**Competition Title:** Paddy Doctor: Paddy Disease Classification

**Group Members:** Alex Pietrick (ajpietrick@wpi.edu), Josh Malcarne (jrmalcarne@wpi.edu), Mira Plante (smplante@wpi.edu), Sarah McCarthy (smccarthy@wpi.edu)

Introduction

Paddy is the unprocessed form of rice, a food staple around the world. Around 60% of the attainable yield is lost each year due to diseases and pests, making crop supervision essential. Machine learning is an aiding factor in identifying crop damage. This competition gave us 10,407 images of paddy leaves with ten classes to distribute each into (nine diseases and one healthy class). We were tasked with successfully identifying whatever a plant was diseased and then correctly interpreting which disease that plant had from by processing images.

Sound Machine Learning

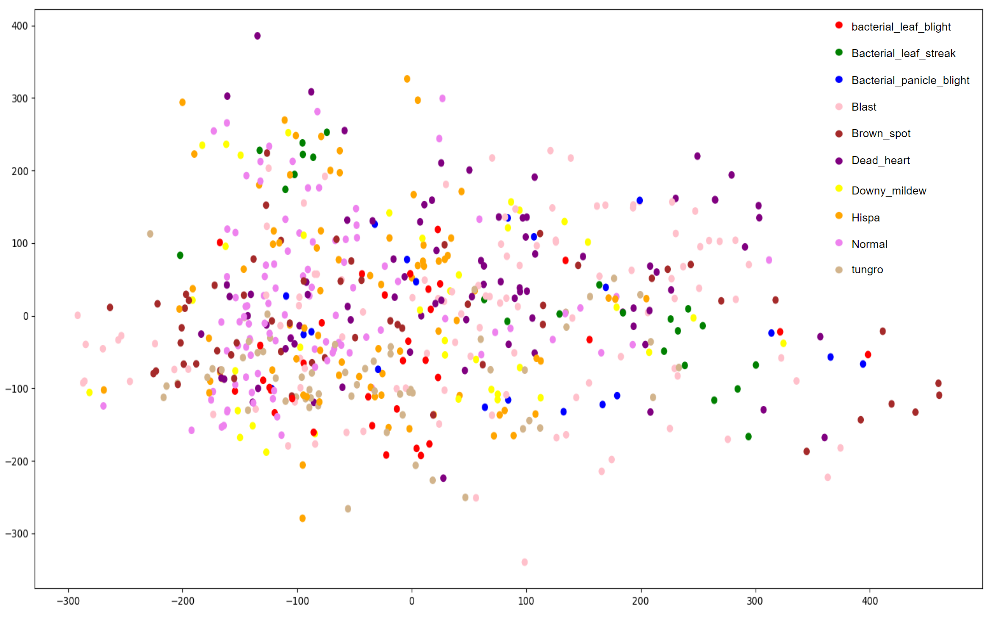
We researched applied machine learning within the plant disease detection field by reading through a few papers. The common idea is to take an image of a plant and feed it through a network to sort into different classes, whether that be simply healthy or unhealthy or more specific classes.[[1]](#footnote-0)[[2]](#footnote-1)

Decision tree learning was used frequently to tackle this classification problem. Decision tree learning is an approach to machine learning where observations will “fall” down a tree of possibilities through the use of binary factors and will eventually be sorted into a single class.[[3]](#footnote-2) For example, in our dataset, there were binary decisions that impacted what class an observation could be fitted into. These decisions ranged from if the plant is healthy or unhealthy to whatever the afflicted disease seemed to be bacterial in nature. We found that this approach was used somewhat frequently in this field and was intrigued by how it operated and sorted the observations.[[4]](#footnote-3) We didn’t have the time to try to implement this approach in our models.

Our group’s implementation of the DNN made use of callbacks. The optimizations offered by callbacks are significant to training DNN models when used appropriately. The callbacks we chose to use included Tensorboard (to track graphed training and validation loss/accuracy), early stop (to stop epochs after plateauing training loss), learning rate adjustment (to decrease learning rate on validation loss plateau), and checkpoints (to save the model’s progress in the event of interrupted training). Separately, early stop better-optimized the number of epochs the model trained for, while learning rate adjustment optimized learning rate during training, decreasing learning rate as needed and allowing SGD to better approach a relative minimum.

We generated more training data for our models by using Keras ImageDataGenerator method. We found great success with flipping and adjusting the brightness of our training images. This would give some variety in the dataset regarding the orientation of the plants and level of light exposure. As for rotating, shifting, and zooming, these methods produced images that ranged from barely different from their original images to completely compressed and mismatched. This raised the questions of what quality we wanted these images to be and how unique they needed to be to have a positive impact on our models. We feared that using this generated data would introduce unintentional biases into the machine. Due to the large training time for our models, we were not able to add any of this generated data into the training set.

For this assignment, we did not use any pre-trained models, though we do think that our task is one that could benefit from using one. We would want to use an image data mining neural-network to extract information from our images and then have the final layers of this network be dedicated to disease classification. Ideally, we would compare this approach to our from-scratch neural network to observe which ended up being more accurate.



PCA Data Visualization

The PCA for our training data can be seen in this image. For better visibility, a random sample of 600 data points were plotted and the color guide shows the label for each image. Some trends we found are that the hispa data points are concentrated between the x values -200 to 50. The normal data points are focused around x = -100 as well. However, the dead heart data leans to the right of the graph.

Models

| Baseline For our baseline model, we had the prediction for every image be the most common label in the training set, which was the healthy class. Given that there were ten distinct classes, this model performed better than randomly guessing, having a PC of ~0.17 on the training data and the testing data as well. Despite this performance, our cross entropy loss was gigantic at around 611.93 on the training data. | Shallow Model An svm was created using SKLearn multiclass for the shallow model. The model performed better than the baseline but was not terribly impressive, getting a 34% testing accuracy. The model was quite time consuming to train given the large dataset, so a random sample of 2000 images were used from the training set instead. | 3-Layer Neural Network Keras was used for the 3NN model. The model performed moderately okay, scoring 42% for testing accuracy with Kaggle. It was able to identify healthy plants well but struggled with determining what disease the plant had. Learning how to use Keras to create the machine was an important step in taking machine learning to the real world. Several features, like certain hyperparameter tuning, could have improved the model that was created beyond its current predictions. While Keras was a powerful tool, only using 3 layers seemed to restrict its potential, so a deep neural net using some aspects of Keras was used next. | Deep Neural Network For our DNN model, we chose to use ResNet50V2 (not pre-trained). The model performed well, scoring 0.88196 for testing accuracy on Kaggle. The training cross-entropy loss of the fully trained model was 0.0086 and accuracy was 0.9991, while the validation cross-entropy loss was 0.3540 and validation accuracy was 0.9111, suggesting overfitting, but no dip in validation accuracy was observed. With more time, data, and technical knowledge of DNN models, it is likely that hyperparameters could be further optimized to achieve better validation loss values and testing accuracies while reducing potential overfitting. |
| --- | --- | --- | --- |

Table of Results

| **Model/Technique** | **Train Accuracy**  (Percent Correct) | **Train Loss**  (Cross Entropy) | **Test Accuracy**  (Percent Correct) |
| --- | --- | --- | --- |
| Most Frequent Y Value | 0.169501 | 611.93407 | 0.16878 |
| SVM through Sklearn | 0.3575 | 64.25 | 0.3428 |
| 3-Layer Neural Network | 0.4409 | 1.6179 | 0.42099 |
| Deep Neural Network | 0.9991 | 0.0086 | 0.88196 |

Conclusions

In general the deep methods worked best, as proven by the 3NN and DNN accuracies over the other techniques. The Deep Neural Network model worked the best out of all of the models. This is because of the numerous hidden layers the model goes through to achieve the best accuracy possible.

References

* <https://github.com/zhixuhao/unet/blob/master/data.py>
  + Used in deep model for putting testing data into a format that could be used with model.predict()
* <https://www.analyticsvidhya.com/blog/2020/08/image-augmentation-on-the-fly-using-keras-imagedatagenerator/>
  + Used for reference with Image Augmentation Generation

Appendix

# Regarding Image Augmentation

# 

|  |  |
| --- | --- |

To show some of the reasons why rotating, shifting, and zooming transformations were problematic, we put together these figures on image augmentation. As you can see, the images are heavily warped for these three transformations and it is questionable whether the image is actually of a plant sometimes.

Shifting and rotating definitely seem to produce the worse result here. What we can do is reduce the factors of these transformations; however, the images are very similar to the original image. This again touched on the issue of quality versus quantity of data for machine learning.

Further images can be explored in the ‘Image Augmentation Visualization’ folder and the dataGeneration.py file in our submission.

# Regarding 3NN Methodology

A validation set was used to optimize the training of the machine.The set was created using seed values and image dataset extraction models found in Keras. Images were converted to the shape (256, 256, 3), flattened, passed through the ReLU activation function, and then passed through the softmax activation function to arrive at the final predictions.

# 

# Regarding DNN Methodology

This model was trained with a learning rate of 0.01 optimized by Keras Tuner, which was further reduced by a factor of 0.1 twice during training using a learning rate adjustment callback observing validation loss. Epochs were also optimized using an early stop callback observing training loss with a patience of 10. Batch size was set to 32, as our available machines couldn’t handle any higher multiples of 2^n, and regularization strength was not available for tuning. Training/validation data was split 80/20 using the split-folders import licensed by MIT, and both were fed to the model.fit() function using an object defined with image\_generator.flow\_from\_directory(). Images were reshaped to (224,224,3) as recommended by the ResNet API, and the tf.keras.applications.resnet\_v2.preprocess\_input() function was used to preprocess all training, validation, and testing data. It is also worth noting that this model was trained on the original rgb data, and not on a grayscale version of the images.

While we did not have time to explore other models in depth, potential future models include ResNet101V2 and ResNet152V2, as these models would allow us to compare the performance of our model of depth 50 with models of depth 101 and depth 152.

1. Khan, Rehan Ullah, et al. “Image-Based Detection of Plant Diseases: From Classical Machine Learning to Deep Learning Journey.” *Wireless Communications and Mobile Computing*, Hindawi, 3 June 2021, https://www.hindawi.com/journals/wcmc/2021/5541859/. [↑](#footnote-ref-0)
2. Xian, Tan Soo, and Ruzelita Ngadiran. “Plant Diseases Classification Using Machine Learning.” *Journal of Physics: Conference Series*, IOP Publishing, 1 July 2021, https://iopscience.iop.org/article/10.1088/1742-6596/1962/1/012024/meta. [↑](#footnote-ref-1)
3. Gupta, Prashant. “Decision Trees in Machine Learning.” *Medium*, Towards Data Science, 12 Nov. 2017, https://towardsdatascience.com/decision-trees-in-machine-learning-641b9c4e8052. [↑](#footnote-ref-2)
4. Rangarjan, Aravind, et al. “Crop Identification and Disease Classification Using Traditional ...” *ResearchGate*, Nov. 2021, https://www.researchgate.net/publication/356208912\_Crop\_identification\_and\_disease\_classification\_using\_traditional\_machine\_learning\_and\_deep\_learning\_approaches. [↑](#footnote-ref-3)