Domain-Aware Model Training as a Service for Use-Inspired Models

1st Zichen Zhang
The Ohio State University
Columbus, USA
0000-0002-8919-6243

2nd Christopher Stewart
The Ohio State University
Columbus, USA
0000-0002-2860-7889

***Abstract*—Use-inspired artificial intelligence (AI) tailors deep-learning models for image processing tasks in targeted scientific domains. These use-inspired models meet domain requirements for accuracy while parsimoniously using compact and efficient model architectures needed for inference in the field. However, before settling upon a model, domain experts repeatedly train and test models over a wide range of hyperparameters, contextual settings, and data configurations, making model training bespoke, time-consuming, and costly. Model-training-as-a-Service (MTaaS), i.e., cloud services designed for generic training workloads, can reduce training costs, but domain-aware designs and runtime adaptations could yield further reductions. This paper characterizes the potential for domain-aware design and runtime adaptation for MTaaS in digital agriculture. First, we studied the time to train models for 10 use-inspired agricultural datasets using pre-trained model weights derived from other agricultural datasets versus pre-trained weights derived from ImageNet, a widely used benchmark. Using agricultural datasets sped training time by up to 2X for some datasets, but provided modest speedups (< 1.07) in the common case; Choosing the right dataset is critical. Next, we present an approach to predict training time given domain-aware pre-trained weights. Our predictions are strongly correlated with training time (r = 0.93). Finally, we studied the use of domain-aware pre-trained weights in a MTaaS under Poisson and bursty arrival patterns for training tasks. Under bursty arrivals and tight memory constraints, domain-aware MTaaS reduced training time by 2.8X and 12.2X compared to model training using pre-trained ImageNet weights and from scratch, respectively.**

***Index Terms*—Transfer Learning, Cloud Service, Dataset Similarity, Neural Network, Distributed System, Model-Training-as-a-Service**

I. Introduction

Use-inspired artificial intelligence (AI) solves domain-specific image processing and classification problems [11], [18], [19], [26]. It relies on image processing via deep-learning models, e.g., convolutional neural networks [31] and vision transformer models [7]. Unlike deep-learning models intended to capture generalized intelligence, use-inspired models must meet accuracy and performance goals for a domain while parsimoniously managing resources to reduce costs. The demand and availability of use-inspired models are growing because, in many domains, these models have economic value when deployed in practice. For example, in digital agriculture, use-inspired models classify crop diseases [38], detect severe crop defoliation [39], assess growth stages, and inform farmers about nutrient levels in the soil [26].

Use-inspired models are created using purpose-specific datasets labeled by domain experts. The model training process uses these labeled images to adjust neural network weights, eventually converging on a model (i.e., weights and architecture) that accepts unseen images and outputs domain-specific classifications. In practice, robust use-inspired models usable in the field require multiple model-training runs to find efficacious training parameters and each training run requires costly GPUs. The total cost for model training mitigates the potential for widespread adoption of use-inspired AI.

Cloud-based Model Training as a Service (MTaaS)reduces training costs by sharing compute resources across generic deep-learning workloads [4], [10], [21]. These platforms train deep learning models for image processing efficiently and do not require upfront investments in hardware and infrastructure. However, model training workloads within a domain may have distinguishing traits that can be exploited to further reduce costs. This paper aims to characterize the potential for such domain-aware designs in an MTaaS.

Model training platforms can initiate neural network weights in many ways. To build fresh models influenced solely by the training datasets, the platform can initialize weights as zero. Alternatively, platforms employing transfer learning initialize weights using pre-trained weights from models trained on other datasets. With pre-trained weights, datasets labeled by domain experts are used to fine-tune existing model weights for specific tasks [28], [30], [37]. ImageNet [5], a large dataset containing over 14 million labeled images across 1,000 object categories, is widely used for pre-trained weights. It has been shown to be effective on model training tasks involving large and general-purposed datasets, making it a go-to choice for transfer learning in many domains [17]. However, ImageNet's pre-trained weights reflect knowledge on generalized image processing tasks that may differ from domain-specific tasks. For example, in digital agriculture, image processing tasks involve leaves, crops, canopies, discoloration patterns, and disease semantics that are not well-represented in ImageNet. In addition, the distinguishing features between classes in digital agriculture are subtle. Domain-aware transfer learning addresses these challenges by using pre-trained weights from other digital agriculture datasets.

This paper explores key issues motivating domain-aware MTaaS. First, we compared 10 digital agriculture datasets to

datasets widely used for general-purpose image processing. We found that the model training performance using agricultural pre-trained weights outperformed that using general-purpose benchmark pre-trained weight in both accuracy and training time. We further discovered that even within a domain, the performance of pre-trained models varies across datasets. For that finding, we proposed a domain-aware approach that uses dataset similarity distance (DSD) to evaluate the effectiveness of transfer learning given pre-trained weights from agricultural datasets, which can run fast with only CPUs. Finally, we developed design and runtime adaptations for domain-aware MTaaS and evaluated upon Poison and bursty workloads with a limited in-memory storage for pre-trained models. Results show that our method can speedup the domain-aware model training workloads up to 3.5 times faster compared to the benchmark using pre-trained ImageNet weight. With the help of domain-aware MTaaS, the effectiveness of use-inspired model training can further reduce training cost an provide better understanding of domain relevance and transfer learning.

The remainder of this paper is organized as follows. Section II presents recent studies on comparing digital agriculture datasets to widely used image processing benchmarks. Section III describes dataset similarity and our predictive model on the efficacy of pre-trained weights. Section IV analyzes the potential for domain-aware design and adaptation in MTaaS, assessing the impact of bursty and non-stationary request arrival patterns. Section V describes related work in MTaaS, use-inspired AI, and transfer learning. Section VI discusses future work and conclusion.

II. Background

In the field of computer vision, researchers have traditionally relied on general-purpose datasets such as ImageNet to benchmark the performance of neural network architectures. These datasets have been widely used to compare and evaluate various models, such as VGG16, EfficientNet, and ResNet50 [13], [31], [32]. However, the effectiveness of these models on domain-specific datasets, particularly in the field of digital agriculture, has not been extensively explored.

Recent studies have highlighted the importance of domain-specific evaluation [25]. Ockerman *et al.* explored the need for individual dataset profiling in digital agriculture by analyzing the model training performance among common general-purpose dataset and domain-specific datasets. They studied different neural network models trained on popular vision benchmarks and digital agricultural datasets, comparing models trained on each dataset by ranking their performance on all the other datasets using Euclidean distance based on accuracy and loss across the entire training process. From the results they noticed that the performance ranking order of models trained on benchmark datasets varies from that trained on digital agricultural datasets, showing that domain-specific datasets display fundamentally different properties than general-purpose datasets. These domain-specific datasets represent unique challenges and characteristics that may not be adequately captured by general-purpose datasets.

Despite the growing recognition of the importance of dataset-specific evaluation, there is still a lack of comprehensive studies utilizing the performance difference across general-purpose and domain-specific datasets in the field of computer vision.To address this gap, our study aims to investigate the performance of three widely used DNN architectures (MobileNet, EfficientNet, and ResNet50) on both general-purpose datasets (ImageNet) and digital agricultural datasets and try to quantify the model training difference using dataset similarity. Our work builds upon the existing literature by providing a comprehensive analysis of model performance across a diverse set of datasets, spanning both general-purpose and domain-specific categories. By quantifying the distances between datasets and correlating dataset similarity distance with efficacy of model training performance, we aim to provide insights into the biases and limitations of relying solely on classical datasets for benchmarking purposes. This study contributes to the growing body of research emphasizing the importance of dataset-specific evaluation and aims to foster a more nuanced understanding of model performance in the context of diverse computer vision tasks.

III. Modeling Pre-Trained Weights Efficacy with Domain-Aware Datasets

In this section, we illustrate how we design our system using pre-trained weights based on domain relevance to profile model training performance. First, we conducted experiments to compare the difference in the model training performance using domain-specific agricultural and general-purpose pre-trained weights. Then, based on our findings, we further proposed a dataset similarity distance ranking algorithm to characterize model training performance using ImageNet as a benchmark. Last but not least, we developed a cloud-based MTaaS system with DSD algorithm. For model training jobs on the cloud, the system is designed to utilize the limited cache space and dataset similarity to speedup model training processes.

Model training with pre-trained weights has been long proved to be an efficient way to boost performance and been commonly adopted in model training tasks. However, how to choose a proper neural network architecture and a transferable knowledge (pre-trained weight) from a most relevant domain available without conducting the entire training processes remains a great challenge. Thus, the most common way is to use the general-purpose ImageNet pre-trained weight by default. To explore domain-aware model training performance, we ran experiments with both agricultural pre-trained weights and ImageNet weight for model training tasks on agricultural dataset. We first trained a weed leaf counting (WLC) pre-trained weight by training a ResNet50 model on the weed leaf counting dataset from scratch. Then we did model training processes for all other agricultural datasets we collected using both WLC and ImageNet pre-trained weight. During the model training process, to make a fair performance comparison, we

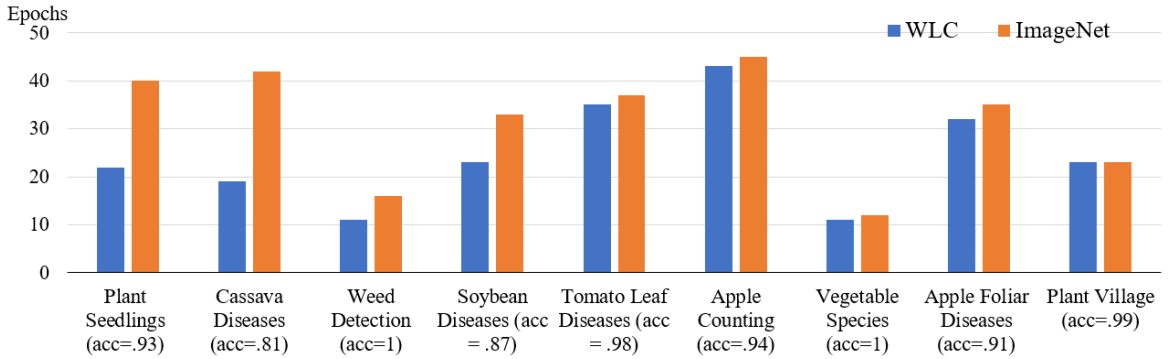


Fig. 1. Model training performance comparison on training epochs with weed leaf counting (WLC) and ImageNet pre-trained weight.

used early stopping techniques to report the training epochs at a close accuracy for both pre-trained weights, as shown in Figure 2. The comparison is reported in Figure 1, for most agricultural datasets, model training using WLC pre-trained weight converge much faster than that using ImageNet pre-trained weight. It's also worth noting that the efficiency for model training using domain-specific pre-trained weight over that using general-purpose pre-trained weight varies among agricultural datasets. For example, on cassava dataset, model training using WLC pre-trained weight converged at a 81% accuracy on 19 epochs while model training using ImageNet pre-trained weight converged at the same accuracy on 42 epochs, which yields a 54.76% speedup. However, that performance boost is not seen in model training processes on some other agricultural datasets, such as plant village dataset where the performance for model training using WLC and ImageNet pre-trained weights are very close.

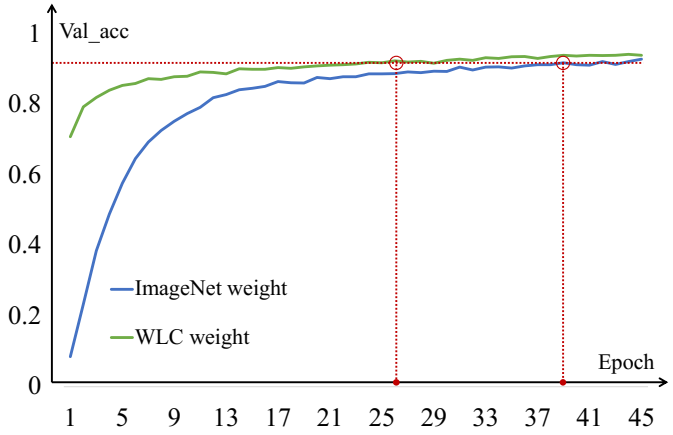


Fig. 2. Comparison of validation accuracy learning curves during model training process. The blue line represents transfer learning with ImageNet-1k weight. While the green line indicates that with the Weed Leaf Counting weight.

To characterize the model training performance using different pre-trained weights, we first adapted and modified the OTDD algorithm [3] to measure the correlation of dataset similarity and efficiency of model training with pre-trained weights. Below is a mathematical equation for the OTDD algorithm:

$$OTDD(\mathcal{D}_A, \mathcal{D}_B) = min_{\pi \in \prod(P_A, P_B)} \int_{\mathcal{Z} \times \mathcal{Z}} d(z, z') d\pi(z, z') \quad (1)$$

The distance between two datasets can be calculated by Equation (1). z represents a feature-label pair in one dataset while z' represents a pair in the other dataset. $d(z, z')$ denotes the distance between two pairs belonging to two datasets. The image feature distance can be assessed by just computing the pixel difference in two images. To compute the label distance, we need to treat the labels as probability distributions first. P_A and P_B denote the conditional probability distributions used to represent labels in the datasets: $P_y = P(X|Y = y)$. Then, the Earth Mover's Distance (Wasserstein Distance) can be use for computing the label distance. Therefore, the distance between two pairs can be computed by the following equation:

$$d(z, z') = \sqrt{d(x, x')^2 + W(P_y, P'_y)^2} \quad (2)$$

We modified OTDD in our method with the goal of making the process of getting dataset similarity measurements fast, memory friendly, and accurate. Our adaptation is as follows: Given two datasets D_A and D_B, first we randomly shuffle and evenly partition them into N_A and N_B units over a constant $unit\_size$, defaulting at 2000. Then select k_A and k_B units from all the units and run OTDD cross all unit combinations. The selection of k varies depending on N, the total number of units of a dataset. It's calculated by a stair-shaped piecewise function. After we have the distance values from $k_a \times k_b$ OTDD functions, the final distance of two datasets will be the average of all values.

OTDD algorithm is good at measuring dataset similarity. However, it lacks the ability to correlate dataset similarity with model training efficiency, as shown in Figure 4. To address the problem, we proposed a novel DSD ranking algorithm based on OTDD that given a new dataset and existing datasets, DSD algorithm will rank the pre-trained weights related to the efficiency of model training performance with the ImageNet weight as a baseline. As illustrated in Figure 3, the DSD ranking algorithm works as follows: First, for a new dataset, it will compute the dataset distance with all existing domain-specific datasets using the modified OTDD algorithm. Once it

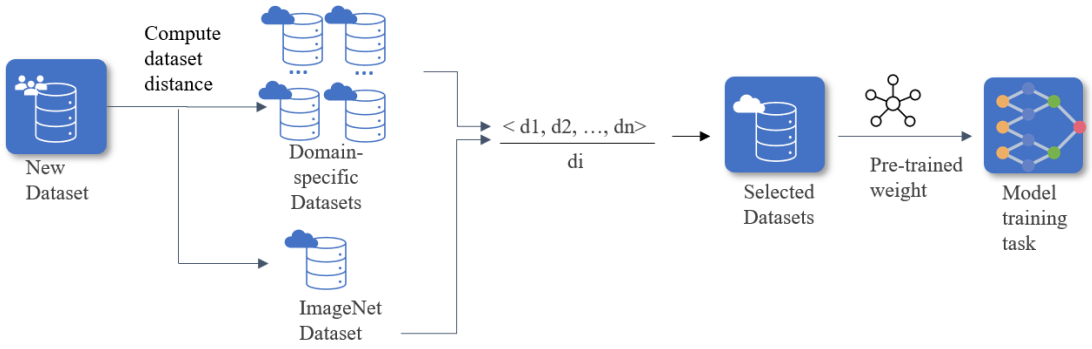


Fig. 3. A systematic overview of the model training process with our DSD algorithm. Given a new dataset, the DSD ranking algorithm will compute dataset distances between the new dataset and all datasets in the system and select the most domain-relevant dataset based on the results. Then the pre-trained weight from selected dataset will be used for model training task on the new dataset.

gets all dataset distances, it will divide each number over the dataset distance of the new dataset to ImageNet dataset. The result vector containing relative dataset distances will be normalized and sorted. Smaller values indicate a relatively better model training performance if trained with the corresponding pre-trained weights. After the pre-trained weight is selected from the DSD ranking result, model training process will be carried out for the new dataset with the selected weight. With the DSD ranking algorithm to characterize the efficiency of model training processes using pre-trained weights, we are able to find the most domain-relevant pre-trained weight from the dataset to accelerate the model training process.

IV. Experiments

We designed and conducted experiments to evaluate our method in several aspects. In Sections IV-A and IV-B, we introduced the datasets and neural network architectures used in our experiments. In Section IV-C, the baseline approach for evaluating our method is discussed. In Section IV-D, we conducted experiments and discussed the correlation between model training performance and the DSD ranking algorithm. In Section IV-E, we studied the use case of our system in a cloud service. Last but not least, in Section IV-F, we illustrated how we used the DSD algorithm to reflect on the performance of different network architectures. And in Section IV-G, we demonstrated the robustness of our system.

A. Data Preparation

To explore the correlation between dataset similarity and transfer learning and to evaluate our model profiling system, we used ten publicly available image classification datasets in agriculture field and ImageNet-1k dataset to conduct our experiments, as shown in Table I. Datasets were obtained from different resources and in RGB format. Some of them are from publicly available websites used in previous researches, while others from all sorts of machine learning challenges. For datasets acquired from challenges, only training sets are labeled. Since we propose our methods in a supervised manner in this paper, datasets with only training sets labeled will be randomly shuffled and then split into training and test sets in a 7:3 ratio.

B. Neural Network Architectures

We used three types of neural network architectures on the open-sourced machine learning platform, Tensorflow [1], to conduct our experiments: ResNet [13], EfficientNet [32], and MobileNet [16].

ResNet. He *et al.* proposed a deep residual learning framework, called ResNet, to ease the training process on very deep neural network models, which allows the usage of much deeper networks [13]. ResNet has many variances with different amount of layers. In this paper, we used ResNet50 [14], the residual network with 50 layers.

EfficientNet. EfficientNet was first invented by Tan and Le motivated from a new scaling up ConvNets idea that used *compound coefficient* to uniformly scale up all dimensions of a neural network architecture [32]. We used EfficientNetV2S [33] to conduct experiments.

MobileNet. For the purpose of building light weight neural networks that can be embedded on mobile devices, Howard *et al.* introduced MobileNet featuring depthwise separable convolutions [16]. Results showed that the MobileNet models are able to achieve a good accuracy while maintaining a relatively small model size. We used MobileNetV3L [15], an improved version of the model in the original paper, in our experiments.

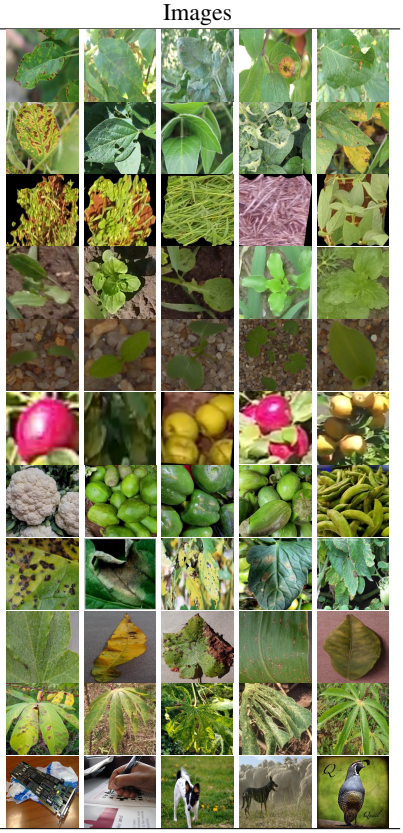
C. Baseline

For the baseline approach, we selected the widely used ImageNet pre-trained weight for performance comparisons in model training tasks through transfer learning and the ImageNet-1k [29] dataset for computing dataset similarity with datasets mentioned above in Section IV-A.

D. Correlation between DSD and Model Training

1) Model Training Performance Comparison: We designed an evaluation method to compare the transfer learning performance with the baseline approach. As shown in Figure 2, performance comparisons are made on the learning curves of validation accuracy. We compare the model performance

TABLE I
WE USE 10 AGRICULTURAL DATASETS AND THE IMAGENET-1K DATASET TO EVALUATE OUR SYSTEM.

Dataset	Number of Classes	Number of Images	Dataset Purpose	Sample Images
Apple Foliar Diseases [35]	6	18,632	Diseases Classification	
Soybean Diseases [39]	6	6,637	Diseases Classification	
Weed Detection [6]	4	15,336	Diseases Classification	
Weed Leaf Counting [34]	9	9,372	Crop Counting	
Plant Seedlings [9]	12	5,539	Species Classification	
Apple Counting [12]	7	70,867	Crop Counting	
Vegetable Species [2]	15	21,000	Species Classification	
Tomato Leaf Diseases [22]	7	21,758	Diseases Classification	
Plant Village [8]	39	61,486	Species & Diseases Classification	
Cassava Diseases [23]	5	5,656	Diseases Classification	
ImageNet-1k [29]	1,000	1,321,167	General	

through different transfer learning configurations with the baseline approach by measuring the number of epochs they take to get to the peak validation accuracy. Therefore, the comparison is only fair if the peak validation accuracy they get to are almost the same. For all our experiments, we define that two accuracy values are considered the same if they are within a 0.1% range.

For comparing a network model with the baseline model trained on the same dataset, first, we train both network models with a fixed number of epochs to get enough data for comparison. Then after validation accuracy learning curves are acquired, we simulate a normal model training process by applying an early stopping technique with a patience of 5. Second, if the peak accuracy of both network models fall into the 0.1% range, we record the number of epochs they take. If not, we will use the peak accuracy of the baseline model from the two network models as a reference to select the peak accuracy from the other model that is closest to it. Third, we compute the epoch ratio as a measurement on model training performance.

In Figure 1, we demonstrated the transfer learning performance comparisons trained on ResNet50 using Weed Leaf Counting and ImageNet-1k pre-trained weights. For each dataset, we recorded the number of epochs they took at getting a very similar validation accuracy. All results were averaged from three-fold cross-validation. Among the results, Plant Seedlings gets the most transfer learning acceleration of 45% faster with Weed Leaf Counting pre-trained weight compared to ImageNet-1k weight. While Plant Village only gets less 1% acceleration.

2) Domain Relevance Experiments: We further conducted experiments on exploring the correlation of domain relevance with transfer learning in four steps. First, we randomly selected the Weed Leaf Counting dataset as the base dataset to conduct our experiments. Weed Leaf Counting pre-trained weights for three network architectures were obtained from model training processes. Second, for each network architecture and remaining dataset, we conducted model training tasks through transfer learning using both Weed Leaf Counting and ImageNet-1k pre-trained weights, resulting in a total of 27 pairs of transfer learning configurations. Furthermore, all model training results were averaged from three-fold cross validation. For each pair of transfer learning configuration, model training performance was recorded by computing the epoch ratio, as shown in Equation (3). f_{ER} denotes the function to compute epoch ratio, f_{NE} denotes the function to obtain the number of epochs a model training process takes

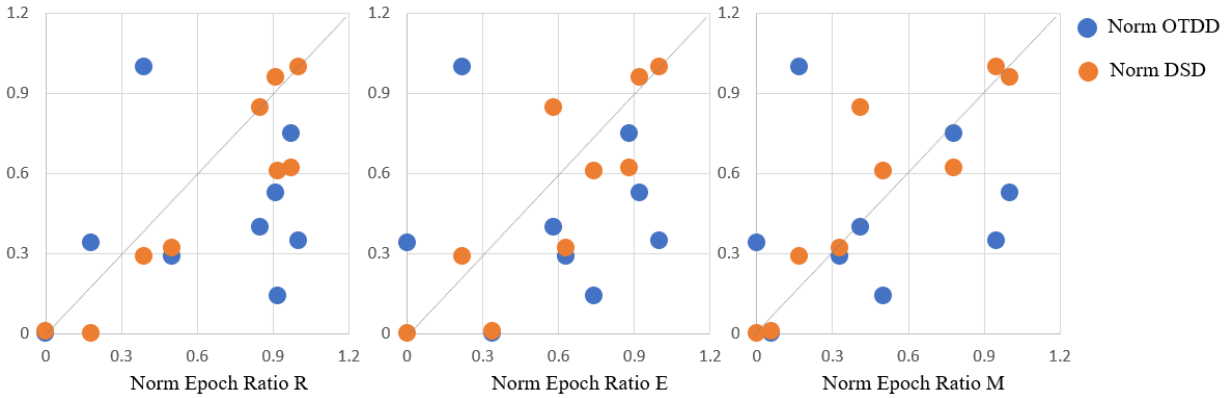


Fig. 4. Comparing DSD with OTDD for the correlation between dataset similarity and performance boost of model training with different network architectures with regard to the weed leaf counting dataset. R, E, M respectively represent ResNet50 (R), EfficientNetV2S (E), and MobileNetV3L (M). The orange dots mean the normalized DSD of datasets with the weed leaf counting dataset, while the blue dots represent the normalized OTDD of datasets with the weed leaf counting dataset. X-axis is the normalized dataset distance while the y-axis is the epoch ratio of model training using weed leaf counting pre-trained weight over that using ImageNet weight.

to get to the peak validation accuracy.

$$f_{ER}(\mathcal{D}) = \frac{f_{NE}(\mathcal{D}, Weed\_Leaf\_Counting)}{f_{NE}(\mathcal{D}, ImageNet-1k)} \quad (3)$$

Third, we ran our DSD algorithm over all dataset pairs with a $unit\_size$ of 2000 and a $image\_size$ of 28×28. Since we used epoch ratio as a measurement for transfer learning performance, we proposed to use the ratio of DSD of a dataset to the Weed Leaf Counting and the ImageNet-1k datasets as a measurement for domain relevance, as shown in Equation (4).

$$f_{DSDRatio}(\mathcal{D}) = \frac{f_{DSD}(\mathcal{D}, Weed\_Leaf\_Counting)}{f_{DSD}(\mathcal{D}, ImageNet-1k)} \quad (4)$$

Last but not least, as shown in Figure 4, we normalized epoch and dataset distance ratios to examine the correlation between model training performance and dataset distance. As a contrast for the DSD ranking algorithm, we also obtained results from the modified OTDD algorithm.

Across all three networks, results showed a stronger correlation between the epoch ratio and DSD ratio compared to the modified OTDD. The DSD ratio and the epoch ratio follow a same trend that if DSD ratio is relatively small, then the epoch ratio tends to have a similar feature. We also calculated the Pearson correlation coefficient to measure the correlation between DSD with epoch ratio. The Pearson correlation coefficient of normalized epoch ratio with normalized DSD for ResNet50, EfficientNetV2S, and MobileNetV3L are respectively 0.93, 0.85, and 0.90, showcasing a strong positive correlation. While the Pearson correlation coefficient of that with the modified OTDD are 0.23, 0.0, and 0.18.

E. MTaaS with DSD Algorithm

In this section, we studied the use case for deploying the DSD algorithm on a cloud-based model training service. We designed experiments to address cloud service memory limitations and accelerate the training process by leveraging the DSD algorithm.

When job queues are submitted to the cloud service, DSD algorithm is initialized on new datasets in the job queue. The algorithm computes normalized similarity scores between these new datasets and those existing datasets stored in the system. The scores aim to help find the best pre-trained weight for accelerating model training process on new datasets. However, with limited memory size, we face the problem of fitting a subset of all pre-trained weights in memory. To reduce the total training time of the job queue, we also need to consider other factors. One factor is dataset size. Speedup on larger datasets may contribute more to time reduction. Another factor is dataset frequency.

These three factors (normalized similarity scores, dataset size and frequency) in the job queue together are considered in a training time estimation (TTE) algorithm for selecting a subset of pre-trained weights that potentially has largest speedup on reducing the total training time, as shown in Equation (5). Based on DSD, the algorithm gives an estimated training time for a new dataset using transfer learning on existing pre-training weight. D_e represents an existing dataset in the system, while D_n represents the new dataset. A higher DSD score means a closer relation between two datasets, and hence means a faster training time. Therefore, we use inverse dataset similarity score to estimate model training time. The results from the estimation algorithm doesn't represent the actual training time. They are merely for comparisons to find the pre-trained weight that has the highest speedup over model training.

$$f_{TTE}(D_e, D_n) = (1 - f_{\text{NormDSD}}(D_e, D_n)) \cdot \text{Size}(D_n) \cdot \text{Frequency}(D_n) \quad (5)$$

As a baseline, we compare our approach against using ImageNet pre-trained weights for all job queues. While ImageNet weights are widely adopted, our method tailors the selection process based on specific job queue characteristics.

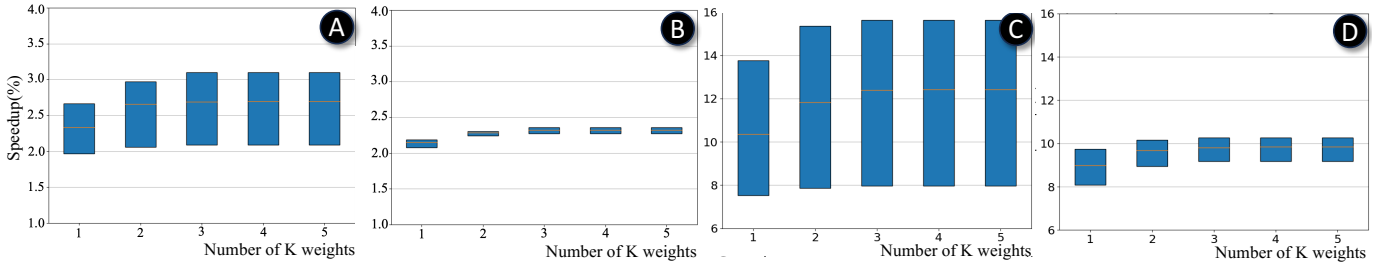


Fig. 5. Box plot on speedup performance for ResNet50 from normal and bursty job queues given different K. K represents the number of pre-trained weights to store in memory.(a) and (b) are speedup performance for bursty and normal job queues over ImageNet pre-trained weight respectively, while (c) and (d) are speedup performance for bursty and normal job queues over random weight respectively.

To simulate real-world scenarios, we consider two types of job queues: Bursty job queue that consisting of multiple duplicate jobs with the same datasets, typically used for hyperparameter optimization. Normal job queue containing diverse jobs with fewer duplicates.

We collected 7 new real-world agricultural datasets for simulating job queues. We randomly selected 5 datasets from all 7 for each job queue. And randomly generated 8 job queues for each type. As shown in Figure 5(a) and (b), the average model training speedup was measured comparing our approach to the benchmark. From the results, the average speedup for normal and bursty job queue are 2.3 and 2.8 respectively. The speedup performance for normal job queues of our algorithm provides a stable result over the ImageNet benchmark. The first quartile and the third quartile are very close to each other given different K values. The most significant speedup happens at K = 1. With only 1 pre-trained weight stored in the memory, it can boost the mean training time for all normal job queues at around 2.2 times. With K growing larger, the mean speedup grows from 2.2 to 2.3, 2.36 times. That is to say, with the available memory getting larger to contain all pre-trained weights, the advantage of our algorithm is gradually converging to 2.36 times. On the other hand, the performance of our algorithm on bursty job queues are very dynamic. The first quartile and the third quartile varies a lot for all K values. This is due to the random duplicate jobs in the job queues. However, with this randomness and job duplicates, the algorithm also provide a higher mean speedup results over the normal job queues. It started with a mean value at around 2.2 times, and eventually converging to a higher 2.7 times. To make the experiments more comprehensive, we also report speedup comparison of our method to model training processes with random weights. As shown in Figure 5(c) and (d), the performance speedup is over 12 times faster for bursty job queues and over 8 times faster for normal job queues. Results showed that our algorithm gained great speedup performance over ImageNet benchmark for both normal and bursty job queues.

F. DSD and Network Architecture Characterization

We also ran experiments to validate our method to deploy the DSD algorithm on network architecture selection. For each dataset, models from three network architectures were trained and used for test set prediction. Then, three predicted test sets were created based on three network models for computing the DSD with the original test set. In Figure 6, we showed results of correlations between the performance of a network model and the DSD of the predicted test set created from the model with the original test set. We also deployed three-fold cross validation technique to make our results reliable. For most datasets, the results hold true that if a network model achieves higher accuracy on the test set of a dataset, then the DSD between its predicted and the original test set will be closer. In other word, the DSD between the predicted and the original test set of a dataset can be used as a measurement for the performance of a network model.

G. DSD Robustness Evaluation

We ran our experiments using the DSD algorithm with a default configuration, a $unit\_size$ of 2000 and an $image\_size$ of 28 $\times$ 28. To test the robustness of our DSD algorithm, we ran different parameter configurations for the domain relevance dataset pool table. The domain relevance dataset pool consists of a 10 $\times$ 10 row-wise normalized matrix. We selected $image\_size$ of 14$\times$14, 28$\times$28 (default), 56$\times$56 and $unit\_size$ of 1000, 1500, 2000 (default), 2500, 3000 and computed the element-wise percentage difference separately. As shown in Figure 7(a), we obtained 5 domain relevance dataset table, each represents one $unit\_size$ configuration, and did a cross comparison between every two table at an element level. The percentage difference of each element is computed and record. In total, data from 1000 pairs of elements were collected. Among them, a majority of 936 counts falls into the percentage difference range of 0 to 5%. The largest range of percentage difference covers from 15% to 20% with just 6 counts. A similar results were collected for testing different $image\_size$ configurations. As shown in Figure 7(b), with 3 $image\_size$ configurations, 300 pairs of elements were collected. 217 out of 300 counts falls into the smallest percentage difference range of 0 to 5%. Only 3 counts fall into the largest range from 20% to 25%. Through the conducted experiments, we established that the DSD algorithm is able to run accurately with different parameter configurations.

Experiments were conducted on demonstrating the running speed of the DSD algorithm. We also ran various configurations of $image\_size$ and $unit\_size$. Results were collected

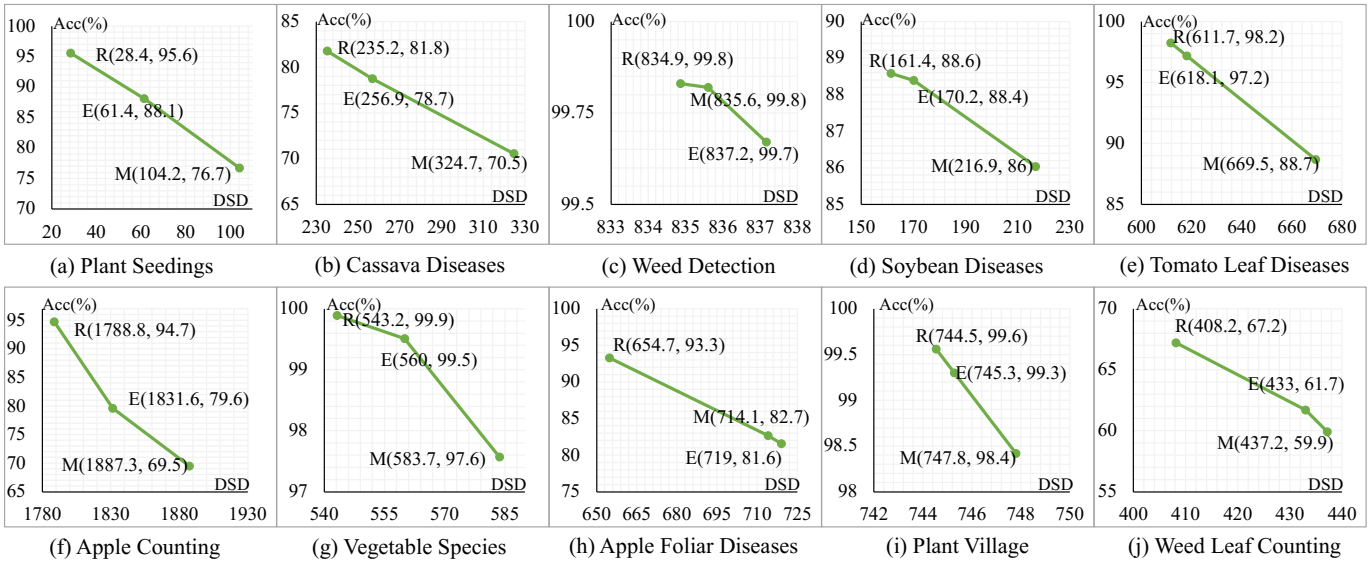


Fig. 6. Relationship between DSD of predicted test sets with the original test set and network model performance on different datasets. R represents ReNet50, E for EfficientNetV2S, and M for MobilNetV3L.

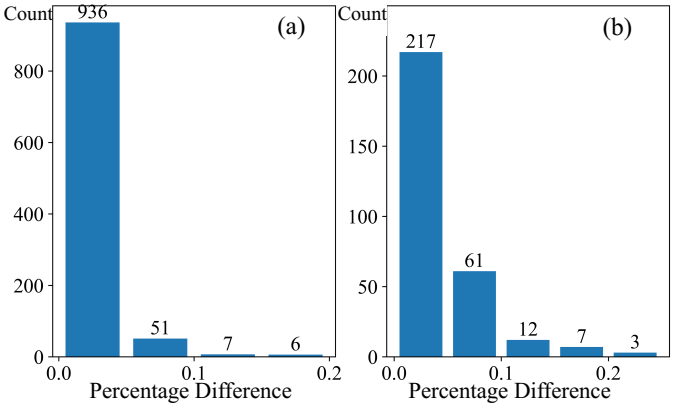


Fig. 7. Element-wise percentage difference of domain relevance dataset pool for various configurations. (a) shows data on different $unit\_size$ of 1000, 1500, 2000, 2500, 3000. (b) shows data on different $image\_size$ of 14×14, 28×28, 56×56. X-axis represents ranges of percentage difference while y-axis represents the count of elements in the range.

from a CPU-only server with two Intel Xeon E5-2620 v4 CPU. Each CPU has 8 cores, one thread per core. As shown in Table II, the speed of the default configuration of the DSD algorithm has an average running time of 235 seconds for any two datasets in our system. Even we increased the $image\_size$ to 56×56, the cost is still less than 5 minutes per computation.

H. Limitations and Future Work

We believe that larger searching pools with more datasets and network architectures have the potential to make our system more accurate. In our work, we only explore the potential of the DSD ranking algorithm on domain-specific agricultural datasets. We think that our work showed the potential of domain-aware model training tasks beyond the field

TABLE II
TABLE A AND B SHOW THE AVERAGE RUNNING TIME OF VARIOUS IMAGE SIZE AND UNIT SIZE, RESPECTIVELY.

Table a. The default unit size is 2000.

image size	14×14	28×28	56×56
average time (s)	192.08	235.23	269.30

Table b. The default image size is 28×28.

unit size	1000	1500	2000	2500	3000
average time (s)	194.69	205.08	235.23	237.26	267.29

of precision agriculture for MTaaS. Datasets from different domains can be used for future study.

V. Related Work

The notion of MTaaS has gradually become a hot topic for both large companies and research institutions with the huge demand for AI. Parallel computing clusters equiped with GPUs are emerging for ML related tasks, such as Amazon AWS, Google Colab, Ohio Supercomputer center, and Alibaba PAI. However, running diverse model training tasks in heterogeneous GPU clusters posts many challenges for high utilization and fast job completion. To speedup model training process, Weng *et al.* proposed a simple shortest-job-first scheduling strategy for solving long queuing delays for short-running task instances [36]. Some research also focused on training models to predict the task runtime and then prioritize them for high utilization. Mahajan *et al.* proposed a scheduling framework for model training workloads that uses auction and bid mode between GPU devices and model training tasks based on available computing resources [20]. Narayanan *et al.* built a heterogeneity-aware scheduler that systematically generalizes a wide range of existing scheduling policies [24]. Peng *et al.* used online fitting to predict model convergence

during training, and set up performance models to accurately estimate training speed as a function of allocated resources in each job [27]. These previous works for job scheduling in parallel computing clusters required either heavy model training for predicting job runtime or customized framework support with complex scheduling algorithms. Our approach stand alone from any framework support and model training. We focus on domain-awareness for model training jobs within a domain, and only use dataset similarity that can be computed fast on CPUs for fast training completion.

VI. Conclusion

In this paper, we explored the effectiveness of model training processes from a domain-specific perspective. A DSD ranking algorithm is proposed to characterize the efficacy of model training tasks using pre-trained weights from datasets, which can be easily computed using CPUs. Furthermore, we designed a cloud-based system for MTaaS that utilized limited cache space and hugely speedup model training jobs for a given dataset without any model training processes needed. Our results demonstrated that within the agricultural domain, model training jobs with domain-specific pre-trained weights for MTaaS could be greatly accelerated up to 12.2 times compared to that with the ImageNet pre-trained weight.

Acknowledgements: This work was funded by ACCESS Computing Startup Grant CIS220074, NSF Grants OAC-2112606, and the Ohio Soybean Council. In addition, we would like to thank Rugved Katole, Rajbabu Velmurugan, Maryam Shojaei Baghini, DK Panda, Raghu Machiraju, Seth Ockerman, Hari Subramoni, and Arpita Sinha for early feedback.

References

[1] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. {TensorFlow}: a system for {Large-Scale} machine learning. In *12th USENIX symposium on operating systems design and implementation (OSDI 16)*, pages 265–283, 2016. 4

[2] M Israk Ahmed, Shahriyar Mahmud Mamun, and Asif Uz Zaman Asif. Dcnn-based vegetable image classification using transfer learning: A comparative study. In *2021 5th International Conference on Computer, Communication and Signal Processing (ICCCSP)*, pages 235–243. IEEE, 2021. 5

[3] David Alvarez-Melis and Nicolo Fusi. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems*, 33:21428–21439, 2020. 3

[4] Xiuhong Chen, Xianglei Huang, Chaoyi Jiao, Mark G Flanner, Todd Raeker, and Brock Palen. Running climate model on a commercial cloud computing environment: A case study using community earth system model (cesm) on amazon aws. *Computers & Geosciences*, 98:21–25, 2017. 1

[5] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009. 1

[6] Alessandro dos Santos Ferreira, Daniel Matte Freitas, Gercina Gonçalves da Silva, Hemerson Pistori, and Marcelo Theophilo Folhes. Weed detection in soybean crops using convnets. *Computers and Electronics in Agriculture*, 143:314–324, 2017. 5

[7] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1

[8] G Geetharamani and Arun Pandian. Identification of plant leaf diseases using a nine-layer deep convolutional neural network. *Computers & Electrical Engineering*, 76:323–338, 2019. 5

[9] Thomas Mosgaard Giselsson, Mads Dyrmann, Rasmus Nyholm Jørgensen, Peter Kryger Jensen, and Henrik Skov Midtiby. A Public Image Database for Benchmark of Plant Seedling Classification Algorithms. *arXiv preprint*, 2017. 5

[10] Ubaid Ullah Hafeez and Anshul Gandhi. Empirical analysis and modeling of compute times of cnn operations on aws cloud. In *2020 IEEE International Symposium on Workload Characterization (IISWC)*, pages 181–192. IEEE, 2020. 1

[11] Mohamed Hanafy and Ruixing Ming. Machine learning approaches for auto insurance big data. *Risks*, 9(2):42, 2021. 1

[12] Nicolai Häni, Pravakar Roy, and Volkan Isler. Minneapple: a benchmark dataset for apple detection and segmentation. *IEEE Robotics and Automation Letters*, 5(2):852–858, 2020. 5

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4

[14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Identity mappings in deep residual networks. In *Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part IV 14*, pages 630–645. Springer, 2016. 4

[15] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1314–1324, 2019. 4

[16] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 4

[17] Minyoung Huh, Pulkit Agrawal, and Alexei A Efros. What makes imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*, 2016. 1

[18] Igor Kononenko. Machine learning for medical diagnosis: history, state of the art and perspective. *Artificial Intelligence in medicine*, 23(1):89–109, 2001. 1

[19] Konstantinos G Liakos, Patrizia Busato, Dimitrios Moshou, Simon Pearson, and Dionysis Bochtis. Machine learning in agriculture: A review. *Sensors*, 18(8):2674, 2018. 1

[20] Kshiteej Mahajan, Arjun Balasubramanian, Arjun Singhvi, Shivaram Venkataraman, Aditya Akella, Amar Phanishayee, and Shuchi Chawla. Themis: Fair and efficient {GPU} cluster scheduling. In *17th USENIX Symposium on Networked Systems Design and Implementation (NSDI 20)*, pages 289–304, 2020. 8

[21] Oksana Markova, Serhii Semerikov, Andrii Striuk, Hanna Shalatska, Pavlo Nechypurenko, and Vitalii Tron. Implementation of cloud service models in training of future information technology specialists. 2019. 1

[22] Ashish Motwani and Qasim Khan. Tomato leaves dataset, 2022. 5

[23] Ernest Mwebaze, Jesse Mostipak, Julia Elliott, and Sohier Dane. Cassava leaf disease classification, 2020. 5

[24] Deepak Narayanan, Keshav Santhanam, Fiodar Kazhamiaka, Amar Phanishayee, and Matei Zaharia. {Heterogeneity-Aware} cluster scheduling policies for deep learning workloads. In *14th USENIX Symposium on Operating Systems Design and Implementation (OSDI 20)*, pages 481–498, 2020. 8

[25] Seth Ockerman, John Wu, Christopher Stewart, and Zitchen Zhang. A reflection on ai model selection for digital agriculture image datasets. In *2nd AAAI Workshop on AI for Agriculture and Food Systems*, 2023. 2

[26] Jigar Patel, Sahil Shah, Priyank Thakkar, and Ketan Kotecha. Predicting stock and stock price index movement using trend deterministic data preparation and machine learning techniques. *Expert systems with applications*, 42(1):259–268, 2015. 1

[27] Yanghua Peng, Yixin Bao, Yangrui Chen, Chuan Wu, and Chuanxiong Guo. Optimus: an efficient dynamic resource scheduler for deep learning clusters. In *Proceedings of the Thirteenth EuroSys Conference*, pages 1–14, 2018. 9

[28] Valerio Perrone, Rodolphe Jenatton, Matthias W Seeger, and Cédric Archambeau. Scalable hyperparameter transfer learning. *Advances in neural information processing systems*, 31, 2018. 1

[29] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large

Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. 4, 5

[30] David Salinas, Huibin Shen, and Valerio Perrone. A quantile-based approach for hyperparameter transfer learning. In *International conference on machine learning*, pages 8438–8448. PMLR, 2020. 1

[31] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1, 2

[32] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 2, 4

[33] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 4

[34] Nima Teimouri, Mads Dyrmann, Per Rydahl Nielsen, Solvejg Kopp Mathiassen, Gayle J Somerville, and Rasmus Nyholm Jørgensen. Weed growth stage estimator using deep convolutional neural networks. *Sensors*, 18(5):1580, 2018. 5

[35] Ranjita Thapa, Kai Zhang, Noah Snavely, Serge Belongie, and Awais Khan. The plant pathology challenge 2020 data set to classify foliar disease of apples. *Applications in plant sciences*, 8(9):e11390, 2020. 5

[36] Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. {MLaaS} in the wild: Workload analysis and scheduling in {Large-Scale} heterogeneous {GPU} clusters. In *19th USENIX Symposium on Networked Systems Design and Implementation (NSDI 22)*, pages 945–960, 2022. 8

[37] Dani Yogatama and Gideon Mann. Efficient transfer learning method for automatic hyperparameter tuning. In *Artificial intelligence and statistics*, pages 1077–1085. PMLR, 2014. 1

[38] Zichen Zhang, Jayson Boubin, Christopher Stewart, and Sami Khanal. Whole-field reinforcement learning: A fully autonomous aerial scouting method for precision agriculture. *Sensors*, 20(22):6585, 2020. 1

[39] Zichen Zhang, Sami Khanal, Amy Raudenbush, Kelley Tilmon, and Christopher Stewart. Assessing the efficacy of machine learning techniques to characterize soybean defoliation from unmanned aerial vehicles. *Computers and Electronics in Agriculture*, 193:106682, 2022. 1, 5