IMAGE BASED PLANT DISEASE IDENTIFICATION

Ryan Raymond Philip
Student# 1007731689
ryan.philip@mail.utoronto.ca

Veer Kunal Kapadia
Student# 1008817773
veer.kapadia@mail.utoronto.ca

Muhammad Abdullah Choudhary
Student# 1008767791
muhammad.choudhary@mail.utoronto.ca

Md Iftikher Zaman Chowdhury
Student# 1008890808
iftikherzaman.chowdhury@mail.utoronto.ca

ABSTRACT

Plant diseases have had devastating effects on agricultural and food security. This project aims to develop a CNN-based plant disease classification model to help farmers in detecting and identifying diseases in their crops quickly and accurately. Using the pre-compiled 'New Plant Diseases Dataset' of 38 different plant species, including a varying range of diseases affecting them, we are training a Convolutional Neural Network (CNN) to complete this botanist task. The work done by each member is documented, including a breakdown of the teams methods of collaboration, distribution of tasks and the future plans for the successful completion of the project. By processing and arranging the Plant Village Data set, as well as using a variety of data augmentations, we aim to force the model to recognize key underlying features rather than simply memorizing the data, and thus being capable of accurate multi-class classification and disease diagnosis. Furthermore, to obtain a realistic testing scenario, our team personally collected images from local farms as well as online image sets that were not present in our training and validation data sets. These images will be used without augmentation for the testing phase to emulate how a farmer would use the model. To get a benchmark for our primary model, a Support Vector Machines (SVM) model was be used as a baseline. SVMs are simple models suitable for multi-class classification tasks, but they lack the ability to capture complex patterns and details that neural networks do. The model was trained in two different ways, each with a different number of classes, to get a deeper understanding of the problem. The results of the baseline model gave us further insight into how the number of classes affect the model for our particular data set and a benchmark accuracy, which was considered when developing the primary model. The primary CNN model was designed with 6 convolutional layers and 3 linear layers. The model makes use of pre-existing techniques such as Layer Normalization, Dropout, ADAM, Weight Decay and the ReLU activation function in order to maximize validation accuracy and reduce overfitting to the test data. It also utilizes skip connections, similar to the ones in ResNet, in order to avoid vanishing and exploding gradients. This model was then trained using a variety of hyperparameters, such as different batch sizes, learning rate, and number of epochs to obtain the best validation accuracy. We also considered the effect of the data augmentations we did in the data processing. By training the model on the datasets with different probabilities of augmentation, we were able to see the effect it produced on our models accuracy. Several challenges were faced when training the model, including limitations of the software and hardware used for training as well as other bugs within the code, but they were able to be resolved by the team using the resources available to us.

—-Total Pages: 9

1 Brief Project Description

Plant disease poses a significant risk to food production, resulting in annual losses of approximately 200 billion US Dollars (Figure 1). Our project on image classification for plant disease detection using Convolutional Neural Networks (CNNs) is motivated by the need to provide farmers with an effective tool to detect and classify plant diseases at an early stage. The primary goal of this project is to empower farmers by enabling them to identify diseases in plants as early as possible, and thus allowing for an early prevention of disease spread to other plants. By achieving this goal, the project aims to contribute to increased agricultural productivity and a higher yield of crops.

The objective of this project is to create a model that can analyze images of plant leaves and determine whether the plant is healthy or diseased. Moreover, the model aims to classify the specific disease and plant species, outputting to the user the disease and species type. By utilizing deep learning algorithms, our model can be trained to identify subtle visual patterns associated with diseased plant leaves. We want to make the disease identification process extremely simple, being possible with a simple click (Figure 2). This early and accurate detection enables timely disease control measures, leading to improved productivity and potential prevention of disease spread.

Figure 1: Diseased corn plant

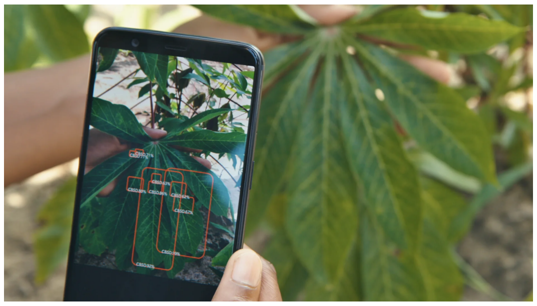
Figure 2: Plant disease identification through camera on an app

Deep learning is particularly a reasonable approach for this project due to its inherent ability to analyze and interpret complex visual data, such as images.With the recent advent in Machine Learning technology and the boom of Neural Networks (Figure 3), deep learning models, like CNNs, have proven to be highly effective in image classification tasks, including medical imaging and object recognition. The hierarchical structure of these models allows them to learn intricate patterns and extract features from images, enabling accurate disease classification. By leveraging the large datasets available of plant images, CNNs can be trained to almost perfectly recognize specific disease patterns, leading to the most reliable and efficient disease identification. This approach has the potential to significantly reduce manual labor, time, and costs involved in traditional disease diagnosis methods, making it the most reasonable and promising solution for assisting farmers in disease management and prevention.

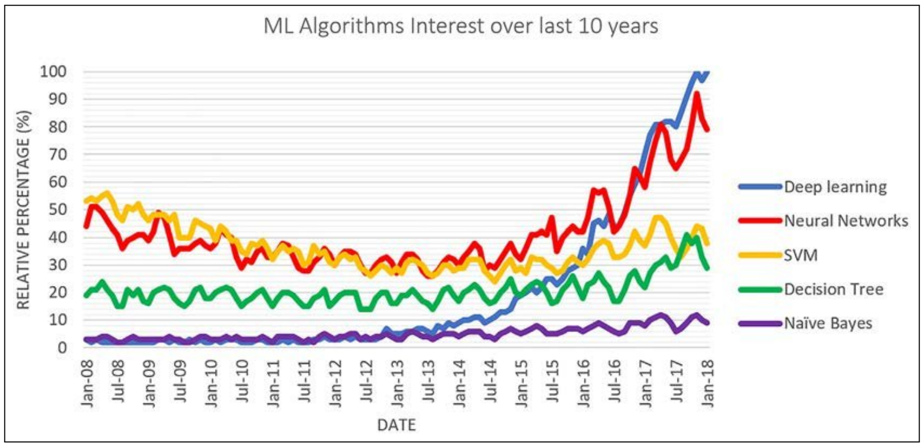

Figure 3: Interest in and use of Deep Learning's performance over the last 10 years

To achieve our goals, we will utilize the pre-compiled 'New Plant Diseases Dataset', which includes images of 38 different plant species affected by various diseases. By training a CNN on this dataset, we aim to equip the model with the ability to perform this botanical task. Each leaf image contains vital information about the plant species, presence of diseases, and specific disease types. By appropriately augmenting, organizing, and processing this data, we aim to train the model to recognize crucial underlying features rather than simply memorizing the data. This approach will enable accurate multi-class classification and disease diagnosis.

2 Individual Contributions and Responsibilities

The team is collaborating remotely by having multiple meetings weekly via Zoom. During meetings, the members discuss their tasks and progress to ensure everyone is on track and ask for help if they encounter any problems. The team is using GitHub and Google Colab notebooks to work on their assigned sections of the project, while also being able to have version control and allow other members to observe progress. Tasks such as model training are time intensive and are susceptible to errors such as the software crashing/timing out. To account for this, multiple members will train versions of the network individually. Additionally, for tasks with a large scope, such as hyperparameter tuning, all members will try different values, and the results will be compared after each iteration to see what model parameters that give the highest validation accuracy.

Table 1: Assigned Tasks and Deadlines

Task	Description	Assigned Person	Internal Deadline	Date Completed
Data Collection	Collected data (38 images of diseased and healthy plant leaves) from in person visits to local farms as well as online image sets. Arranged those images in accordance with the labels existing in the source data sets. Resized all images into an uniform dimension.	**Iftikher**	08/07/2023	08/07/2023

Data Loading and Augmentation	Downloaded the dataset zip file, uploaded it to colab, and extracted the contents. Loaded the training, validation and testing folders, resized all the images and wrote functions to perform the transformations which included adding gray blocks, adding gaussian noise, and applying the random crop and center crop filters to the images.	**Veer**	08/07/2023	07/07/2023
Baseline Model	Implemented a baseline model using SVM. Split the data into two numpy arrays, for labels and images, converted to tensors and then trained models.	**Ryan:** Worked on data preprocessing (splitting, resizing) and the training code. **Muhammad:** Debugged the training code and worked on the testing code.	09/07/2023	08/07/2023
Primary Model	Developed a CNN network with several convolution layers and fully connected layers. Utilized skip connections to prevent vanishing gradient problem + weight decay and dropout to prevent overfitting + layer normalization and Adam Optimizer.	**Ryan:** Wrote functions to get accuracy, evaluate model and helped debug skip connections code. **Muhammad:** Initialized model, implemented skip connections, weight decay and dropout.	12/07/2023	12/07/2023
Model Training and Evaluation	Used augmented data to train the network, while displaying training + validation loss and accuracy. Initially, the model was overfitted with a small data set to verify the model was learning and after confirmation, the full data was used.	**Muhammad:** Wrote the code for the training function. **Ryan:** Debugged the training function, and wrote code for the plotting function. **Veer:** Overfitted the model with the small dataset and ran the model with the full dataset.	25/07/2023	Ongoing
Hyperparameter Tuning and Testing	Various hyperparameters are being tuned, such as learning rate, batch size, model architecture and epochs.	**Muhammad:** Tested the model with augmentation probability set to 0.4 and reported accuracy. Will now adjust the number of layers. **Veer:** Tested the model with batch sizes 128/256 and 25/50 epochs and reported accuracies. Will now adjust the number of layers.	25/07/2023	Ongoing

Final Project Presentation	Dividing the content to prepare for the presentation among team members. Assigning tasks of video editing and proofreading speeches among team members.	**Iftikher**: Problem and Data sections and proofreading; **Muhammad:** Model and Demonstration sections; **Ryan:** Quantitative and Qualitative results sections; **Veer:** Data Processing and Takeaways sections and Video Editing.	01/08/2023	Ongoing

3 NOTABLE CONTRIBUTIONS

3.1 DATA PROCESSING:

The dataset that we will be using for this project is the 'New Plant Diseases Dataset', which is an augmented version of the 'PlantVillage' dataset, and our team has further augmented the former in the hope of obtaining better performance from our model. The 'New Plant Diseases Dataset' consists of 87,900 RGB images of the leaves of diseased and healthy plants, categorized into 38 different classes and split into 87,867 images for training and validation and 33 images for testing. The training and validation images have been split in an 80:20 ratio (70,295 for training and 17,572 for validation) by the team that originally used this dataset. The original directory structure of the PlantVillage Dataset has been preserved. Each of the 38 directories consist of about 1800 training and 500 validation images.

Data Cleaning performed by the 'New Plant Diseases Dataset' team: The team studied a few images from the 'New Plant Diseases Dataset' and observed that certain filters such as 'rotation' and other brightness augmentations have already been performed on the dataset by the team working on it (Figures 4 and 5).

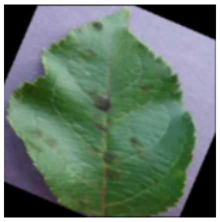
Figure 4: Rotated Image of a leaf

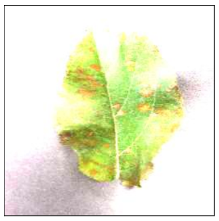
Figure 5: Brightened Image of a leaf

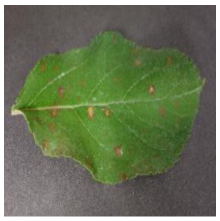
Figure 6: Leaf Alignment Sample 1

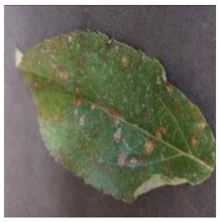
Figure 7: Leaf Alignment Sample 2

The team observed that approximately 23 percent of the training images had been rotated and about 10 percent had increased levels of brightness. It is worth noting that, as far as our team could tell, all the images that had been rotated, had been rotated by the same degree or angle. Moreover, the team also observed that even the images that had not been rotated had been taken with the leaves aligned in different directions (Figures 6 and 7).

Data Cleaning Performed by our team: In terms of the data cleaning and augmentations performed by our team for our project, we focused on the following steps:

- Resizing the Images: We first used the 'transforms.Resize' function to ensure that all the images were of the same dimensions (256x256).
- Next, as mentioned earlier, our team decided to perform more advanced augmentations on the dataset that had not been performed by the previous team so that our model per-

forms better at classifying the plant diseases. We wrote functions to perform the following augmentations on the data:

- **Adding a gray block to the image**: A square of size 50x50 pixels would be added to the image to block out a part of the leaf (Figure 8), thereby forcing the model to learn from a smaller section of the leaf.
- **Adding Gaussian noise**: Gaussian noise with the standard deviation set as 0.9 would be added to the image (Figure 9).
- **Center Crop:** A uniform portion of the border of the image is cropped and only the central section of the image is visible (Figure 10). The 'transforms.CenterCrop' function is used and the 256x256 sized image is reduced to a 180x180 sized image.
- **Random Crop:** A non-uniform portion of the border is randomly cropped and the image is reduced from 256x256 pixels to 180x180 pixels (Figure 11). The 'transforms.RandomCrop' function is used to perform this augmentation.

Each of the 4 augmentations have a 5 percent probability of being applied to a given image, and only a single augmentation can be applied to said image. This implies that 20 percent of our training data has some form of transformation applied to it.

The team was challenged to find augmentations that would be appropriate for our deep learning model, given that a lot of available augmentations are redundant and not useful in making our model perform better in the real world.

Data Collection: In addition to the 33 testing images that were part of the dataset, the team collected 38 images of diseased and healthy plant leaves from in person visits to local farms as well as online image sets. Each image belongs to a different class of our dataset and we labeled the images accordingly, following which we resized them to uniform dimensions. During testing, we will be passing this data directly to the model without performing any augmentations on it.

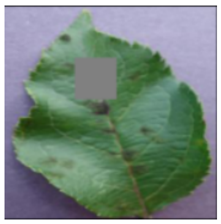
Figure 8: Leaf image with a gray square

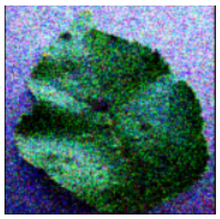
Figure 9: Leaf image with Gaussian noise

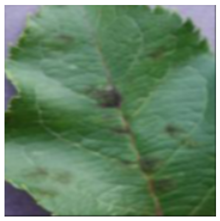
Figure 10: Center crop transformation

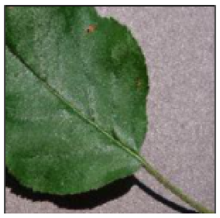
Figure 11: Random crop transformation

3.2 Baseline Model

The baseline model employed for the plant disease detection task is the Support Vector Machines (SVM) algorithm. The goal of the project is to build an accurate and efficient model for automatically detecting plant diseases based on input images. The decision to use SVMs instead of other models, like Logistic Regression, was driven by SVMs' being able to fulfill this goal through their capability of multi-class classification. Unlike Logistic Regression, which can only handle binary classification, SVMs can effectively handle multiple classes.

SVMs are simple models that do not possess the same pattern recognition abilities as neural networks do. However, in our project, we can employ SVMs for multi-class classification using the one-against-all approach. These approaches separate classes by identifying an optimal hyperplane that effectively distinguishes between them. Considering this, SVMs are easy to implement and require minimal tuning.

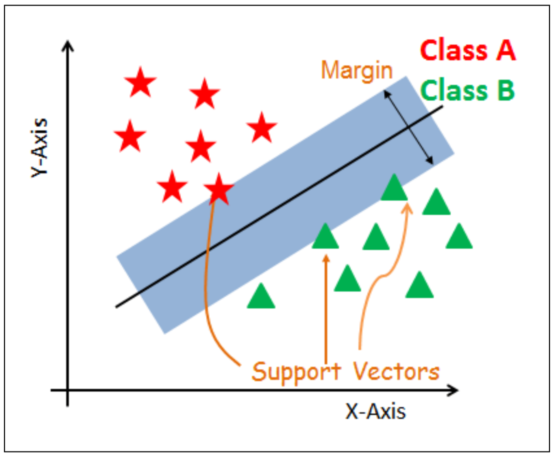

Figure 12: SVM model sample hyperplane for 2 classes

Figure 12 shows an example of how a plane will provide a decision boundary that divides the data for 2 classes.

In our case, the SVM model was trained on the same dataset of plant disease images with 38 classes. It was then compared against our primary CNN model to assess its feasibility in achieving our objective. The SVM model consisted of the following steps:

- Data Pre-processing: The input images are resized, normalized, and split into label and image Numpy arrays.
- Training: The SVM model is trained using the extracted features and corresponding disease labels from the training set and tested on the validation set.
- Testing and Prediction: The trained SVM model is used to classify new, unseen images by extracting features from them and predicting the disease label based on the learned decision boundaries.

Although setting up and training it is a simple process, during the development of the baseline SVM model, the team encountered several challenges that needed to be addressed. Given the large number of images (approximately 90,000), we decided to utilize the Kaggle API to store the images on the Google Colab disk space. Once the images were stored, the next challenge was to train the SVM model, which required passing two separate numpy arrays—one containing the images and the other containing the corresponding labels. We could extract the labels by using the train_test_split() function, but the data was already split into separate train, validation, and test folders. However, attempts to extract labels using our own loop resulted in run-time crashes due to the significant amount of images and Google Colab's free version having limited available RAM.

To address this issue, we leveraged our own computer's GPU in conjunction with the Google Colab RAM, thereby distributing the memory load, while also cloning and detaching each image during each iteration of the for loop to minimize RAM usage. Once the label extraction hurdle was overcome, we encountered a further obstacle when attempting to train the SVM model. Again, the issue of insufficient available RAM emerged, compounded by the limited runtime provided to free users of Google Colab. To mitigate this challenge, we experimented with adjusting certain parameters. By increasing the batch size for training and reducing the values of C, gamma, and kernel for the SVM model to search through, we were able to get the model train within the available RAM.

The baseline SVM model achieved a validation accuracy of 43%. This result serves as a reference point for evaluating the performance of subsequent models. Although the accuracy is relatively low, it provides insights into the potential challenges faced in the task of plant disease identification. This low accuracy can be explained by the amount of classes present. Due to there being an overwhelming variety of classes, it is tougher for the model to generate decision boundaries capable enough of differentiating between all 38 classes, making this 43% accuracy relatively well for a non-neural-network classifier. This is because a random chance model will predict 2.6% for each class so our model performs significantly better.

Upon further inspection, it was noted that the SVM model had difficulty distinguishing between visually similar disease patterns or subtle variations in plant health conditions.

This was seen when we trained the model on a limited number of classes that were visually different, the resulting validation accuracy being 90% which is much higher. This suggests that, unlike neural networks, capturing fine-grained details and complex patterns is not something SVMs are adept at doing. There is also the implication that a larger number of classes, the complexity of the classification problem also increases, leading to a lower overall accuracy, which must be taken into account when designing our primary model. Since we aim to utilize all 38 classes, we will use the SVM model accuracy of 43% as a benchmark to compare our primary model with.

3.3 Primary Model

For the task of plant disease classification, we implemented a CNN with 6 convolutional layers and 3 linear layers. The architecture includes skip connections to prevent gradient loss, ADAM as the optimizer, and ReLU activation function for non-linearity. To avoid overfitting, we applied dropout with a probability of 0.3 on the linear layers. Additionally, we used L2 weight decay to prevent the model from memorizing the image data (Figure 13). This architecture was inspired by ResNET due to its monumental performance on the ImageNet dataset. Our dataset is also extremely large (80000 images), and we have 38 classes, making this architecture the right starting point due to its advanced capability of recognizing underlying features and patterns in order to classify many classes accurately.

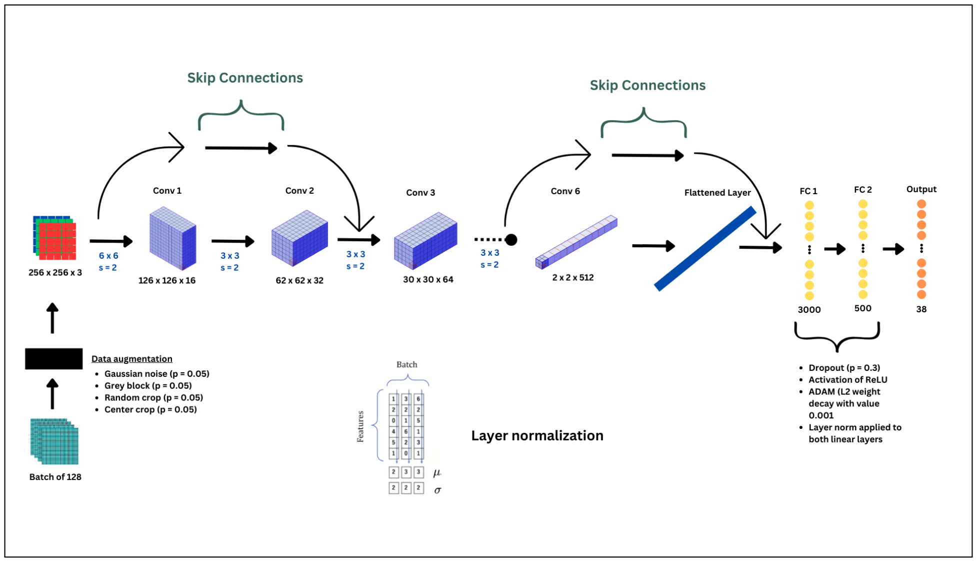

Figure 13: Visual representation of model architecture

During the model development process, our team experimented with different hyperparameter values to find the best configuration. Each team member ran various models with different hyper parameters on their computers in order to save time and compare the results of each model (one model takes about 5 hours to be trained) in order to find out the best hyper parameters. Initially, we used a batch size of 64, trained the model for 30 epochs, set the learning rate to 0.01, and weight decay value to 0.001. We compared the results of various models. For example, one of the initial models that we ran comprised of the same hyper parameters as the best performing model, except the probability for an augmentation to be applied to the images was changed to 0.4. This model's highest validation accuracy was 0.9363 at epoch 17. From this, we understood that changing the probability for augmentations to 0.4 did not improve the model's performance. After comparing the results of various models, we found that the best performing model had the following hyperparameter: a probability of 0.2 for image augmentations to be applied, a batch size of 128, 25 epochs, a learning rate of 0.001, and a weight decay value of 0.001.

This best model achieved a peak validation accuracy of 0.9385 and a peak training accuracy of 0.9353 on epoch 23 (Figure 14 & 15). By plotting the validation and training accuracies, we observed that the model performed well without significant overfitting or underfitting, reaching a plateau in accuracy improvement. In terms of specific classes, the model performed extremely well throughout all classes. Due to run time errors and long training times, we were unable to identify specific classes on which the model performed even better, or the ones that the model mis-classified. Our next goal is to identify these outlier classes/ images.

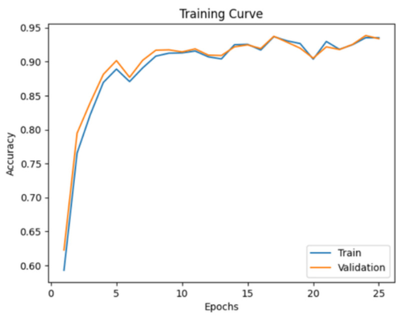

Figure 14: Training curve for loss and validation accuracies

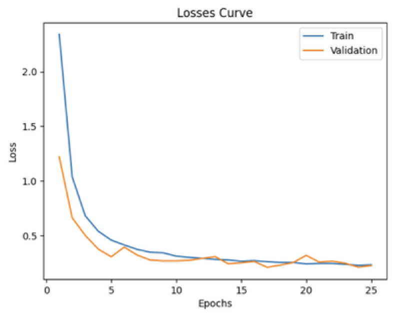

Figure 15: Training curve for loss and validation losses

In terms of challenges, we encountered several difficulties during the model development process. Firstly, we had to refer to multiple documentations and Stack Overflow pages to implement certain hyperparameters like L2 weight decay, who's implementation was not covered in lectures. Debugging the code was also a challenge, especially when using CUDA for GPU. We spent several hours trying to resolve the issue of input images not being the correct format in the criterion. We though the problem was within the train function, only to discover that it was caused by us forgetting to use an if statement for CUDA usage and conversion of images and labels to the GPU version within a helper function that evaluates the model's performance.

Furthermore, we faced some setbacks due to simple mistakes. For example, when running two models, one with data augmentation and one without, we obtained a low validation accuracy of around 0.103 for the model with data augmentations applied. This theoretically didn't make sense, and especially with such a bi drop in the accuracy. It turned out that we forgot to uncomment a line of code which normalizes the images after augmentation.

The most significant challenge we encountered was the limited runtime and RAM availability on Google Colab for free users. To mitigate the issue, we used a larger batch size in efforts to avoid crashing the RAM since we had a large dataset of 80,000 images. Additionally, we reduced the number of epochs to approximately 30 in order to ensure that the training completed within the available runtime. We also started saving the weights of each iteration to disk, allowing us to resume training from the most recent checkpoint in case of a runtime crash.

4 Link To GitHub

`https://github.com/MuAbCh/CNN-Plant-Disease-Classification`

References

Bacterial spot of septoria leaf spot?, a. URL `https://plantvillage.psu.edu/posts/7693-tomato-bacterial-spot-of-septoria-leaf-spot`.

Two-spotted spider mite, b. URL `https://entomology.ca.uky.edu/ef310`.

Caring for the raspberry bushes - the martha stewart blog, Sep 2020. URL https://www.themarthablog.com/2020/09/caring-for-the-raspberry-bushes.html.

Climate change fans spread of pests and threatens plants and crops, new fao study, 2021. URL https://www.fao.org/news/story/en/item/1402920/icode/.

Squash (cucurbita spp.)-powdery mildew, Apr 2023. URL https://pnwhandbooks.org/plantdisease/host-disease/squash-cucurbita-spp-powdery-mildew.

Tomato yellow leaf curl virus, Mar 2023. URL https://www.greenlife.co.ke/tomato-yellow-leaf-curl-virus/.

GS Admin. Boost plant health to reduce stress in soybeans, Feb 2021. URL https://knowmoregrowmore.com/boost-plant-health-to-reduce-stress-in-soybeans/.

Fred Alcober. Ai takes root, helping farmers identify diseased plants, Jun 2018. URL https://blog.google/technology/ai/ai-takes-root-helping-farmers-identity-diseased-plants/.

Emmarex. Plant disease dataset. https://www.kaggle.com/datasets/emmarex/plantdisease, 2018.

Chatterjee A. Savadjiev P Forghani, R. Ml algorithms interest over last 10 years. radiomics and artificial intelligence for biomarker and prediction model development in oncology. URL https://www.researchgate.net/figure/Google-trends-data-for-machine-learning-algorithms-between-2008-and-2018_fig1_334435214.

PEAT GmbH. Tomato late blight: Pests amp; diseases. URL https://plantix.net/en/library/plant-diseases/100046/tomato-late-blight/.

Amy Grant. Tomato leaf mold treatment: How to treat leaf mold of tomato plants, Sep 2022. URL https://www.gardeningknowhow.com/edible/vegetables/tomato/managing-tomato-leaf-mold.htm.

Marie Iannotti. Save your tomatoes from septoria leaf spot, Jul 2023. URL https://www.thespruce.com/identifying-and-controlling-septoria-leaf-spot-of-tomato-1402974.

Avinash Navlani. Scikit-learn svm tutorial with python (support vector machines), Dec 2019. URL https://www.datacamp.com/tutorial/svm-classification-scikit-learn-python.

Regina. Potato diseases amp; common pests, Jul 2022. URL https://plantura.garden/uk/vegetables/potatoes/potato-diseases.

Garvit Singh. Plant disease detection dataset. https://www.kaggle.com/code/garvitsingh/plant-disease-detection/input, 2023.

FAO (2021) Alcober (2018) Forghani Emmarex (2018) Singh (2023) FAO (2021) Regina (2022) ras (2020) Admin (2021) squ (2023) tom (a) GmbH Grant (2022) Iannotti (2023) tom (b) tom (2023) Navlani (2019)