 5 shows the structure of the FCN head used in our work.

Our fine-tuning framework supports three fine-tuning strategies:

Fig. 7. Samples from PP2021 dataset (Left: Healthy, Middle: Powdery Mildew, Right: Scab) (PP2021 includes a single crop: Apple).

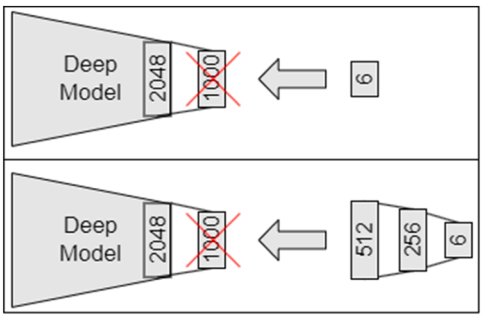

Fig. 8. Replacement of the classifier head. (Top: Softmax head, Bottom: FCN head).

Table 5
The structure of the Fully Connected Network (FCN) classifier used in our work. It is composed of 3 linear layers and 2 dropout modules in between. The dropout probabilities are equal to 0.5. The non-linear activations of the first 2 linear layers are obtained by the Rectified Linear Unit (ReLU) function.

Name	Description
input	D dimensional feature vector
dense1r	Fully connected D → 512, ReLU
drop1	Dropout, $p = 0.5$
dense2r	Fully connected 512 → 256, ReLU
drop2	Dropout, $p = 0.5$
dense3	Fully connected 256 → 6
output	Softmax

HeadOnly, HeadThenBody and HeadAndBody. While hyperparameter details will be reported in Section 3.2, a brief explanation of each strategy is given in this section. Any fine-tune model created initially has its classifier head parameters unfrozen (trainable), and its body (feature extractor) parameters frozen. The body of a model contains all parameters excluding its head. HeadOnly fine-tuning trains the classifier head with a normal learning rate from scratch. In the experiments, the FCN head with the structure given in Table 5 was trained. as part of this strategy. HeadThenBody strategy first trains the classifier head, then unfreezes the model's body and trains all model parameters with a small learning rate. Using a small learning rate is the preferred choice while fine-tuning. HeadAndBody strategy trains the head and fine-tunes the body at the same time. This strategy has not been implemented in our code, yet, and has never been used in our experiments. In the future implementation of this strategy, Pytorch's parameter groups will be used to set different learning rates for head and body. This way, the model's head and body can be trained and fine-tuned at the same training pass.

Fig. 9 illustrates HeadOnly and HeadThenBody strategies applied to supervised and mean-teacher trainings. For details of mean-teacher training refer to Fig. 2. HeadOnly training runs a single training loop. It trains the classifier head, and then stops. In mean-teacher training classifier of the student model is trained. All parameters of the teacher model including its classifier are frozen. HeadThenBody trains the classifier head as in the HeadOnly training, then starts a second training loop to train the body (feature extractor) of the model. HeadThenBody training is computationally more expensive than the HeadOnly training, mainly because it trains all model parameters in the second training loop. One way to reduce the training time is to start from a previous HeadOnly training result. Our fine-tuning framework allows HeadThenBody training to start directly from the body training by loading a previously recorded model via the "head-checkpoint" argument.

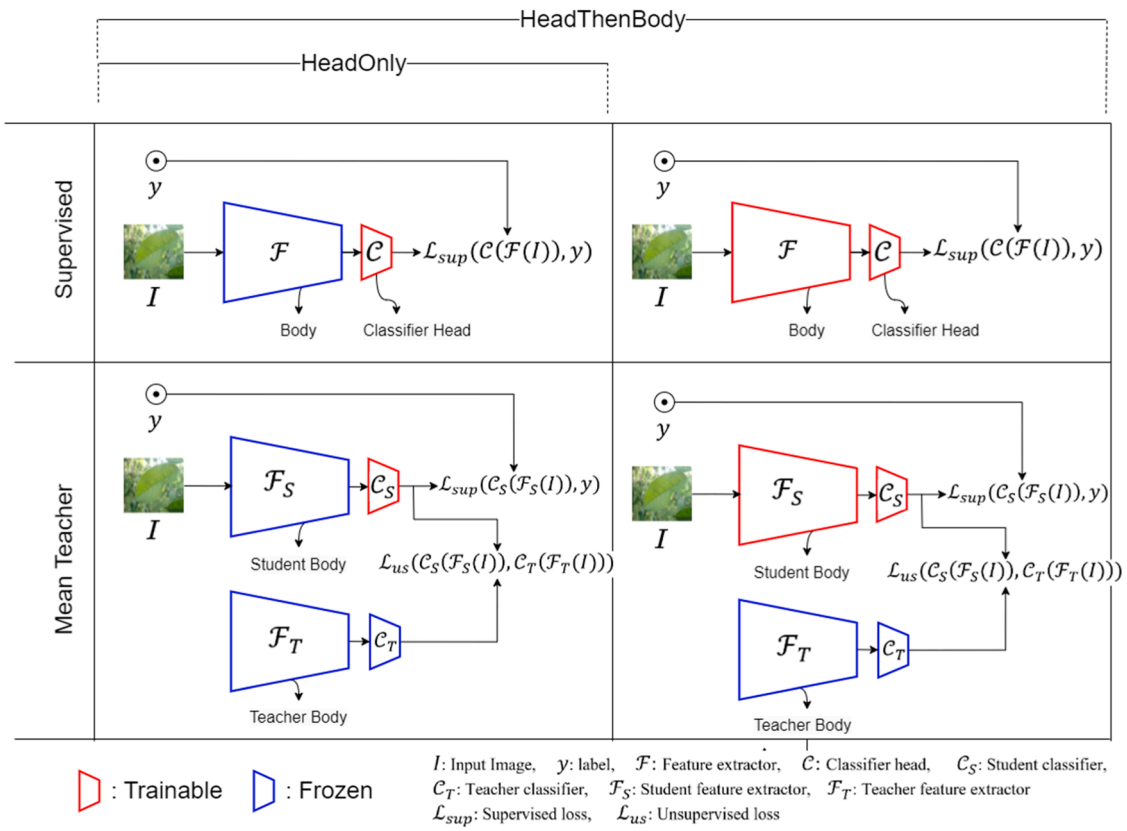

Fig. 9. Training procedure of HeadOnly and HeadThenBody strategies for supervised and mean-teacher trainings. HeadOnly trains **C** in the supervised training, and $\mathbf{C}_S$ in the mean-teacher training, then stops. HeadThenBody trains **C** and $\mathbf{C}_S$ as in the HeadOnly training, then starts a new training loop to train **F** in the supervised training and $\mathbf{F}_S$ in the mean-teacher training.

3. Experimental results

3.1. Preparation of PP2021 dataset

PP2021 Dataset is provided with the Plant Pathology 2021 Challenge [50]. It officially has 5000 test images, but access to this data is provided only to the participants of the challenge. For this reason, the training dataset is treated as a full dataset and split into training and validation sets. Stratified sampling is used as our splitting strategy to maintain the data distribution in the new training and validation splits. Only the single-label classes are kept (Scab, Frogeye leaf spot, Rust, Powdery mildew, Complex and Healthy), and the multi-label classes are omitted (Frog eye leaf spot/Complex, Powdery mildew/Complex, Rust/Complex, Rust/ Frog eye leaf spot, Scab/Frog eye leaf spot and Scab/Frog eye leaf spot/Complex).

The training set distribution of PP2021 diverges from the PP2021 report [48]. Table 6 compares the training set distribution obtained from the challenge page [50] with that provided in the report. The difference in the Complex class is particularly striking. Samples were added to the Complex class from multi-label classes, such as Rust/Complex or Powdery mildew/Complex, on the suspicion that this number was reached by adding samples from multi-label classes. This number could not be reached in any of our trials, so it is being kept as is.

The training and validation sets were prepared from a 90:10 stratified split of the training set of PP2021. A folder structure required by the torchvision.datasets.ImageFolder that used to read the samples into our fine-tuning framework was created (Fig. 10). From now on, we refer to this dataset as PP2021TS as its the training set split of PP2021. Class distribution of training and validation sets of PP2021TS is given in Table 7.

3.1.1. Balanced dataset

Experiments were conducted with a balanced dataset, in addition to the dataset prepared using all data. The under-sampling strategy was adopted to balance the dataset, reducing the number of samples in all categories to 1184, which matches the number of samples in the Powdery Mildew category (the category with the least number of samples). The dataset was split into training and validation sets using a 90:10 random split, resulting in 1065 training samples and 119 validation samples for each category.

Table 6
PP2021 training set distribution comparison (paper vs challenge page).

Foliar diseases	Training (paper)	Training (ours)
Scab	4827	4826
Frogeye leaf spot	3183	3181
Ceder apple rust	1863	1860
Powdery mildew	1210	1184
Complex	2888	1602
Healthy	4631	4624
Total	18,602	17,277

```
|-- train                    |-- val
    |-- complex                |-- complex
    |-- frog_eye_leaf_spot     |-- frog_eye_leaf_spot
    |-- healthy                |-- healthy
    |-- powdery_mildew         |-- powdery_mildew
    |-- rust                   |-- rust
    `-- scab                   `-- scab
```

Fig. 10. Image folder structure of PP2021TS dataset.

Table 7
Training and validation set distribution of PP2021TS.

Foliar diseases	Training	Validation
Scab	4343	483
Frogeye leaf spot	2862	319
Ceder apple rust	1674	186
Powdery mildew	1065	119
Complex	1441	161
Healthy	4161	463
Total	15,546	1731

3.1.2. Subsets of PP2021TS dataset

Sections 3.2 and 3.3 addresses performance of supervised and semi-supervised learning algorithms under low data regimes. Three low data settings are represented by the following percents of the training set of PP2021TS dataset: 25 %, 10 % and 5 %. Table 8 shows the number of images in each subset. Subsets are created via stratified sampling to preserve the class distribution of the full training set. Preservation of the class distribution is obvious in Fig. 11. Subsets were created with the same percents for the balanced dataset (Section 3.1.1) resulting in 266, 106 and 52 samples per class, for 25 %, 10 % and 5 % subsets, respectively.

3.2. Supervised training experiments

Intensive supervised training tests were conducted on the PP2021TS dataset before examining the effect of semi-supervised training with the mean-teacher model. Section 3.2.1 discusses the factors that adversely affect the supervised learning through the ResNET-50 model. Section 3.2.2 evaluates the classification performance of several deep architectures under limited data regime.

3.2.1. Supervised learning ablation study on ResNET-50

This section explores the impact of the following factors on supervised learning through the ResNET-50 model.

- Training set size
- Balanced/Unbalanced dataset
- Batch size
- Fine-tuning strategy

We give hyperparameters we kept constant in all our supervised experiments. Our fine-tuning framework is bult on Pytorch [51] API. The ResNET-50 model with 25.5 million parameters and pre-trained on ImageNet was used as the feature extractor. Classifier head of the network is the FCN head discussed in Section 2.4. Stochastic gradient descent was used with an initial learning rate of 0.01 for head training and 10^{-4} for body fine-tuning. Other SGD parameters, momentum and L2 weight decay are set to 0.9 and 10^{-4}, respectively. The network was trained for 90 epochs with early stopping. The early stopping patience was kept at 5 epochs in all experiments, allowing training to be stopped if no improvement in validation loss was observed within 5 epochs. The learning rate was decayed by a factor of 0.01 at 5/6 of 90 epochs (75) (Fig. 12) during head training and was kept constant (10^{-4}) during body fine-tuning. Table 9 lists all hyperparameters.

Table 8
Statistics for the training sets used in experiments: All, 25 %, 10 % and 5 %. Training subsets (25 %, 10 % and 5 %) are obtained via stratified sampling of the full training set with the given percent. The validation set of the full dataset is common to all training sets.

Class	All	25 %	10 %	5 %	Validation
Scab	4343	1085	434	217	483
Frogeye leaf spot	2862	715	286	143	319
Ceder apple rust	1674	418	167	83	186
Powdery mildew	1065	266	106	53	119
Complex	1441	360	144	72	161
Healthy	4161	1040	416	208	463
Total	15,546	3884	1553	776	1731

The top-1 accuracy of the ResNET-50 model under several training conditions is shown in Table 10. Details of the training sets are given in Section 3.1.2 and Table 8. Number seen after the @ sign indicates at which epoch was this accuracy achieved with early stopping. First, training is initiated with batch sizes of 16 and 32 for several data sizes (training sets) and balanced/unbalanced datasets. As expected, regardless of the batch size and dataset balance, top-1 accuracy drops when the number of samples is limited. Better accuracy is consistently produced by batch size 32 and unbalanced datasets. At this point, it was decided to fix those two factors (unbalanced dataset and batch size 32) and utilize them in all subsequent experiments.

Next, experiments were conducted to observe the impact of our second fine-tuning strategy: HeadThenBody. While fine-tuning with this strategy requires the training of the classification head, first, as it has already been trained in the HeadOnly experiment, training was resumed from this checkpoint and started from the fine-tuning the body stage. Again, as expected, network performance substantially increased at every data size. Top-1 accuracy increased by 6.3 %, 3.87 % and 5.55 % when trained with all labels, 25 % of the labels and 10 % of the labels, respectively.

3.2.1.1. Weak and strong augmentations. Temporal ensembling and mean-teacher papers (Section 2.1) emphasize the impact of the input augmentations on training with the unlabeled data. Performance of the network was examined under two different augmentation regimes: weak and strong. Those augmentations are from the mean-teacher paper. Augmentations, what we refer to as weak (Resize, RandomCrop, RandomHorizontalFlip), are applied to CIFAR-10 samples, and augmentations, what we refer to as strong (RandomRotation, RandomResizedCrop, RandomHorizontalFlip, ColorJitter), are applied to ImageNET samples in the mea-teacher paper. As our dataset is different, network performance is tested under both augmentations. Fig. 13 shows weak and strong augmentation examples.

Table 11 compares the performance under two augmentation regimes. One obvious trend is that under strong augmentations the network can not reach its previous performance with HeadOnly training. However, after the network body is started to be fine-tuned, accuracy reaches to those obtained under weak augmentations. Another finding is that under strong augmentations early stopping occurs less frequently. For those cases, it may be necessary to extend the training period.

3.2.2. Evaluation of different deep architectures

This section extends the experiments in Section 3.2.1 to other deep architectures. The performances of several deep architectures were measured with limited training sets and two fine-tuning strategies given the conditions of batch size 32 and unbalanced dataset determined in Section 3.2.1. We fine-tuned ResNET-50 [45], Inception V3 [52], DenseNET [53], ConvNext [54] and MobileNET V3 [55] models. All architectures are loaded with weights pretrained on ImageNet, and the FCN head is used for the classification task. Training hyperparameters are also the same as in Section 3.2.1 and are listed in Table 9. Model parameter counts are listed in Table 12.

Fig. 14 shows the best top-1 validation accuracy of the networks for different training set sizes (All, 25 %, 10 % and 5 %). Details of the training sets are given in Section 3.1.2 and Table 8. Numerical values of the figure are given in Tables 17 and 18. Square data markers indicate the HeadOnly fine-tuning results, and circular data markers indicate HeadThenBody fine-tuning results. The same trend observed in Section 3.1.2 also emerges in these experiments: network performance degrades as the training data is limited. This trend is observed regardless of the deep architecture. Further, the suboptimal classification performance caused by the HeadOnly fine-tuning is again observed. The positive

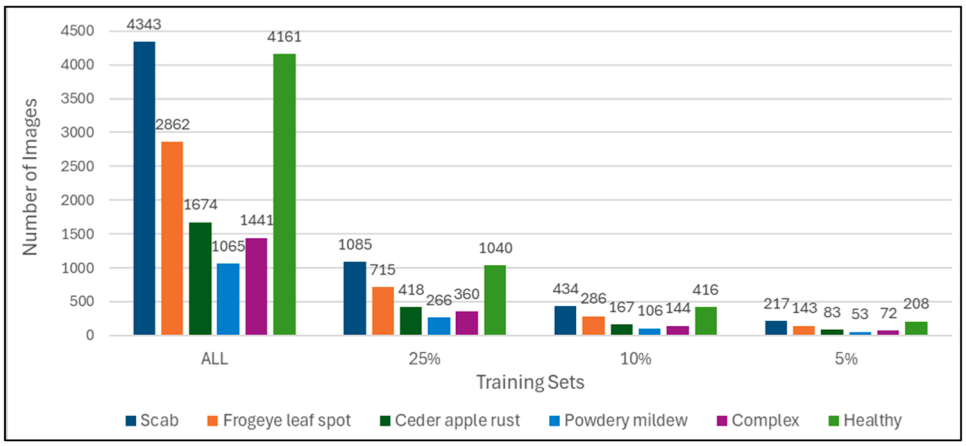

Fig. 11. Class distribution of the training sets used in experiments: All, 25 %, 10 % and 5 %.

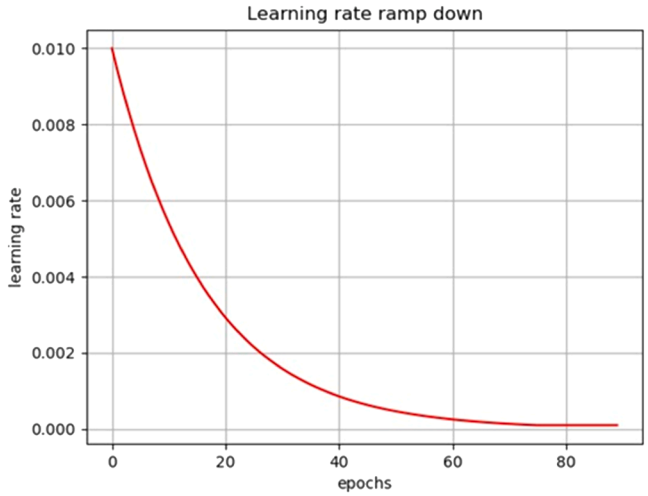

Fig. 12. Learning rate ramp down during head training used in our experiments.

Table 9
Training hyperparameters used in HeadOnly and HeadThenBody fine-tuning.

Hyperparameter	HeadOnly	HeadThenBody
Epochs	90	90
SGD initial learning rate	0.01	0.0001
LR decay	exponential decay	no decay
SGD momentum	0.9	0.9
SGD weight decay	0.0001	0.0001
Early stopping patience	5	5

impact of fine-tuning the model body is a clear result of these experiments. Looking beyond these general trends, we primarily see that ResNET-50 model, which we used as the backbone for the semi-supervised methods in our study, lags behind other models. We identify this point as a future work. The poor performance of the Inception V3 model in HeadOnly fine-tuning suggests that this model is not suited for training a classifier alone. However, when all parameters of the Inception V3 are involved in the training, we see performances that are on par with the other models and even ahead of them for some data sizes. Lastly, ConvNext model shows an out of distribution performance. The generalization performance of the ConvNext model with limited data is poor. In the training of the FCN classifier (HeadOnly), it outperforms the other models when trained with reasonable amounts of data (All and 25 % sets), while the performance drops drastically when the data is severely limited (10 % and 5 % sets). This trend is also reflected in the HeadThenBody fine-tuning results. When fine-tuning the ConvNext model with all parameters, it outperforms the other architectures by a large margin for All, 25 % and 10 % data amounts, but lags behind the other architectures when data is severely limited (5 % set). Other architectures exhibit similar trends. It is noteworthy that MobileNET V3 architecture performs close to other architectures with its limited trainable parameters (Table 12).

In this section, we first examined the factors influencing supervised learning with the PP2021TS dataset. Through hyperparameter tuning, we identified unbalanced dataset and batch size of 32 as optimal. We also assessed the impact of weak and strong input augmentations on network performance. Initially, we focused on training set size and strategy in the specific case of the ResNet-50 model, and then extended the analysis to four other widely accepted architectures. Our findings revealed that limiting the training set consistently degraded performance across all architectures, while fine-tuning the body of the model improved it.

Table 10
Supervised-only training results of ResNET-50 on PP2021TS dataset. BS indicates batch size.

Training Set	Top1-HeadOnly Balanced	Top1-HeadThenBody BS.16	BS.32	BS.128	BS.256	BS.32
All	True	85.4342@17	85.9944@25	–	-	–
All	False	86.7129@29	87.0017@25	86.6551@36	85.0376@42	93.2987@24
25 %	True	82.7731@23	80.9524@30	–	–	–
25 %	False	82.7267@19	83.4778@28	–	–	87.3484@13
10 %	True	77.1709@17	76.4706@59	–	–	–
10 %	False	79.2028@16	80.7048@21	–	–	86.2507@45

accuracy@epoch: At which **epoch** was this **accuracy** achieved with early stopping?

Fig. 13. Weak and strong augmentations. (Top: No augmentation, Middle: Weak augmentations, Bottom: Strong augmentations).

Table 11
Network performance under weak and strong augmentations. BS indicates batch size.

Training Set	Augmentation	Top1-HeadOnly BS.32	Top1-HeadThenBody BS.32
All	Weak	87.0017@25	93.2987@24
All	Strong	84.3443@55	93.1831@44
25 %	Weak	83.4778@28	87.3484@13
25 %	Strong	80.9936@75	91.0456@89
10 %	Weak	80.7048@21	86.2507@45
10 %	Strong	77.4119@89	87.4061@89
5 %	Weak	77.3541@38	81.7447@43
5 %	Strong	69.7285@36	78.7406@48

accuracy@epoch: At which **epoch** was this **accuracy** achieved with early stopping?

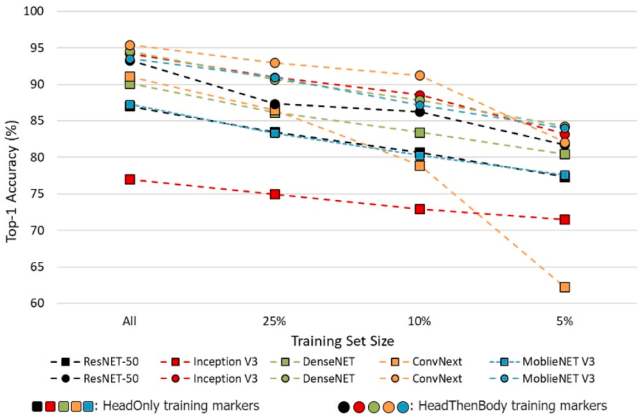

Fig. 14. Top-1 accuracy of several deep architectures.

Table 12
Deep architectures and number of parameters used in the evaluation.

Architecture	# of parameters
ResNET-50 [45]	~25.5M
Inception V3 [52]	~27.2M
DenseNET [53]	~20.0M
ConvNext [54]	~28.6M
MobileNET V3 [55]	~5.5M

3.3. Semi-supervised training experiments

3.3.1. Mean teacher vs ResNET-50

This section compares performance of the mean-teacher (MT) method and the ResNET-50 model. First, we give training hyperparameters specific to mean-teacher training, then compare the performances. Refer to Section 2.1 for details of the mean-teacher method.

Student model of the mean-teacher method is a ResNET-50 pretrained on ImageNET. Teacher model has the same structure (ResNET-50), as it holds the exponential moving average (EMA) of the student weights. The choice of consistency type is mean squared error (MSE). The EMA decay value is 0.99. This is the value used in the "4000 labels/ ResNET on CIFAR-10″ test in the MT paper. 4000 labels in CIFAR-10 constitute 8 % of the CIFAR-10 training set, which is close to our 10 % training set. As the 10 % label ratio is in the middle of our tests (between 5 % and 25 %), 0.99 EMA decay was adopted for all experiments. The final consistency weight is set to 30. On the mean-teacher's GitHub page [56] their recommendation for the final consistency weight for MSE is a value between the number of classes and the number of classes squared. This range is [6,36] for PP2021TS dataset, so 30 is chosen for the consistency weight. Consistency ramp-up is 5 epochs. This is the common choice in the MT paper. One batch consists of 10 labeled and 30 unlabeled samples. Refer to Section A.2 for computation of these sizes. All other parameters are equal to supervised training parameters (Table 9). As an exception early stopping was not used, as the validation loss increases in the first 5 epochs due to unsupervised weight ramp-up. Table 13 lists all hyperparameters specific to mean-teacher training.

Table 13
Hyperparameters specific to mean-teacher training.

Hyperparameter	Value
Consistency type	MSE
EMA decay	0.99
Consistency weight (final)	30
Consistency ramp-up	5 epochs
Batch size (labeled, unlabeled)	(10, 30)

Table 14 compares performance of mean-teacher method and ResNET-50 model. All results are obtained using unbalanced dataset and weak augmentations. ResNET-50 training results are provided with early stopping turned off for fair comparison. When only the classifier head is trained with the MT method (HeadOnly), accuracy dropped 2.64 % in 25 % training set, stayed almost same in 10 % training set and increased 3.63 % in 5 % training set compared to ResNET-50 . HeadOnly strategy benefits from mean-teacher more when labeled data size is smaller. When we switch to fine-tuning the body after head training, we start to observe clear gains from the MT training. Fine-tuning with the mean-teacher method surpasses the ResNET-50 performance at every training set. Top-1 accuracy increased 1.13 %, 4 % and 5.52 % at 25 %, 10 % and 5 % training sets, respectively. Fig. 15 shows samples correctly classified by the MT method and misclassified by the ResNET-50 model on the 5 % training set.

The previous experiments (Table 14) compared the performance of mean-teacher and ResNET-50 on equal amounts of training data. In Fig. 16, we aim to find how much data is needed for the ResNET-50 model to reach the mean-teacher performance. The performance of ResNET-50 is given over finer-grained data amounts: now, training set sizes are from 5 % to 30 % in increments of 2.5 %. Square markers indicate HeadOnly training accuracies and round markers indicate HeadThenBody training accuracies. Black markers indicate ResNET-50 accuracies. Mean-teacher accuracies, given in Table 13, are marked with red, green and blue colors for 5 %, 10 % and 25 % training sets, respectively. These markers are positioned on the ResNET accuracy curve by intersecting a horizontal line, drawn at the mean-teacher accuracy, with the corresponding ResNET accuracy curve. Projections of these data points onto the horizontal axis of the graph (training set size axis) give the training set size needed for ResNET-50 model to reach this accuracy. Refer to Table 19 to find the numerical values of the accuracies and actual data amounts under the “number (#) of training samples” column of the table.

Fig. 16 clearly shows that the real impact of the mean-teacher method emerges in the HeadThenBody training. We repeat the comment of the previous experiment for the HeadOnly training result: mean-teacher becomes effective when the data is heavily limited. In HeadOnly training, the performance that mean-teacher achieves when trained with 5 % of the training set, ResNET can reach using 7.5 % of the training set. In HeadThenBody training, ResNET needs much more data to achieve mean-teacher performances. The performances achieved by mean-teacher with 5 %, 10 % and 25 % of the training set can be achieved by the ResNET model using approximately 12 %, 26 % and 30 % of the training set, respectively. ResNET needs to be trained with approximately 2.5 times more data in order to reach the performances of mean-teacher method with 5 % and 10 % of the training set.

In this section, we showed the superiority of semi-supervised learning over traditional supervised learning through the mean-teacher method and the ResNET-50 model. We demonstrated that the mean-teacher method is more effective with limited data. Additionally, we found that its accuracy improves significantly when the model’s body is included in the training process.

3.3.2. *Mean teacher vs other semi-supervised learning methods*

This section compares the mean-teacher method to other semi-supervised methods. The semi-supervised methods considered are Virtual Adversarial Training (VAT) [32] and MixMatch [57]. We first provide a brief description of the methodologies used in VAT and MixMatch, and then compare their performance to mean-teacher method.

Adversarial training is performed by applying perturbations to the sample inputs in the adversarial direction. The adversarial direction is defined as the direction that moves the prediction distribution farthest from the sample label. Perturbations in the adversarial direction increase the robustness of the network to difficult (adversarial) samples. VAT allows semi-supervised learning by computing these perturbations over the model prediction for unlabeled samples. VAT can be easily integrated into training with a few lines of code thanks to gradient-based optimization frameworks already in use, such as PyTorch.

MixMatch combines several techniques, such as consistency regularization, data augmentation, and label guessing, to improve classification performance. The method first applies data augmentation to both labeled and unlabeled data. Then, it produces "guessed" labels for the unlabeled data by averaging predictions from different augmentations. These guessed labels are treated as pseudo-labels. MixMatch then mixes both labeled and pseudo-labeled data using MixUp [58], a technique that linearly interpolates between data points and their labels.

We compare VAT and MixMatch performance with the mean-teacher results provided in Section 3.3.1. Table 13 lists the hyperparameters used in mean-teacher training.

VAT is controlled by two hyperparameters: (a) the norm constraint ε which controls the size of the perturbations, (b) the regularization co-

Table 14
Top-1 accuracy of mean-teacher (M.T.) method and ResNET-50 model trained on PP2021TS dataset. HeadOnly and HeadThenBody training applied for each method.

	Top1-HeadOnly		Top1-HeadThenBody	
Training Set	ResNET-50	M.T.	ResNET-50	M.T
25 %	84.1518	81.529	91.0156	92.1433
10 %	81.7522	81,6964	87.221	91.2189
5 %	77.567	81.25	82.9799	88.5038

Fig. 15. One sample from each category correctly classified by the MT model using 5 % of the labels. (From left to right, Top: complex, frog eye leaf spot, healthy, Bottom: powdery mildew, rust, scab).

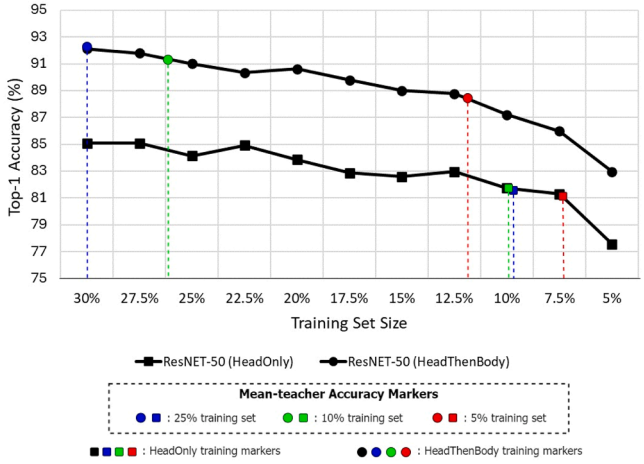

Fig. 16. Top-1 accuracy of ResNET-50 model and mean-teacher method for several training set sizes. Mean-teacher accuracies of Table 14 are positioned appropriately on the ResNET-50 curves. The verticals descending from these data points to the training set axis give the amount of data needed by ResNET-50.

efficient α which controls the relative balance between supervised and unsupervised losses. Miyato et al. [32] (VAT paper) reported that these two hyperparameters play similar roles and fixing $\alpha = 1$ and tuning ε is enough in practice. In our experiments, we fixed α to 1 and set ε to 10. VAT paper reports good performance on the CIFAR-10 dataset in the vicinity of 10 for ε.

MixMatch is controlled by four hyperparameters: (a) number of augmentations applied to unlabeled images, K, (b) sharpening temperature, T, (c) parameter that controls the amount of blending between images and labels (MixUp), α and (d) final unsupervised loss weight, λ_u. For all MixMatch experiments we set $K = 2$, $T = 0.5$, $\alpha = 0.75$, and $\lambda_u = 75$. Unsupervised loss is increased from 0 to $\lambda_u = 75$ with a ramp-up function.

The backbone model for VAT and MixMatch is ResNET-50. One batch consists of 24 labeled and 24 unlabeled samples for both methods. One training loop lasts 90 epochs, and early-stopping is turned off. All other hyperparameters specific to backbone training is same as listed in Table 9.

Table 15 compares performance of mean-teacher, virtual adversarial training (VAT) and MixMatch methods trained on PP2021TS dataset. HeadOnly and HeadThenBody training protocols discussed in Section 2.4 are applied to each method. The most obvious result is that mean-teacher lags behind the other methods in almost all training sets and strategies. This is, in fact, an expected result given the binary comparison between mean-teacher and MixMatch. Mean-teacher adopts consistency regularization as a precursor to MixMatch. MixMatch, on the other hand, is a holistic approach that uses multiple semi-supervised approaches, including consistency regularization, collectively at the same time. VAT is orthogonal to other methods, adopting a completely different approach. It mostly produced results in between mean-teacher and MixMatch. VAT has shown remarkable results with the HeadThenBody training protocol, achieving over 90 % accuracy even in the experiment where 5 % of the training set was used. Overall, MixMatch performed better in 4 of the 6 experiments than other methods, VAT produced accuracies comparable to MixMatch and mean-teacher demonstrated low performances lagging behind up to 2–3 % below other methods.

In this section, we compared the mean-teacher method to other alternatives, virtual adversarial training (VAT) and MixMatch. We found that the mean-teacher was behind other methods. The results showed that this study can be carried out further with other semi-supervised learning methods.

3.4. Zero-shot learning experiment on CLIP

In this section, we measure the performance of zero-shot learning on the PP2021TS dataset. Zero-shot models are at the most extreme end of the spectrum in terms of the goal of reducing data collection dependency, which is our focus in this study. These models are usually trained on very large datasets to generalize across unseen categories. Data types can be text, image, paired image-text data or multimodal data containing audio, video and text which allows the model work across different input types.

CLIP (Contrastive Language-Image Pre-Training) [59] is a zero-shot model developed by OpenAI that can understand both images and text in a unified manner. It is trained using a contrastive learning approach where it learns to associate images with their corresponding textual descriptions. The model is trained on a vast dataset of images and their associated captions, enabling it to generalize well across different visual and textual tasks. CLIP can classify images without needing task-specific labeled data, simply by being provided with natural language descriptions of the categories.

Following the practice on CLIP's GitHub page [60], we expect CLIP to classify diseased and healthy leaf images in the PP2021TS dataset according to the query texts specified. Query text for each category is listed in Table 16. Cues about the image context were provided by the terms 'leaf' and 'plant disease' in the queries.

Image features of the validation set images of PP2021TS dataset were compared to text features generated from the queries. The category of the query that is most similar to the image in the feature space is identified as CLIP's prediction. CLIP produced overall accuracy of 36.28 %. Fig. 17 compares the predicted label count to the true label count of the validation set. CLIP is highly biased towards selecting the 'Scab' category.

In this section, we applied zero-shot learning to plant disease recognition over the PP2021TS dataset. We adopted the CLIP model. Despite the queries augmented with image context cues, we found that CLIP performed poorly in discriminating plant diseases.

4. Conclusion

In this paper, we investigated the potential of semi-supervised learning to reduce the data and labeling dependency of plant disease recognition with deep architectures. Three datasets from the literature were evaluated according to various criteria, and one of them was decided upon to conduct our study. First, an ablation study was conducted on the supervised training of the ResNet-50 model to assess its sensitivity to dataset and training parameters. The study evaluated the effects of labeled data quantity, dataset balance, batch size, and fine-tuning strategy on model performance. It was found that an unbalanced dataset and a batch size of 32 produced good results, leading to the fixation of these two factors. The study also demonstrated that limiting labeled data hindered the ResNet-50 model's supervised learning performance, while including the backbone network in the training improved model adaptability. The final supervised learning ablation experiment assessed the model's sensitivity to various levels of input augmentation. In the second phase of the supervised training experiments, five widely accepted deep architectures were trained on different training set sizes. All architectures performed poorly with limited data but achieved peak performance using the HeadThenBody training protocol.

Table 16
Query text per category used in CLIP predictions.

Category	Query Text
Scab	'A leaf image with scab plant disease'
Frogeye leaf spot	'A leaf image with frogeye leaf spot plant disease'
Ceder apple rust	'A leaf image with rust plant disease'
Powdery mildew	'A leaf image with powdery mildew plant disease'
Complex	'A leaf image with more than one plant disease'
Healthy	'A healthy leaf image'

Table 15
Top-1 accuracy of mean-teacher (M.T.), virtual adversarial training (VAT) and MixMatch methods trained on PP2021TS dataset. HeadOnly and HeadThenBody training protocols applied for each method.

	Top1-HeadOnly			Top1-HeadThenBody		
Training Set	M.T.	MixMatch	VAT	M.T.	MixMatch	VAT
25 %	81.529	84.4308	85.3790	92.1433	94.0290	93.6380
10 %	81,6964	83.5938	82.8130	91.2189	93.3594	92.6340
5 %	81.25	82.7009	78.7760	88.5038	89.9554	90.5690

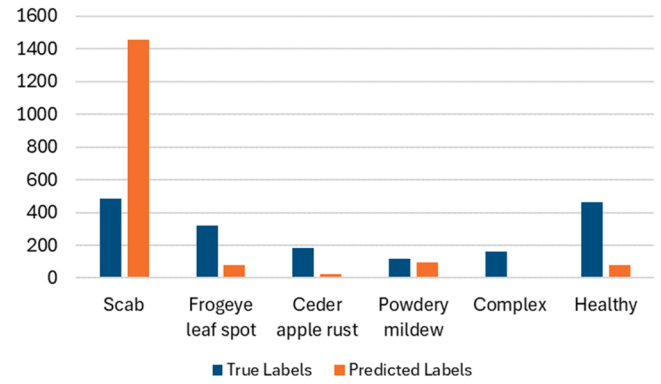

Fig. 17. Label count predicted by CLIP vs true label count of the validation set.

Finally, we transitioned to semi-supervised learning experiments using the mean-teacher model. Initially, we demonstrated the superiority of semi-supervised learning over traditional supervised learning through comparisons between the mean-teacher method and the ResNet-50 model. In the first set of experiments, we compared the methods using equal training data sizes. While the mean-teacher model lagged behind the ResNet-50 model with HeadOnly training, it outperformed ResNet-50 with HeadThenBody training. In the second set of experiments, we assessed how much data the ResNet model requires to match the performance of the mean-teacher model. Specifically, for smaller training sets, we found that the ResNet model needed up to 2.5 times more data to achieve comparable performance. We also compared the mean-teacher method to other semi-supervised learning methods and found that it performed worse. This suggests that further investigation with alternative semi-supervised methods may be warranted. Lastly, we explored whether zero-shot learning could serve as an alternative to semi-supervised learning for plant disease recognition. Our findings indicated that the CLIP model used for zero-shot learning lagged significantly behind the semi-supervised learning methods.

In future work, we aim to improve our work in the following directions: (1) expanding the dataset to include more crop and disease classes, (2) using different backbone networks, (3) measuring semi-supervised learning performance under strong augmentations, and (4) training the model with the fine-tuning strategy where the classifier and backbone network are trained simultaneously. Besides, our source codes and pretrained models are available at https://github.com/milsever/plant-pathology.

Ethics statement

Not applicable: This manuscript does not include human or animal research.

CRediT authorship contribution statement

Murat Ilsever: Writing – original draft, Visualization, Validation, Software, Investigation, Data curation. **Ipek Baz:** Writing – review & editing, Methodology, Investigation.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgement

M.İ. acknowledges the support of TUBITAK BIDEB 2211/A National PhD Scholarship Program. The numerical calculations reported in this paper were partially performed at TUBITAK ULAKBIM, High Performance and Grid Computing Center (TRUBA resources).

Appendix

A.1. Supervised learning performance of deep architectures

Table 17
HeadOnly fine-tuning performance of several deep architectures.

Training Set	ResNET-50	Inception V3	DenseNET	ConvNext	MobileNET V3
All	87.0017@25	76.9903@19	90.1228@25	91.0714@41	87.2768@37
25 %	83.4778@28	74.9814@20	86.1607@20	86.4955@35	83.3147@28
10 %	80.7048@21	72.9353@18	83.4263@14	78.9063@83	80.2827@33
5 %	77.3541@38	71.5216@54	80.4808@20	62.2582@85	77.6228@34

accuracy@epoch: At which **epoch** was this **accuracy** achieved with early stopping?

Table 18
HeadThenBody fine-tuning performance of several deep architectures.

Training Set	ResNET-50	Inception V3	DenseNET	ConvNext	MobileNET V3
All	93.2987@24	94.2522@24	94.5313@12	95.4241@22	93.5268@28
25 %	87.3484@13	90.9598@30	90.6808@13	92.9688@14	90.9598@43
10 %	86.2507@45	88.5608@53	87.8348@15	91.2388@32	87.1652@56
5 %	81.7447@43	83.2031@34	84.2634@15	82.0871@36	83.9844@52

accuracy@epoch: At which **epoch** was this **accuracy** achieved with early stopping?

A.2. Computation of the "aux-batch-size" argument in mean-teacher training

The aux-batch-size argument of our training script determines the size of a batch of labeled and unlabeled samples. Another argument, labeled batch-size, determines the batch size in supervised training and the labeled data size in MT training. The unlabeled data size is calculated as the

difference between these two arguments: (aux-batch-size - labeled-batch-size). In MT training one epoch lasts until all unlabeled samples are fetched. In "4000 labels/ResNET on CIFAR-10″ test of MT paper aux-batch-size is 128, and labeled-batch-size is 31. Then, 97 samples in a batch are unlabeled. The training set of CIFAR10 has 50,000 samples, of which 4000 are labeled and 46,000 are unlabeled. Eventually, one epoch lasts 46,000/97=474 iterations. We aimed to reach 474 iterations in our 10 % tests. Our 10 % test has 13,992 unlabeled samples. To obtain 474 iterations in one epoch, unlabeled samples in one batch should be 13,992/474~=30. Then, the number of labeled samples in a batch should be 10, as it is one-third of the unlabeled samples, and aux-batch-size should be 40.

A.3. Performance of ResNET-50 model on closely spaced training set sizes

Table 19
Top-1 accuracy of ResNET-50 model on several training set sizes given for HeadOnly and HeadThenBody training protocols.

Training Set	# of Training Samples	Top1-HeadOnly ResNET-50	Top1-HeadThenBody ResNET-50
30 %	4663	85.1004	92.1317
27.5 %	4275	85.1004	91.7969
25 %	3886	83.4778	90.6412
22.5 %	3497	84.933	90.346
20 %	3109	83.8728	90.625
17.5 %	2720	82.8683	89.7879
15 %	2331	82.5893	89.0067
12.5 %	1943	82.9799	88.7835
10 %	1553	80.7048	87.5217
7.5 %	1165	81.3058	85.9933
5 %	777	77.3541	83.8244

Data availability

Data will be made available on request.

References

[1] W. Albattah, M. Nawaz, A. Javed, M. Masood, S. Albahli, A novel deep learning method for detection and classification of plant diseases, Complex Intell. Syst. 8 (2022) 507–524, https://doi.org/10.1007/s40747-021-00536-1.

[2] V. Singh, A.K. Misra, Detection of plant leaf diseases using image segmentation and soft computing techniques, Inf. Process. Agric. 4 (2017) 41–49, https://doi.org/10.1016/j.inpa.2016.10.005.

[3] B. Güven, İ. Baz, B. Kocaoğlu, E. Toprak, D.E. Barkana, B.S. Özdemir, Smart farming technologies for sustainable agriculture: from food to energy. A Sustainable Green Future: Perspectives on Energy, Economy, Industry, Cities and Environment, Springer, 2023, pp. 481–506, https://doi.org/10.1007/978-3-031-24942-6_22.

[4] M. Pagola, R. Ortiz, I. Irigoyen, H. Bustince, E. Barrenechea, P. Aparicio-Tejo, C. Lamsfus, B. Lasa, New method to assess barley nitrogen nutrition status based on image colour analysis. Comparison with SPAD-502, Comput. Electron. Agric. 65 (2009) 213–218, https://doi.org/10.1016/j.compag.2008.10.003.

[5] H. Onoyama, C. Ryu, M. Suguri, M. Iida, Nitrogen prediction model of rice plant at panicle initiation stage using ground-based hyperspectral imaging: growing degree-days integrated model, Precis. Agric. 16 (2015) 558–570, https://doi.org/10.1007/s11119-015-9394-9.

[6] I.I. Tartachnyk, I. Rademacher, W. Kühbauch, Distinguishing nitrogen deficiency and fungal infection of winter wheat by laser-induced fluorescence, Precis. Agric. 7 (2006) 281–293, https://doi.org/10.1007/s11119-006-9008-7.

[7] C.T. Chen, S. Chen, K.W. Hsieh, H.C. Yang, S. Hsiao, I.C. Yang, Estimation of leaf nitrogen content using artificial neural network with cross-learning scheme and significant wavelengths, Trans. ASABE 50 (2007) 295–301.

[8] Y. Lin, LiDAR: an important tool for next-generation phenotyping technology of high potential for plant phenomics? Comput. Electron. Agric. 119 (2015) 61–73, https://doi.org/10.1016/j.compag.2015.10.011.

[9] S.P. Mohanty, D.P. Hughes, M. Salathé, Using deep learning for image-based plant disease detection, Front. Plant Sci. 7 (2016), https://doi.org/10.3389/fpls.2016.01419.

[10] S. Sladojevic, M. Arsenovic, A. Anderla, D. Culibrk, D. Stefanovic, Deep neural networks based recognition of plant diseases by leaf image classification, Comput. Intell. Neurosci. 2016 (2016), https://doi.org/10.1155/2016/3289801.

[11] P. Goncharov, G. Ososkov, A. Nechaevskiy, A. Uzhinskiy, I. Nestsiarenia, Disease detection on the plant leaves by deep learning, Stud. Comput. Intell. (2019) 151–159, https://doi.org/10.1007/978-3-030-01328-8_16.

[12] M. Kerkech, A. Hafiane, R. Canals, Vine disease detection in UAV multispectral images using optimized image registration and deep learning segmentation approach, Comput. Electron. Agric. 174 (2020) 105446, https://doi.org/10.1016/j.compag.2020.105446.

[13] S. Branson, G. Van Horn, S. Belongie, P. Perona, Bird species categorization using pose normalized deep convolutional nets, in: Proceedings of the British Machine Vision Conference, BMVC 2014, 2014.

[14] N. Zhang, J. Donahue, R. Girshick, T. Darrell, Part-based R-CNNs for fine-grained category detection, in: Proceedings of the ECCV 2014, 2014, pp. 834–849.

[15] X.S. Wei, C.W. Xie, J. Wu, C. Shen, Mask-CNN: localizing parts and selecting descriptors for fine-grained bird species categorization, Pattern Recognit. 76 (2018), https://doi.org/10.1016/j.patcog.2017.10.002.

[16] Y. Zhang, X.S. Wei, J. Wu, J. Cai, J. Lu, V.A. Nguyen, M.N. Do, Weakly supervised fine-grained categorization with part-based image representation, IEEE Trans. Image Process. 25 (2016) 1713–1725, https://doi.org/10.1109/TIP.2016.2531289.

[17] X. He, Y. Peng, Weakly supervised learning of part selection model with spatial constraints for fine-grained image classification, in: Proceedings of the 31st AAAI Conference on Artificial Intelligence AAAI 2017, 2017, pp. 4075–4081, https://doi.org/10.1609/aaai.v31i1.11223.

[18] W. Ge, X. Lin, Y. Yu, Weakly supervised complementary parts models for fine-grained image classification from the bottom up, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2019, pp. 3029–3038, https://doi.org/10.1109/CVPR.2019.00315.

[19] T.Y. Lin, A. RoyChowdhury, S. Maji, Bilinear CNNs for fine-grained visual recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1449–1457, https://doi.org/10.1109/ICCV.2015.170.

[20] Y. Gao, O. Beijbom, N. Zhang, T. Darrell, Compact bilinear pooling, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, pp. 317–326, https://doi.org/10.1109/CVPR.2016.41.

[21] Y. Li, N. Wang, J. Liu, X. Hou, Factorized bilinear models for image recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2098–2106, https://doi.org/10.1109/ICCV.2017.229.

[22] C. Yu, X. Zhao, Q. Zheng, P. Zhang, X. You, Hierarchical bilinear pooling for fine-grained visual recognition, in: Proceedings of the Lecture Notes in Computer Science, 2018, pp. 595–610, https://doi.org/10.1007/978-3-030-01270-0_35 (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics).

[23] H. Zheng, J. Fu, Z.J. Zha, J. Luo, Learning deep bilinear transformation for fine-grained image representation, Adv. Neural Inf. Process. Syst. 32 (2019).

[24] A. Dubey, O. Gupta, R. Raskar, N. Naik, Maximum-entropy fine grained classification, Adv. Neural Inf. Process. Syst. (2018) 31.

[25] A. Dubey, O. Gupta, P. Guo, R. Raskar, R. Farrell, N. Naik, Pairwise confusion for fine-grained visual classification, in: Proceedings of the Lecture Notes in Computer Science, 2018, pp. 71–88, https://doi.org/10.1007/978-3-030-01258-8_5 (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics).

[26] X. Liu, W. Min, S. Mei, L. Wang, S. Jiang, Plant disease recognition: a large-scale benchmark dataset and a visual region and loss reweighting approach, IEEE Trans. Image Process. 30 (2021), https://doi.org/10.1109/TIP.2021.3049334.

[27] J.T. Springenberg, Unsupervised and semi-supervised learning with categorical generative adversarial networks, ArXiv Prepr. ArXiv1511.06390 (2015). http://arxiv.org/abs/1511.06390.

[28] A. Odena, Semi-Supervised Learning with Generative Adversarial Networks, ArXiv Prepr. ArXiv1606.01583 (2016). http://arxiv.org/abs/1606.01583.

[29] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, X. Chen, Improved techniques for training GANs, Adv. Neural Inf. Process. Syst. (2016) 2234–2242. http://arxiv.org/abs/1606.03498.
[30] S. Laine, T. Aila, Temporal ensembling for semi-supervised learning, in: Proceedings of the 5th International Conference on Learning Representations, ICLR 2017, 2017, pp. 1–13. Conf. Track Proc.
[31] A. Tarvainen, H. Valpola, Mean teachers are better role models: weight-averaged consistency targets improve semi-supervised deep learning results, Adv. Neural Inf. Process. Syst. (2017).
[32] T. Miyato, S.I. Maeda, M. Koyama, S. Ishii, Virtual adversarial training: a regularization method for supervised and semi-supervised learning, IEEE Trans. Pattern Anal. Mach. Intell. 41 (2019) 1979–1993, https://doi.org/10.1109/TPAMI.2018.2858821.
[33] S. Qiao, W. Shen, Z. Zhang, B. Wang, A. Yuille, Deep co-training for semi-supervised image recognition, in: Proceedings of the European Conference on Computer Vision, 2018, pp. 135–152.
[34] Q. Xie, M.T. Luong, E. Hovy, Q.V. Le, Self-training with noisy student improves imagenet classification, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2020, pp. 10684–10695, https://doi.org/10.1109/CVPR42600.2020.01070.
[35] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, G. Hinton, Big self-supervised models are strong semi-supervised learners, Adv. Neural Inf. Process. Syst. (2020) 22243–22255, 2020-Decem.
[36] L. Schmarje, M. Santarossa, S.M. Schröder, R. Koch, A survey on semi-, self- and unsupervised learning for image classification, IEEE Access 9 (2021), https://doi.org/10.1109/ACCESS.2021.3084358.
[37] P. Sharma, A. Sharma, A novel plant disease diagnosis framework by integrating semi-supervised and ensemble learning, J. Plant Dis. Prot. (2024) 131, https://doi.org/10.1007/s41348-023-00803-y.
[38] J. Liu, Y. Hu, Q. Su, J. Guo, Z. Chen, G. Liu, Semi-supervised one-stage object detection for maize leaf disease, Agriculture 14 (2024) 1140, https://doi.org/10.3390/agriculture14071140.
[39] C. Chen, W. Zhang, R. Fu, T. Wang, T. Pang, J. Wu, Z. Zheng, X. Wang, S2MD2: a method for semi-supervised maize leaf disease detection, in: Proceedings of the IEEE 33rd International Symposium on Industrial Electronics, IEEE, 2024, pp. 1–6, https://doi.org/10.1109/ISIE54533.2024.10595796.
[40] W. Tao, J. Ma, J. Shi, W. Lv, M. Zhao, L. Zheng, L. Huang, S. Weng, Dual-strategy semi-supervised learning method based on GAN for recognition of tomato leaf diseases, Int. J. Remote Sens. 43 (2022), https://doi.org/10.1080/01431161.2022.2123722.
[41] Y. Li, X. Chao, Semi-supervised few-shot learning approach for plant diseases recognition, Plant Methods 17 (2021), https://doi.org/10.1186/s13007-021-00770-1.
[42] S.S. Keh, Semi-supervised noisy student pre-training on EfficientNet architectures for plant pathology classification, (2020).
[43] G.E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, R.R. Salakhutdinov, Improving neural networks by preventing co-adaptation of feature detectors, (2012). http://arxiv.org/abs/1207.0580.
[44] L. Wan, M. Zeiler, S. Zhang, Y. LeCun, R. Fergus, Regularization of neural networks using DropConnect, in: Proceedings of the 30th International Conference on Machine Learning. ICML 2013, 2013.
[45] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, https://doi.org/10.1109/CVPR.2016.90.
[46] D. Singh, N. Jain, P. Jain, P. Kayal, S. Kumawat, N. Batra, PlantDoc: a dataset for visual plant disease detection, in: Proceedings of the ACM International Conference Proceeding Series, 2020, pp. 249–253, https://doi.org/10.1145/3371158.3371196.
[47] D.P. Hughes, M. Salathe, An open access repository of images on plant health to enable the development of mobile disease diagnostics, (2015). http://arxiv.org/abs/1511.08060.
[48] R. Thapa, Q. Wang, N. Snavely, S. Belongie, A. Khan, The Plant Pathology Challenge 2021 data set to classify foliar disease of apples, (2021). https://vision.cornell.edu/se3/wp-content/uploads/2021/09/029.pdf.
[49] R. Thapa, K. Zhang, N. Snavely, S. Belongie, A. Khan, The plant pathology challenge 2020 data set to classify foliar disease of apples, Appl. Plant Sci. 8 (2020), https://doi.org/10.1002/aps3.11390.
[50] R. Thapa, Q. Wang, N. Snavely, S. Belongie, A. Khan, Plant Pathology 2021–FGVC8, Kaggle, 2021. https://www.kaggle.com/c/plant-pathology-2021-fgvc8.
[51] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Köpf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, S. Chintala, PyTorch: an imperative style, high-performance deep learning library, Adv. Neural Inf. Process. Syst. (2019).
[52] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2016, https://doi.org/10.1109/CVPR.2016.308.
[53] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, 2017, https://doi.org/10.1109/CVPR.2017.243.
[54] Z. Liu, H. Mao, C.Y. Wu, C. Feichtenhofer, T. Darrell, S. Xie, A ConvNet for the 2020s, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2022, https://doi.org/10.1109/CVPR52688.2022.01167.
[55] A. Howard, M. Sandler, B. Chen, W. Wang, L.C. Chen, M. Tan, G. Chu, V. Vasudevan, Y. Zhu, R. Pang, Q. Le, H. Adam, Searching for mobileNetV3, in: Proceedings of the IEEE International Conference on Computer Vision, 2019, https://doi.org/10.1109/ICCV.2019.00140.
[56] A. Tarvainen, H. Valpola, “Mean teachers are better role models” Github, (2017). https://github.com/CuriousAI/mean-teacher (accessed January 3, 2023).
[57] D. Berthelot, N. Carlini, I. Goodfellow, A. Oliver, N. Papernot, C. Raffel, MixMatch: a holistic approach to semi-supervised learning, Adv. Neural Inf. Process. Syst. (2019).
[58] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, MixUp: beyond empirical risk minimization, in: Proceedings of the 6th International Conference on Learning Representations, ICLR 2018, 2018. Conf. Track Proc.
[59] A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, Learning transferable visual models from natural language supervision, in: Proceedings of the Machine Learning Research, 2021.
[60] A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, I. Sutskever, “CLIP” Github, (2021). https://github.com/openai/CLIP (accessed September 17, 2024).