HARVEST: High-Performance Artificial Vision Framework for Expert Labeling using Semi-Supervised Training

Nawras Alnaasan*, Matthew Lieber*, Aamir Shafi*, Hari Subramoni*,
Scott Shearer‡, and Dhabaleswar K Panda*
*Department of Computer Science and Engineering,
‡Department of Food, Agricultural, and Biological Engineering,
The Ohio State University, Columbus, Ohio, USA,
{alnaasan.1, lieber.31, shafi.16 subramoni.1, shearer.95}@osu.edu, panda@cse.ohio-state.edu

***Abstract*—**
Supervised Deep Learning (DL) thrives on Big Data; however, it inherits a major limitation—training and testing datasets must be fully annotated to train Deep Neural Networks (DNNs). To mitigate this bottleneck, we propose HARVEST—a distributed computer-vision framework that employs state-of-the-art semi-supervised learning (SSL) algorithms to train accurate DNNs using Distributed Data Parallelism (DDP) on High-Performance Computing (HPC) systems with only a small subset of labeled data samples. HARVEST offers an intuitive and interactive web-based interface that enables domain experts with no prior DL or HPC knowledge to easily unlock the power of DL and leverage the computational resources offered by HPC systems, furthering the mission of democratizing AI. We conduct a comprehensive evaluation of several Digital Agriculture use cases as an example domain that can benefit from HARVEST as data is collected frequently, in large volumes, and for a variety of applications. Our evaluations yield accuracies within 3% compared to fully supervised training using less than 80 labeled samples per class. Furthermore, we show that HARVEST delivers near-linear scaling, reducing the training time from 7.8 hours on a single NVIDIA A100 GPU up to 31 minutes by using DDP on 16 GPUs. To the best of our knowledge, HARVEST is the first framework that allows end-users to perform interactive labeling and distributed training using state-of-the-art SSL algorithms.

***Index Terms*—Semi-supervised Learning, Distributed Data-Parallelism, High-Performance Computing, Interactive Labeling**

I. Introduction

Recent advances in Artificial Intelligence (AI) have profoundly impacted a diverse range of environmental science domains, guiding a new era of data-driven innovation [1]. Supervised Deep Learning (DL) [2] is one of the most widely adopted paradigms to solve image classification problems in these domains due to its ability to capture complex input features and deliver state-of-the-art performance on Big Data. However, supervised DL comes with a caveat; it relies heavily on labeled data for training and evaluation. Acquiring and annotating large volumes of labeled data can be time-consuming, costly, and infeasible for certain applications, especially when the data is collected frequently and for different use cases.

*This research is supported in part by NSF #2112606.

Unsupervised DL [3], on the other hand, can be applied in specific scenarios to address some limitations of supervised DL. While valuable for discovering patterns and structures in unlabeled data, unsupervised DL has its own drawbacks: 1) it lacks a clear objective function or ground truth for evaluation, leading to poor accuracies compared to supervised training, and 2) it has limited applicability when it comes to traditional image classification tasks in which data points need to be assigned to predefined classes [4]. To overcome the limitations of supervised and unsupervised DL, semi-supervised learning (SSL) [5] uses a more balanced approach where a DL model can be trained on a small subset of labeled data and a large unlabeled dataset while maintaining high testing accuracies.

Examples of use cases that motivate our research can be found in the Digital Agriculture domain, where large datasets are automatically collected on a daily basis using unmanned aerial vehicles (UAVs) [6], unmanned ground vehicles (UGVs), or camera traps [7], rendering manual image labeling essentially impractical. Furthermore, domain experts may lack the technical knowledge to apply this data for creating accurate and scalable DL solutions for their use cases.

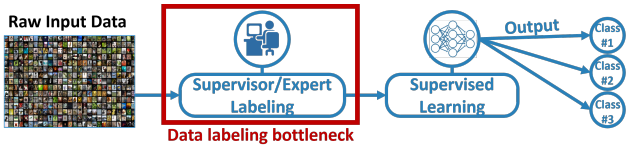
Therefore, we propose HARVEST—a distributed computer-vision framework that employs state-of-the-art (SSL) techniques and distributed data parallelism (DDP) to quickly train accurate DL models for image classification use cases using only a small subset of labeled data samples. HARVEST is an interactive and user-friendly framework—equipped with an easy-to-use interface—that enables domain experts with no prior experience in DL or High-Performance Computing (HPC) to easily create customized and efficient DL solutions for their use cases. HARVEST is built on top of PyTorch [8] and `torch.distributed` [9] to perform DNN training with data parallelism, Unified semi-supervised learning Benchmark (USB) [10] to utilize various SSL algorithms, SCAN [11] to perform unsupervised clustering for assisted user labeling, and Open OnDemand [12] to integrate a web-based interface that interacts with HPC systems. Additionally, we conduct a comprehensive evaluation on Digital Agriculture use cases as an example domain that can benefit from HARVEST.

A. Overview of the ICICLE Institute

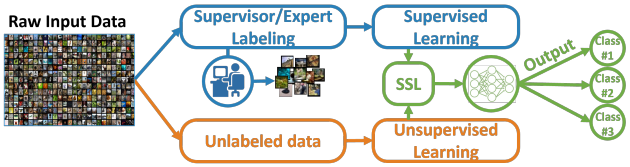
This work is part of the Intelligent Cyberinfrastructure with Computational Learning in the Environment (ICICLE) [13] project, which is an NSF-funded AI institute with a core mission of making Artificial Intelligence (AI) more accessible in environmental domains and driving its further democratization in the larger society. ICICLE aims to transform today's AI landscape from a narrow set of privileged disciplines to one where democratized AI empowers domains broadly through integrated plug-and-play AI. Research at ICICLE builds cyberinfrastructure (CI) for AI and AI for CI to address challenges in use-inspired science projects, including Digital Agriculture, Animal Ecology, and Smart Foodsheds. In the context of this paper, we emphasize ICICLE's core mission of democratizing AI by enabling domain experts with no prior DL or HPC expertise to easily create customized solutions for their use cases using HARVEST.

B. Challenges and Motivation

The ML/DL Challenge: Supervised ML and DL algorithms require fully labeled datasets to train image classification models. Figure 1a shows the traditional pipeline for supervised algorithms where the unlabeled raw input data is fully labeled by an expert, leading to a major bottleneck in this pipeline. The more samples and classes in a dataset, the more human labor is needed to produce high-quality annotations. For instance, creating the original ImageNet [14] dataset, which consisted of 1.2 million images, was a collaborative effort that involved many people and took several years to complete. To put this into perspective, if a person can label one image every minute on average, then it would require around 2.3 years to label 1.2 million images, given that the labeler does nothing but label images during that time frame. We address the following questions under the ML/DL challenge: 1) How can we train accurate DNNs for different use cases when unlabeled data is collected frequently in large volumes? 2) What is the impact of using different training algorithms, base DNN models, number of labeled images, and other training hyperparameters on the accuracy of DNNs?



(a) Workflow of supervised learning which requires fully labeled data.



(b) SSL combines both supervised learning to train on a small subset of labeled data and unsupervised learning to train on the large unlabeled dataset.

Fig. 1: (a) shows the data labeling bottleneck in the traditional supervised learning pipeline for image classification. (b) shows that semi-supervised learning (SSL) alleviates this bottleneck by only requiring a small subset of labeled images.

The Computing Challenge: The computational requirements for training DL models increase with the size of the dataset and complexity of the DL model. Furthermore, training complex DL models on large datasets demands substantial training times to converge to good solutions. Using a single machine for DL workloads may result in unreasonably long training time frames, rendering it pivotal to parallelize the training using distributed environments like High-Performance Computing (HPC) systems. Modern HPC systems, equipped with high-end clusters of CPUs and GPUs, fast interconnect, and large storage, provide the necessary computational muscle to accelerate DL applications. We address the following questions under the computing challenge: 1) How can we design a distributed framework that reduces the labeling bottlenecks and leverages HPC resources to enable accurate and efficient DNN training? 2) What is the expected distributed scaling efficiency using distributed data parallelism (DDP) on algorithms that mitigate the data labeling bottleneck?

The Application Domain Challenge: Digital Agriculture is an example domain where data is collected frequently, in large volumes, and for a variety of applications. There is a myriad of challenges in Digital Agriculture that can use DL solutions, ranging in use cases like crop classification, disease/defoliation detection, crop development, soil assessment, growth prediction, etc. Moreover, domain experts like agricultural engineers or farmers may lack the technical knowledge to train accurate DL models for their different use cases in distributed computing environments. In this paper, we look at several Digital Agriculture use cases including four publicly available datasets: PlantVillage [15], Weed Detection in Soybean [16], Sugar Cane-Spittle Bug [17], and Fruits-360 [18]; and 2 custom datasets for detecting stresses in soybean and corn crops [19]. We address the following questions under the application domain challenge: 1) How can we integrate an intuitive and user-friendly interface with HPC systems to enable domain experts with no prior DL or HPC experience to label data and train DNNs? 2) What is the expected performance in terms of accuracy metrics for Digital Agriculture use cases with algorithms that mitigate the data labeling bottleneck?

Overall, *the broad motivation of this paper is to design an intuitive and interactive framework for image classification using SSL techniques, alleviating the data labeling bottleneck and allowing domain experts with no prior DL or HPC knowledge to train accurate models quickly on HPC systems.*

C. Contributions

1) Design and implement HARVEST—an interactive and distributed framework for image classification that employs state-of-the-art SSL algorithms and DDP techniques to train accurate DNNs on HPC systems.
2) Create a training workflow by combining unsupervised learning and SSL techniques using SCAN [11] and Unified Semi-supervised learning Benchmark (USB) [10] in order to alleviate the data labeling bottleneck.

3) Integrate a web-based interface with HARVEST using Open OnDemand [12] to enable domain experts with no prior DL or HPC expertise to create customized and scalable solutions for image classification use cases.
4) Provide comprehensive analysis for training DNNs using different state-of-the-art SSL algorithms including MixMatch [20], FixMatch [21], AdaMatch [22], FreeMatch [23], and FlexMatch [24]. We also fine-tune hyperparameters for different DNNs including ResNet50 [25] and ViT [26].
5) Conduct scaling and accuracy evaluations on several Digital Agriculture datasets as an example domain. Our evaluations show that HARVEST yields high testing accuracies within 3% of fully supervised training and near-linear scaling on 16 A100 GPUs, reducing the training time from 7.8 hours on a single GPU to 31 minutes on 16 GPUs.
6) To the best of our knowledge, HARVEST is the first framework that allows end-users to perform interactive labeling and distributed training using state-of-the-art SSL algorithms.

The rest of this paper is organized as follows: Section II covers background on ML/DL paradigms, DDP, and Open OnDemand. Section III presents the architecture, workflow, and design of the proposed HARVEST framework. Section IV includes a comprehensive evaluation of SSL algorithms, base DNN models, and scaling efficiency on several public and custom Digital Agriculture datasets. In section V, we review related work. Finally, we conclude the paper in section VI.

II. Background

A. Machine and Deep Learning Paradigms

1) Supervised Learning: In supervised learning [2], the algorithm learns from labeled training data, where each data sample is associated with a target label. The main objective is to minimize the error on training data and generalize the model on unseen (testing) data. For training DNNs, a loss value is calculated by comparing the generated prediction and ground truth label, then backpropagating the loss to the rest of the model layers to adjust their weights. To train accurate models, supervised learning requires a large amount of labeled data with accurate annotations. Common tasks in supervised learning include classification and regression.

2) Unsupervised Learning: The main objective of unsupervised learning [3] is to recognize inherent patterns or relationships within the data and deduce useful information. State-of-the-art unsupervised learning techniques combine both DL methods for feature extraction and ML methods to analyze the extracted features. Common tasks of unsupervised learning include clustering, similarity search, and anomaly detection.

3) Semi-Supervised Learning: Semi-supervised learning (SSL) [5] falls between the supervised and unsupervised learning paradigms. Figure 1b shows the workflow of SSL algorithms for image classification. The raw input data is first sampled to label a small subset of images by an expert. This labeled data is then used by a supervised algorithm, which learns by generating predictions and comparing them to the target labels. At the same time, the large unlabeled dataset is fed into an unsupervised algorithm, which learns by assessing differences and similarities found in data features. The role of the SSL algorithm is to combine the information gained from both the supervised and unsupervised learning approaches. State-of-the-art SSL algorithms have shown significant potential to train accurate DNNs using only a few labeled images per class. One example is the FixMatch [21] algorithm, which uses two loss functions for the labeled and unlabeled data. Figure 2 shows a high-level overview of how FixMatch learns from unlabeled data. It performs weak and strong augmentation on the same image. The augmented images are then fed to the same DNN to generate predictions. The objective is to produce the same prediction for both weakly and strongly augmented images. If the generated predictions do not match, the loss is computed using cross-entropy loss, combined with the supervised loss, and then backpropagated to update the model weights.

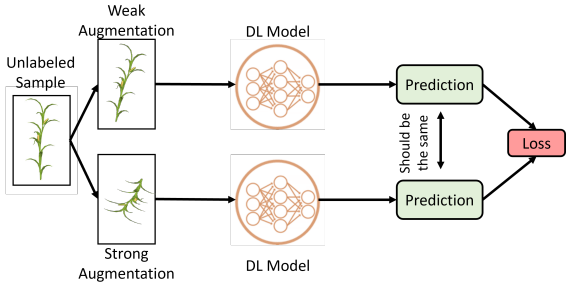


Fig. 2: High-level overview of the unsupervised procedure of the FixMatch [21] SSL algorithm training on unlabeled data.

B. Distributed Data-Parallel Deep Learning

Distributed Data-Parallelism (DDP) in DL is a powerful training technique that enables the processing of multiple data samples across several compute nodes to accelerate the training of Deep Neural Networks (DNNs). Data parallelism works by replicating the same DNN model across different processes and distributing the dataset as mini-batches. The forward and backward passes are performed on each process to calculate the activations, generate predictions, calculate the loss, and compute the gradients. Since each process is fed different data, we end up having different local gradients on each process. Therefore, synchronization is needed to aggregate and average the gradient values across all processes. This synchronization is performed with the Allreduce collective operation using Message Passing Interface (MPI) [27] libraries or NVIDIA Collective Communications Library (NCCL) [28] on NVIDIA GPUs as a communication backend. Allreduce collects the gradient values from all processes, aggregates them, and then broadcasts them back. At this point, all processes will have the same global gradients, which can be used to update the model parameters and progress to the next training iteration.

C. Open OnDemand

Open OnDemand [12] is an open-source and customizable web-based framework for interacting with HPC systems.

It allows integration with various HPC resources and job schedulers, such as Slurm [29], PBS, and LSF, to make HPC resources more accessible to users who may not be familiar with command-line interfaces. It has features such as job submission, file management, and remote visualization, providing a streamlined and user-friendly experience for researchers, engineers, and scientists. It's widely used by research institutions and commercial companies.

III. Proposed HARVEST Framework

HARVEST is an interactive and distributed framework that employs state-of-the-art unsupervised and semi-supervised training techniques to mitigate the bottleneck of data labeling and accelerate DNN training using distributed data parallelism (DDP). It allows users with no prior DL or HPC experience to create customized DL models for their image classification use cases through an intuitive web-based interface. In this section, we first show the layered architecture of the proposed framework HARVEST. We then take a look at the different components of HARVEST and how they interact with each other, with the user, and with the underlying HPC systems. Next, we explain the design for DDP with SSL algorithms. Finally, we show the workflow from the user's perspective using the web-based interface.

A. Architecture of the HARVEST Framework

Figure 3 depicts the layered architecture overview of the proposed HARVEST framework. The uppermost layer shows the image classification job. This job is defined by the user in terms of the provided dataset and list of target classes. The next layer shows the HARVEST framework, which consists of several components, including 1) the unsupervised trainer, which is responsible for clustering the data before displaying it to the user for labeling; 2) the image labeler, which is an interactive web-based tool that asks the user to label a number of samples for each class; 3) the semi-supervised trainer, which uses SSL algorithms to train a vision DL model using the labeled and unlabeled data; 4) the distributed training scheduler, which is responsible for preparing and submitting distributed DDP jobs; and 5) the inference engine, which generates the labels for the original unlabeled dataset and performs inference on new data samples.

HARVEST is based on various software packages including PyTorch [8] for DNN training, Unified Semi-supervised learning Benchmark (USB) [10] for implementations of various SSL algorithms, and SCAN [11] that is an unsupervised learning design used for data clustering. HARVEST also utilizes Open OnDemand [12] to build the web-based interface and link it to the underlying HPC system. The next layer shows `torch.distributed` [9], which is used by HARVEST to distribute the training jobs on HPC systems. Below are the communication and hardware libraries dependencies including MPI [27], NCCL [28], CUDA, and cuDNN. The final layer shows the modern HPC systems consisting of the interconnect, distributed file system, and several compute nodes, each with multi-core CPUs and GPUs.

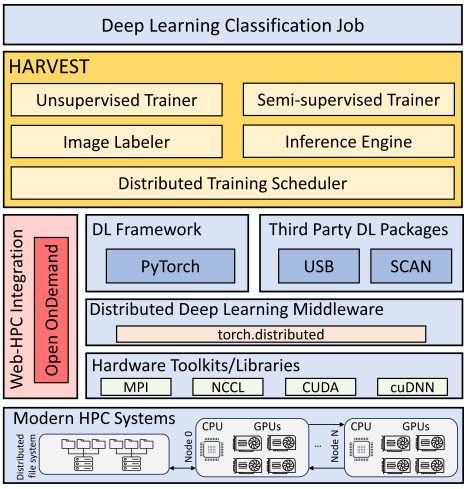


Fig. 3: Layered Architecture of the proposed *HARVEST* framework.

B. Workflow and Components of the HARVEST Framework

In this subsection, we take a closer look at the workflow of HARVEST and how its components interact with one another, the user, and the HPC system. Figure 4 depicts the overall flow of execution and operation of HARVEST. Below is a detailed description of each step:

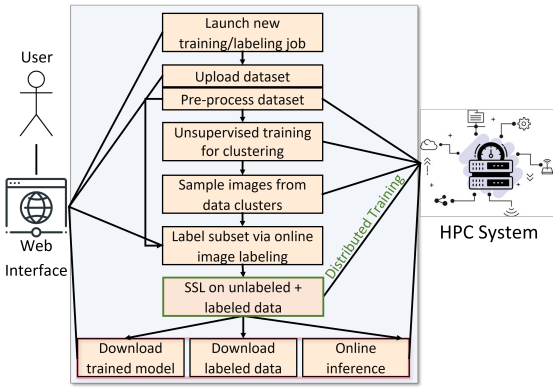


Fig. 4: High-level flow of execution and the steps involved in the operation of the *HARVEST* framework.

1) Launching New Training Jobs: At this step, the user is asked to define their use case through the web-based interface by specifying the number of classes and listing classes' names. Defined classes can be categorical, binary, or percentage-based. Furthermore, advanced users have the option of tweaking several hyperparameters to fine-tune their application, such as the number of labeled images per class, number of epochs, SSL algorithm, base DNN model, and learning rate. However, these hyperparameters are set by default to the configurations based on our comprehensive evaluation on Digital Agriculture use cases in section IV-B. Therefore, beginner users can expect satisfactory performance without any fine-tuning.

2) Data Pre-Processing: The user uploads the dataset to the file system, provides a pointer to their dataset, and submits their use case. Next, a Slurm [29] job is scheduled on the HPC system to initiate the pre-processing of the dataset. Data pre-processing includes rearranging the directory structure so it can be processed by the unsupervised and SSL algorithms.

3) Unsupervised Training for Image Clustering: When the data pre-processing step is finished, an unsupervised training job is scheduled by HARVEST on the HPC system using Slurm to cluster the dataset. The number of clusters is set to the number of classes defined by the user. To cluster the data, we use SCAN [11], a state-of-the-art unsupervised auto-labeling and clustering method that utilizes DL and ML methods such as SimCLR and K-nearest neighbors.

4) Image Sampling: After clustering the data using SCAN, we sample images from each cluster to be presented to the user in groups. This step aids the user by grouping images with similar visual features together to speed up the labeling process. We should point out that the unsupervised clustering is an optional step. The alternative approach is to randomly sample images from the entire dataset and present them to the user for labeling without prior clustering. If we sample enough images, the random sampling will approximate the actual data distribution. The only downside is that similar images will not be grouped but randomly presented to the user.

5) User Assisted Image Labeling: After images are sampled by either using unsupervised training or random sampling, the user is notified via the web-based interface and presented with a group of images for labeling. For each class defined by the user in the initial step, the user is asked to label a small subset of images from the sampled data. We later show in the evaluation section IV-B that labeling less than 80 images per class is sufficient to reach high accuracies within 3% of the fully supervised training accuracy. Indices and targets for the user-labeled images are saved on the HPC system to be used by the SSL algorithm in the next step.

6) Distributed SSL Training on Unlabeled and Labeled Data: SSL training requires both the small labeled data subset collected in the previous step and the original unlabeled dataset. We split the labeled data subset into training and evaluation data. Next, we use the Unified Semi-supervised learning Benchmark (USB) [10] as a backend, which provides implementations on top of PyTorch for different SSL algorithms such as FixMatch [21], FlexMatch [24], AdaMatch [22], and MixMatch [20]. We extend USB to support our own format of customized datasets as generated by the HARVEST framework. Furthermore, we implement our own distributed data-parallel (DDP) scheme using `torch.distributed` and the `torchrun` launcher to ensure efficient and fault-tolerant scaling in multi-GPU and multi-node settings. DDP requires adjusting the batch size and learning rate depending on weak scaling or strong scaling settings. We analyze and expand on these concepts later on in the evaluation section IV-D. Finally, after the SSL training task is done, we save the DNN model with the highest accuracy on the evaluation subset.

7) Post-Training Tasks: Once the best DNN model is trained and saved on the HPC system, the user can perform the following tasks using the HARVEST framework:

- Download the trained model in PyTorch format. Trained models can be directly deployed on edge devices to perform inference on the field.
- Generate and export labels for the unlabeled dataset. For this option, we use the trained model to perform inference on the original dataset. We generate a list of indices for each class and map the indices to the full file paths.
- Perform inference on new data. For this option, the user can either upload a single image or a batch of images to be classified using the trained model.

C. Implementing Distributed Data Parallelism (DDP) for SSL

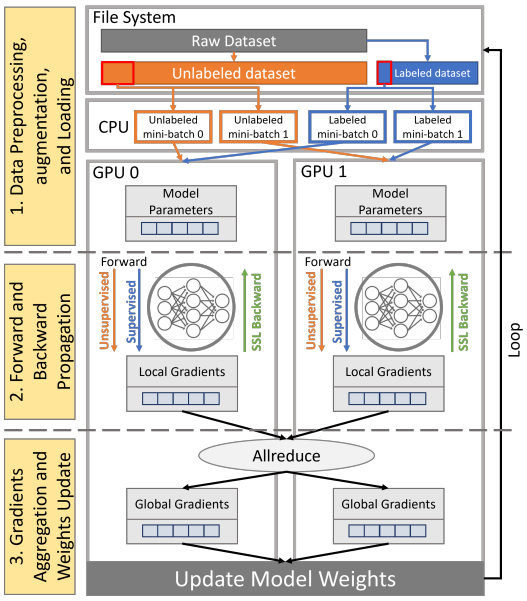


Fig. 5: HARVEST'S approach for performing distributed data parallelism (DDP) with SSL algorithms.

Figure 5 shows our approach to implementing DDP for SSL algorithms. The example in the figure uses two GPUs, but our method generalizes to any number of GPUs. We explain this method in three main steps from top to bottom:

1) Data Preprocessing, Augmentation, and Loading: First, the raw dataset is rearranged into unlabeled and labeled datasets. The unlabeled data contains all data samples without target labels, and the labeled dataset is selected from the raw dataset based on the labels provided by the user. Next, the CPU randomly samples images from the two datasets and performs weak augmentation on the unlabeled data and weak and strong augmentations on the labeled data. Mini-batches are then created for each GPU using the augmented data.

2) Forward and Backward Propagation: There are two forward propagations for the SSL algorithms we use, one for the unsupervised and supervised training each. The loss value for the unsupervised training is calculated by comparing the generated labels for the strongly and weakly augmented images. The supervised training loss is calculated by comparing the generated and target labels. The two loss values are combined and backpropagated to compute the gradients. We end up having different local gradients on each GPU.

3) Gradients Aggregation and Weights Update: To synchronize the models on the two GPUs, we use the Allreduce

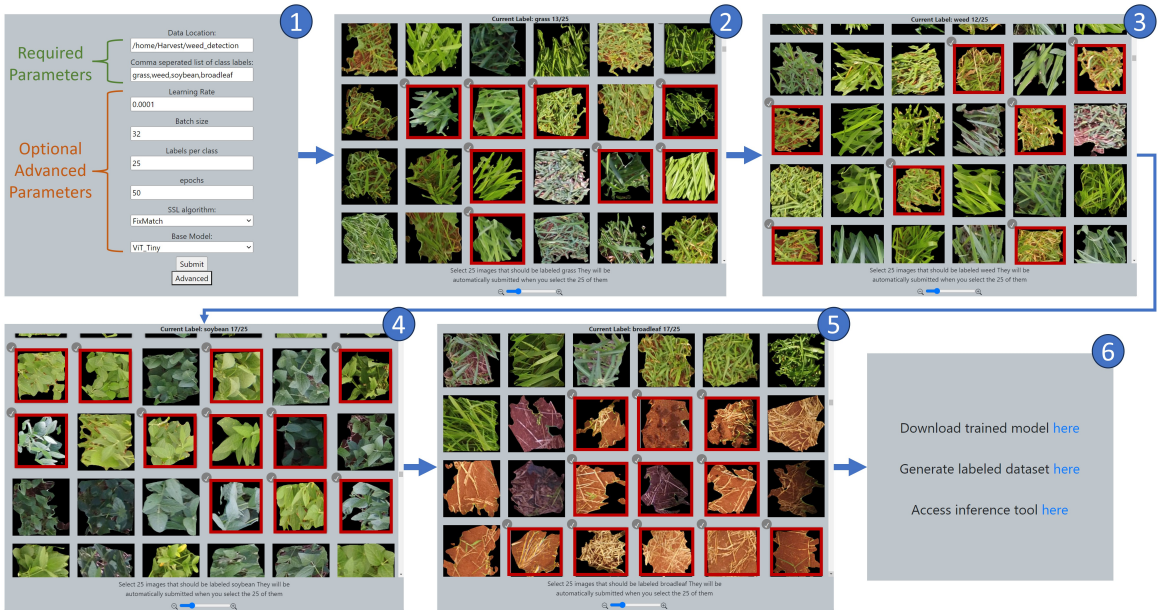


Fig. 6: Example use case with the Weed Detection in Soybean Crops dataset using HARVEST's interactive web interface. In step 1, the user defines their classification use case which consists of 4 classes. In steps 3-5, the user selects images to be labeled for each of the classes. Images highlighted in red with a check mark indicated that images are already labeled by the user. Step 6 shows the post-training tasks.

collective operation, which gathers the local gradients, aggregates them, and distributes them back to each GPU. At this point, we have the same global gradients on both GPUs. We use these global gradients to update the model weights and progress to the next training iteration.

D. HARVEST's Interactive Web Interface

In this subsection, we showcase the workflow of HARVEST from the the user's perspective by following the steps involved in the interactive web-based interface. HARVEST's interface is a tool that is designed to be intuitive and user-friendly, requiring only a few steps from beginner users while offering flexibility for more advanced users to fine-tune their training job. This tool is built on top of Open OnDemand to implement a Flask Python web server that gathers user input, create and submit the job scripts, and serve the subset of images to the user. The front-end uses JavaScript to construct the visual interface and make calls to the back-end Flask server.

Figure 6 shows an example image classification job using the Weed Detection in Soybean [16] dataset. In step 1, the user is asked to define their use case by entering a pointer to their uploaded dataset on the HPC system and a list of target classes to be used for classification. In this example, the user defines 4 classes. Under the advanced tab, users have the option to fine-tune their job by tweaking several training hyperparameters. When the user submits their classification use case, an unsupervised job is scheduled to cluster the data and sample images from each cluster to be presented to the user. In step 2, the user is asked to select 25 images to be labeled for the first class that was defined previously in step 1. Selected images are highlighted in red with a check mark next to them. Once 25 images are labeled for the first class, we move to steps 3-5, in which the user labels 25 images for the rest of the classes. It can be observed that images with similar visual features are already grouped together due to the unsupervised clustering step. After the user provides labels for all of their defined classes, a distributed data-parallel SSL job is scheduled on the HPC system. Finally, in step 6, the user can download the trained model, generate labels for the unlabeled training dataset, or perform inference on new data.

IV. Performance Evaluation and Analysis of the Proposed HARVEST Framework

This section presents the performance evaluation of HARVEST using both publicly available and custom Digital Agriculture datasets. The metrics used for our evaluations are expressed in terms of testing error rate and training time. The baseline that we use for comparison with SSL algorithms is the testing error of fully supervised DNN training. The baseline for the scaling experiments is training using a single GPU. In this section, we first layout the experimental setup, then we explain our findings for the following experiments: 1) Evaluation of SSL algorithms, 2) Evaluation of base DNN models, 3) Scaling analysis on multi-node, 4) Analysis of unsupervised training, 5) Evaluation of custom use cases.

A. Experimental Setup

1) Hardware Setup: We perform our evaluations on the Ohio SuperComputing Center's (OSC's) [30] Ascend cluster. Ascend is a GPU-based HPC system consisting of Power Edge XE 8545 nodes equipped with 2 AMD EPYC 7643 (Milan) processors @2.3 GHz each with 44 usable cores (88 in total), 4 NVIDIA A100 GPUs with 80GB memory, 921GB usable RAM, and 12.8TB NVMe internal storage. The system is interconnected using Mellanox 200 Gbps HDR InfiniBand. We use up to 4 nodes (16 GPUs) for our scaling experiments.

2) Software Setup: The following versions of software packages are used: CUDA v11.6.1, cuDNN v8.3.2, NCCL v2.14.3 [28], Python v3.8.16, PyTorch v1.13.1 [8], HuggingFace v4.21.3 [31], scikit-learn v1.0.2, SCAN [11] and our modified branch of the Unified Semi-supervised Learning Benchmark (USB) [10] based on version 0.3.0.

3) Datasets Description: We use four publicly available Digital Agriculture datasets for our initial evaluations: 1) PlantVillage [15] 2) Weed Detection in Soybean Crops [16], 3) Sugar Cane-Spittle Bug [17], and 4) Fruits-360 [18]. We use the following two custom datasets to mimic the actual use cases of HARVEST: 1) Corn dataset, and 2) Soybean dataset [19]. Although the used datasets are fully labeled, we strip all labels from the data and treat all datasets as fully unlabeled. Table I shows the characteristics of each dataset in terms of the number of classes and total number of samples.

TABLE I: Description of the Digital Agriculture datasets used in the evaluation section

Dataset	Number of classes	Number of samples	Use case
PlantVillage [15]	39	43430	Plant disease classification
Weed Detection in Soybean [16]	4	10635	Weed detection in soybeans
Sugar Cane-Spittle Bug [17]	2	10100	Pest bugs detection
Fruits-360 [18]	81	40998	Fruits classification
Corn dataset [19] (custom)	12	9558	Plant stresses detection
Soybean dataset [19] (custom)	6	5636	Plant stresses detection

B. Evaluation of SSL Algorithms

In this subsection, we evaluate the testing error rate of five state-of-the-art SSL algorithms on four unlabeled versions of the public Digital Agriculture datasets. The SSL algorithms used for this evaluation are MixMatch [20], FixMatch [21], AdaMatch [22], FlexMatch [24], and FreeMatch [23]. We use ViT Tiny model as a base DNN for all datasets and algorithms. We use the testing error rate of supervised training using the fully labeled datasets as a baseline for comparison.

Figure 7a shows the testing error rate for the PlantVillage dataset. Using 5 labeled images per class (195 labeled images in total) yields a testing error rate between 12.67% for MixMatch and 7.04% for FlexMatch. Increasing the number of labeled images per class to 10 decreases the error rate; however, by increasing the number of labels beyond 10 labeled images, we observe only a small improvement (around 1-2%) in the testing error. The best testing error of 2.40% is achieved by the FlexMatch algorithm at 80 labeled images. The baseline fully supervised training achieves a testing error of 0.90%.

Figure 7b shows the testing error rate for Weed Detection in the Soybean Crops dataset. We observe similar trends to the previous dataset. However, the lowest testing error of 0.71% is achieved by the MixMatch algorithm at 80 labels. The baseline of fully supervised training yields a testing error of 0.43%.

Figure 7c shows the testing error rate for the Sugar Cane-Spittle Bug dataset. For this dataset, we observe more consistent improvement in the testing error rate, up to 40 labeled images per class. The best testing error rate is achieved by the FreeMatch algorithm at 10.21% compared to 7.36% for the fully supervised training.

Finally, figure 7d shows the testing error rate for the Fruits-360 dataset. We observe an interesting trend of increasing error rates for several SSL algorithms as we increase the number of labeled images beyond 10 per class. Furthermore, we can see that some SSL algorithms have lower error rates compared to the fully supervised baseline at 1.11%. The best error rate is achieved by the FreeMatch algorithm at 0.81% and 10 labeled images per class. The reason for this trend is due to overfitting on the training data as we increase the number of labels.

Based on this evaluation, we set the default SSL method in HARVEST to FreeMatch as it achieves the best performance on average for the different Digital Agriculture datasets. Additionally, we select the optimal hyperparameters that gave the lowest testing error rate on average including number of labeled images per class, learning rate, and number of epochs.

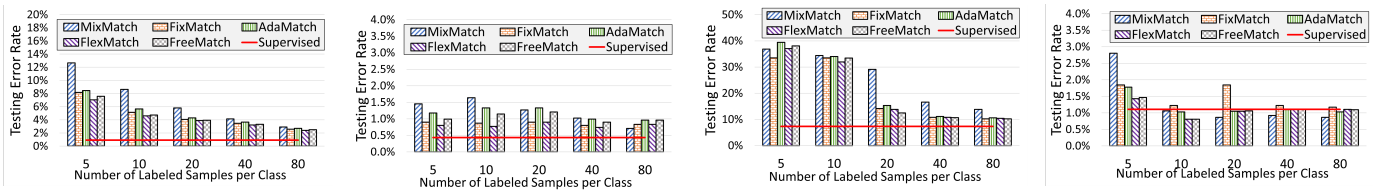
C. Evaluation of Base DNN Models

In this subsection, we evaluate the impact of different base DNNs on the testing error rate. For this evaluation, we chose the Sugar Cane–Spittle Bug dataset, which showed the highest error rate in the previous section. We fix the number of labeled images per class to 80 and use the FreeMatch SSL algorithm. The base DNN models we train on are the Convolutional Neural Network (CNN) ResNet50 [25] and three variants of the vision transformer model ViT [26]. All base DNN models used for this evaluation are pretrained on the ImageNet-1k [14] dataset. Figure 8 shows that ResNet50 yields the highest testing error rate at 16.18%. The ViT variants achieve error rates of 10.21%, 8.47%, and 6.22% for ViT Tiny Patch2_32, ViT Small Patch2_32, and ViT Base Patch16_96, respectively. We observe similar trends for the rest of the SSL algorithms. Therefore, we conclude that increasing the model size achieves better overall accuracy with SSL algorithms.

D. DDP Scaling Analysis on Multi-node

This subsection shows the distributed data-parallel scaling performance using the HARVEST framework on up to 16 NVIDIA A100 GPUs. We perform the training on the Sugar Cane–Spittle Bug dataset using the FreeMatch SSL algorithm with the ViT Small model as a base DNN. Our evaluation consists of two scaling methods:

1) Weak scaling: With weak scaling, we fix the local batch size as we increase the number of GPUs. For example, if the local batch size on a single GPU is 32, and we distribute the training on the 2 GPUs, then the global batch size would be 32*2=64. With weak scaling, we increase the learning rate by a scaling factor corresponding to the number of GPUs.
2) Strong scaling: With strong scaling, we fix the global batch size as we increase the number of GPUs. For



(a) PlantVillage [15] (b) Weed Detection in Soybean [16] (c) Sugar Cane-Spittle Bug [17] (d) Fruits-360 [18]

Fig. 7: Testing error on unlabeled Digital Agriculture dataset using SSL algorithms with different number of labels per class and the ViT Tiny model as the underlying DNN. We use supervised training on the fully labeled versions of the datasets as a baseline for comparison.

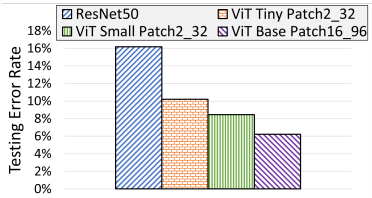


Fig. 8: Comparing the testing error rates for different base DNN models trained on the Sugar Cane–Spittle Bug dataset using the FreeMatch SSL algorithm.

example, if the global batch size on a single GPU is 32, and we distribute the training on 2 GPUs, then the local batch size on each GPU would be 32/2=16. With strong scaling, we fix the value of the learning rate.

Figure 9 shows the training time per epoch and speedup using weak scaling on local batch sizes of 32 and 64. We observe near-linear scaling as we increase the number of GPUs. Training with 16 GPUs yields a speedup of 14.84x and 15.43x for batch sizes 32 and 64, respectively.

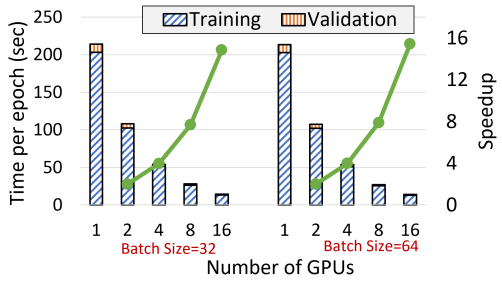


Fig. 9: Weak scaling performance on up to 16 NVIDIA A100 GPUs in terms of time per epoch using the FreeMatch SSL algorithm, ViT Small model as a base DNN, and training on the Sugar Cane–Spittle Bug dataset.

Figure 10 shows the same evaluation but for strong scaling. We see a speedup of 8x for global batch size 32 and 10.51x for global batch size 64. The reason for the performance degradation compared to the weak scaling method is that each GPU will have fewer images to process in parallel as we increase the number of GPUs. In terms of accuracy, we don't see any performance change by comparing weak and strong scaling for different numbers of GPUs. Therefore, we adopt the weak scaling method for performing DDP with HARVEST.

E. Analysis of Unsupervised Training

In this subsection, we analyze the performance of the unsupervised clustering step before presenting the images to be

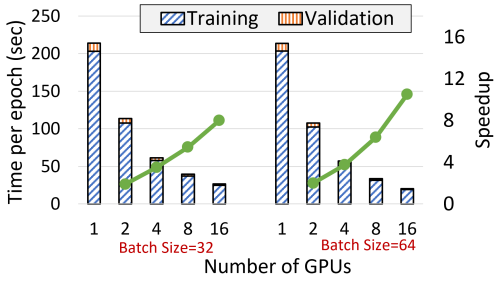


Fig. 10: Strong scaling performance on up to 16 NVIDIA A100 GPUs in terms of time per epoch using the FreeMatch SSL algorithm, ViT Small model as a base DNN, and training on the Sugar Cane–Spittle Bug dataset.

labeled by the user. HARVEST uses the SCAN [11] method to cluster the unlabeled data. For this analysis, we use the Weed Detection in Soybean Crops dataset, which consists of the broadleaf, grass, soil, and soybean classes. We run SCAN for 25 epochs in total. Figure 11 shows the confusion matrix generated by SCAN. Numerical values in white along the diagonal axis indicate the accuracy for clustering the four classes. The clusters for the soil and soybean classes show low accuracy, whereas the broadleaf and grass classes show better accuracy. The reason for the lower accuracy is due to similar visual features in the soil and soybean classes. It should be noted that other datasets in our evaluation showed below 60% clustering accuracy due to more complex visual features and similar classes. Therefore, while the unsupervised clustering step may be helpful to assist the user in performing faster labeling for some use cases by grouping simillar images, we conclude that it is an optional step, with the alternative being random sampling from the dataset.

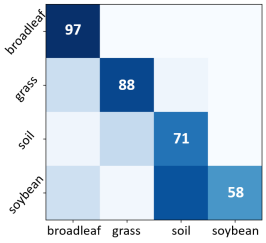


Fig. 11: The confusion matrix for the optional unsupervised training step using SCAN on the Weed Detection in Soybean Crops dataset.

F. Evaluation for Custom Digital Agriculture Use Cases

1) Problem Description: In [19], Frank et al. address plant stress identification for protecting crops through the growing

season. A stress factor is an external condition that negatively affects plants' growth, development, or productivity. The authors curate several datasets for different types of crops and categorize stress factors into classes. Two of the datasets they generate are for corn and soybean crops. They use supervised DL combined with hierarchical confidence thresholding to train the ResNet-18 from scratch on their different use cases. Their trained baseline model, without confidence thresholding, achieves 68.8% accuracy on the Corn dataset and 80.0% accuracy on the Soybean dataset. In the following subsection, we evaluate the performance of HARVEST on these two use cases using only a small subset of labeled images.

2) Performance on the Corn and Soybean datasets using HARVEST: We use HARVEST to train on unlabeled versions of the Soybean and Corn datasets. We first label 80 images per class for each dataset. Next, we use the FreeMatch SSL algorithm with the Vit Base model to train on both the large unlabeled datasets and small labeled subsets for a total of 60 epochs. Table II shows the testing accuracy, precision, recall, and F1 score for training on both datasets. HARVEST achieves 97.08% and 93.07% accuracy for the Corn and Soybean datasets, respectively. The significant boost in performance we observe compared to the methods used in section IV-F1 is due to using a more capable and pre-trained ViT Base model on ImageNet-1k compared to training ResNet-18 from scratch.

TABLE II: HARVEST's performance on the Soybean and Corn datasets using only 80 labels per class with FreeMatch as an SSL algorithm and the ViT Base model as base DNN.

Dataset	Accuracy	Precision	Recall	F1 Score
Corn Dataset	97.08%	91.77%	95.43%	92.61%
Soybean Dataset	93.07%	88.64%	92.40%	89.61%

Furthermore, we use HARVEST's scaling features to distribute the training on up to 16 A100 GPUs. Figure 12 shows the total training time and speedup for the Corn dataset. The time needed to train using a single GPU is 7.8 hours. Using 16 GPUs, HARVEST reduces the total training time to 31 minutes, achieving a 15.19x speedup and maintaining the same accuracy. Figure 13 shows the same evaluation for the Soybean dataset. The time needed to train using a single GPU is 4.6 hours. HARVEST reduces the total training time to 20 minutes on 16 GPUs, achieving a 14.25x speedup while also maintaining the same accuracy.

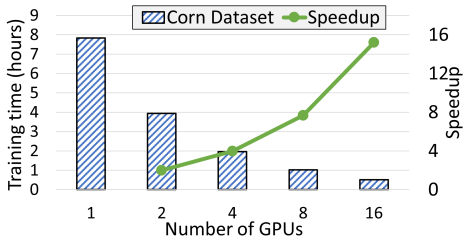


Fig. 12: Scaling performance on up to 16 NVIDIA A100 GPUs for training the Corn dataset [19] using the FreeMatch algorithm and ViT Base Patch16_224 as a base DNN model for a total of 60 epochs.

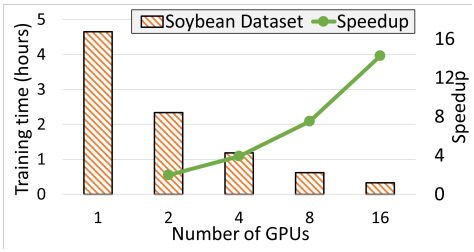


Fig. 13: Scaling performance on up to 16 NVIDIA A100 GPUs for training the Soybean dataset [19] using the FreeMatch algorithm and ViT Base Patch16_224 as a base DNN model for a total of 60 epochs.

V. Related Work

Several designs exist in the literature [20]–[24] that show the potential of using semi-supervised learning (SSL) for image classification as an alternative solution for supervised learning while maintaining high training and testing accuracies. HARVEST extends these solutions by combining unsupervised learning and SSL methods, using distributed-data parallelism (DDP), and enabling interactive labeling. Many commercial and open-source image annotation tools exist online [32]–[34]; however, these tools are disconnected from DL training, requiring additional overhead to transfer, reformat, and preprocess data to be used by DNNs. Other studies [35]–[37] use unsupervised learning with pre-trained DNNs or generative pretraining as an automated solution for image annotation. While these solutions may be useful in scenarios where classes exist in the training data or are highly distinct, they achieve low accuracies for complicated and domain-specific datasets. The authors of [38] use few-shot learning, which is applicable for scenarios where images are labeled but the number of samples is limited. This approach is practical when unseen classes are frequently encountered in new data.

VI. Conclusions and Future Work

In this paper, we proposed HARVEST—a distributed framework for image classification that employs state-of-the-art semi-supervised learning (SSL) algorithms for training accurate Deep Neural Networks (DNNs) on High Performance Computing (HPC) systems using only a small subset of labeled images. Our work comes as a part of the Intelligent Cyberinfrastructure with Computational Learning in the Environment (ICICLE) [13] AI institute. HARVEST emphasizes on ICICLE's core mission of democratizing AI by integrating an interactive and intuitive web-based interface that allows domain experts with no prior DL or HPC expertise to create customized scalable solutions for their image classification use cases. We used Open OnDemand [12] to connect HARVEST's interface to the underlying HPC systems. HARVEST mitigates the data labeling bottleneck by using an advanced workflow that combines unsupervised and SSL solutions, including SCAN [11] and Unified Semi-supervised learning Benchmark (USB) [10]. We implemented distributed data parallelism (DDP) using PyTorch [8] and `torch.distributed` [9] for efficient and fault-tolerant DNN training. We conducted a comprehensive evaluation and analysis using Digital Agriculture image classification use cases as an example domain

that can benefit from HARVEST. Our evaluations showed that HARVEST can deliver high accuracies within 3% of fully supervised training using less than 80 labeled samples per class. Furthermore, we showed that HARVEST can scale efficiently, delivering up to 15.43x speedup and reducing the training time from 7.8 hours on a single NVIDIA A100 GPU to 31 minutes by using DDP on 16 GPUs. *To the best of our knowledge, HARVEST is the first framework that allows end-users to perform interactive labeling and distributed training using state-of-the-art SSL algorithms.* For future work, we plan to extend HARVEST's pipeline to support easy inferencing deployment of trained DNNs on edge devices, which may have memory and computational constraints. Additionally, since HARVEST is a generalized and portable framework, we plan to test it on Cloud systems and other scientific domains.

REFERENCES

[1] K. Virts, A. Shirey, G. Priftis, K. Ankur, M. Ramasubramanian, H. Muhammad, A. Acharya, and R. Ramachandran, "A quantitative analysis on the use of supervised machine learning in earth science," in *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, pp. 2252–2255, 2020.

[2] M. A. Wani, F. A. Bhat, S. Afzal, and A. I. Khan, *Basics of Supervised Deep Learning*, pp. 13–29. Singapore: Springer Singapore, 2020.

[3] J. Karhunen, T. Raiko, and K. Cho, "Chapter 7 - unsupervised deep learning: A short review," in *Advances in Independent Component Analysis and Learning Machines* (E. Bingham, S. Kaski, J. Laaksonen, and J. Lampinen, eds.), pp. 125–142, Academic Press, 2015.

[4] M. Alloghani, D. Al-Jumeily, J. Mustafina, A. Hussain, and A. J. Aljaaf, *A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science*, pp. 3–21. Cham: Springer International Publishing, 2020.

[5] J. E. van Engelen and H. H. Hoos, "A survey on semi-supervised learning," *Machine Learning*, vol. 109, pp. 373–440, Feb 2020.

[6] A. Caruso, S. Chessa, S. Escolar, J. Barba, and J. C. López, "Collection of data with drones in precision agriculture: Analytical model and lora case study," *IEEE Internet of Things Journal*, vol. 8, no. 22, pp. 16692–16704, 2021.

[7] M. S. Norouzzadeh, A. Nguyen, M. Kosmala, A. Swanson, M. S. Palmer, C. Packer, and J. Clune, "Automatically identifying, counting, and describing wild animals in camera-trap images with deep learning," *Proceedings of the National Academy of Sciences*, vol. 115, no. 25, pp. E5716–E5725, 2018.

[8] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems 32*, pp. 8024–8035, Curran Associates, Inc., 2019.

[9] PyTorch, "torch.distributed." https://pytorch.org/docs/stable/distributed.html, 2021. [Online; accessed November 17, 2023].

[10] Y. Wang, H. Chen, Y. Fan, W. Sun, R. Tao, W. Hou, R. Wang, L. Yang, Z. Zhou, L.-Z. Guo, H. Qi, Z. Wu, Y.-F. Li, S. Nakamura, W. Ye, M. Savvides, B. Raj, T. Shinozaki, B. Schiele, J. Wang, X. Xie, and Y. Zhang, "Usb: A unified semi-supervised learning benchmark for classification," in *Thirty-sixth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2022.

[11] W. Van Gansbeke, S. Vandenhende, S. Georgoulis, M. Proesmans, and L. Van Gool, "Scan: Learning to classify images without labels," in *Proceedings of the European Conference on Computer Vision*, 2020.

[12] D. Hudak, D. Johnson, A. Chalker, J. Nicklas, E. Franz, T. Dockendorf, and B. L. McMichael, "Open ondemand: A web-based client portal for hpc centers," *Journal of Open Source Software*, vol. 3, no. 25, p. 622, 2018.

[13] "Intelligent Cyberinfrastructure with Computational Learning in the Environment (ICICLE)." https://icicle.osu.edu/.

[14] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[15] A. Bruno, D. Moroni, R. Dainelli, L. Rocchi, S. Morelli, E. Ferrari, P. Toscano, and M. Martinelli, "Improving plant disease classification by adaptive minimal ensembling," *Front Artif Intell*, 2022. PMID: 36160929; PMCID: PMC9499023.

[16] A. dos Santos Ferreira, D. Matte Freitas, G. Gonçalves da Silva, H. Pistori, and M. Theophilo Folhes, "Weed detection in soybean crops using convnets," *Computers and Electronics in Agriculture*, vol. 143, pp. 314–324, 2017.

[17] C. Vasquez and C. Castiblanco, "Sugar cane - spittle bug." https://data.mendeley.com/datasets/pwprmck9h5/1.

[18] H. Mureşan and M. Oltean, "Fruit recognition from images using deep learning," 2021.

[19] L. Frank, C. Wiegman, J. Davis, and S. Shearer, "Confidence-driven hierarchical classification of cultivated plant stresses," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 2503–2512, January 2021.

[20] D. Berthelot, N. Carlini, I. Goodfellow, N. Papernot, A. Oliver, and C. Raffel, "Mixmatch: A holistic approach to semi-supervised learning," 2019.

[21] K. Sohn, D. Berthelot, C.-L. Li, Z. Zhang, N. Carlini, E. D. Cubuk, A. Kurakin, H. Zhang, and C. Raffel, "Fixmatch: Simplifying semi-supervised learning with consistency and confidence," 2020.

[22] D. Berthelot, R. Roelofs, K. Sohn, N. Carlini, and A. Kurakin, "Adamatch: A unified approach to semi-supervised learning and domain adaptation," 2022.

[23] Y. Wang, H. Chen, Q. Heng, W. Hou, Y. Fan, Z. Wu, J. Wang, M. Savvides, T. Shinozaki, B. Raj, B. Schiele, and X. Xie, "Freematch: Self-adaptive thresholding for semi-supervised learning," 2023.

[24] B. Zhang, Y. Wang, W. Hou, H. Wu, J. Wang, M. Okumura, and T. Shinozaki, "Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling," 2022.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[26] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *CoRR*, vol. abs/2010.11929, 2020.

[27] Message Passing Interface Forum, *MPI: A Message-Passing Interface Standard*, Mar 1994.

[28] NVIDIA, "Nvidia collective communication library (nccl)." https://developer.nvidia.com/nccl.

[29] A. B. Yoo, M. A. Jette, and M. Grondona, "Slurm: Simple linux utility for resource management," in *Job Scheduling Strategies for Parallel Processing* (D. Feitelson, L. Rudolph, and U. Schwiegelshohn, eds.), (Berlin, Heidelberg), pp. 44–60, Springer Berlin Heidelberg, 2003.

[30] O. S. Center, "Ohio supercomputer center," 1987.

[31] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. L. Scao, S. Gugger, M. Drame, Q. Lhoest, and A. M. Rush, "Huggingface's transformers: State-of-the-art natural language processing," 2020.

[32] "V7 - the ai data engine to computer vision and generative ai." https://www.v7labs.com/.

[33] "Superannotate: The ultimate training data platform for ai." https://www.superannotate.com/.

[34] Tzutalin, "Labelimg. git code (2015)." https://github.com/HumanSignal/labelImg.

[35] Y. Chen, L. Liu, J. Tao, X. Chen, R. Xia, Q. Zhang, J. Xiong, K. Yang, and J. Xie, "The image annotation algorithm using convolutional features from intermediate layer of deep learning," *Multimedia Tools and Applications*, vol. 80, pp. 4237–4261, Jan 2021.

[36] X. Ke, J. Zou, and Y. Niu, "End-to-end automatic image annotation based on deep cnn and multi-label data augmentation," *IEEE Transactions on Multimedia*, vol. 21, no. 8, pp. 2093–2106, 2019.

[37] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever, "Generative pretraining from pixels," in *Proceedings of the 37th International Conference on Machine Learning* (H. D. III and A. Singh, eds.), vol. 119 of *Proceedings of Machine Learning Research*, pp. 1691–1703, PMLR, 13–18 Jul 2020.

[38] S. X. Hu, D. Li, J. Stühmer, M. Kim, and T. M. Hospedales, "Pushing the limits of simple pipelines for few-shot learning: External data and fine-tuning make a difference," 2022.