CAI 4104/6108: Machine Learning Engineering

Project Report: Detection of Plant Diseases in Species of Corn

Cameron Brown
(573-280-6403)
cbrown14@ufl.edu

Pablo Gonzalez
(352-779-6005)
pablogonzalez@ufl.edu

Yash Hegde
(786-800-9274)
yash.hegde@ufl.edu

Gabriel Isenberg
(239-249-1181)
g.isenberg@ufl.edu

Joshua Thomas
(702-663-4914)
joshuathomas@ufl.edu

April 26, 2024

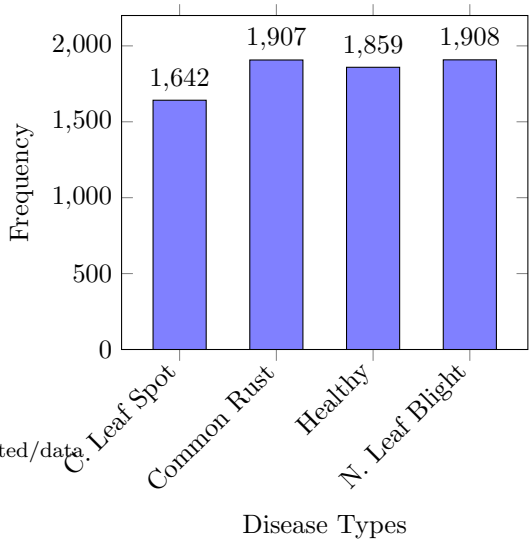
1 Introduction

Our project focuses on the efficient classification of the presence of three types of plant disease in species of corn. Detecting diseases in corn could lead to significantly greater yields for farmers, and could reduce food waste across the globe. A simple disease detection model could allow farmers to remove diseased crops before they infect other crops. To solve this problem, our team implemented a basic model pipeline, beginning with preprocessing, through training a model, through seeing the output of our results. Our team assembled a convolutional neural network (CNN) which is receives an image of a possibly diseased species of corn as input and outputs its predicted class label. Our model consistently demonstrated validation accuracy of 97% or more. In order to validate the results of our model, we compared the results of our model to accessible comparison models and baselines, including MobileNetV2 [3], most frequent class predictions, and stratified predictions. Note that our initial dataset that we had proposed in the project proposal had an extremely high bias with the dataset, which resulted in big discrepancies in the validation and training accuracies that could not be remedied through any methods taught in the scope of this course. Thus, we resorted to choosing an entirely new dataset with new methods of baselines and comparative models in our pipeline.

2 Approach: Dataset(s) & Pipeline(s)

2.1 Dataset

The dataset[1] our team chose consists of 9,145 public domain training images showcasing healthy corn, corn diseased by Common Rust, corn diseased by Northern Leaf Blight, and corn diseased by Cercospora Leaf Spot.



2.2 Pipeline

2.2.1 Preprocessing

The first step in our pipeline is preprocessing the dataset to ensure that standards for bias and classification are met. The first step of preprocessing was ensuring that we did not have biased data, and we confirmed this by ensuring that each label had a similar number of images. After validating this, we used one-hot encoding to convert the class label numbers to numbers.

[1] https://www.kaggle.com/datasets/tushar5harma/plant-village-dataset-updated/data

2.2.2 Model Architecture

Next, we move into our CNN architecture. Our model is designed and compiled through Keras and incorporates multiple convolutional, max-pooling, dense, and output layers. The general steps of our model are as follows:

1. **Convolutional Layers**: Our model features six convolutional layers, each utilizing a 3x3 kernel and the ReLU activation function, the LeCun Uniform kernel initializer, and the same padding to keep spatial dimensions intact. The first two layers consist of 16 filters, followed by 32 filters in the next two, followed by 64 filers in the final two layers.

2. **Max-Pooling Layers**: Three max-pooling layers integrate with the convolutional layers to reduce spatial dimensions to a 2x2 pooling window. This helps to reduce the computational complexity of our model and reduces the opportunity for overfitting by reducing the spatial size of representation.

3. **Flattening Layer**: The flatten layer is responsible for transforming the output of our 3D image data representation into a 1D vector, which supports the ability of our model to provide numerical classifications when faced with new images.

4. **Dense Layers, Dropout Layers**: Our network has two fully connected dense layers, with 64 and 32 neurons, both using the ReLU activation function. Between these layers, we have added dropout layers with a dropout rate of 0.25 to prevent overfitting.

5. **Output Layer**: We have a final dense output layer which outputs a probability distribution for each of the four classes, which helps to support classification and determine the most likely label for a new image.

6. **Compilation**: We chose to use the Adam optimizer with a learning rate of 0.001 when compiling our model. We use categorical cross-entropy loss because of our model's orientation towards multi-class classification.

3 Evaluation Methodology

To evaluate our model, we chose to primarily review the validation accuracy purported after training, while also ensuring that test accuracy was not overly high. We evaluated the performance of our CNN by comparing it to baseline models trained with the same image data and labels. To ensure consistency during model training, we employed identical training data and labels for both our CNN model and the baseline models, and minimized any potential sources of variability that could have influenced the comparative assessment of their performance.[1]

Furthermore, the evaluation methodology we used involved assessing the performance of each model, including our CNN, on a standardized set of validation images from the dataset. The subset was provided by the dataset and correlated with a rough train-test-validation split of 80%-2%-18%. This allowed for a direct comparison of their classification results and performance metrics under consistent testing conditions. As a result, we were able to determine the efficacy of our CNN model relative to the baseline models, providing valuable insights into its superiority in disease classification tasks.

4 Results

Our research yielded substantial insights into the effectiveness of deep learning models for image classification tasks. We obtained the following results from our primary convolutional neural network (CNN) model and the MobileNetV2 model which was used for comparison.

4.1 Baseline Comparison

The most frequent class baseline (predicting the most common class in the dataset) did not provide an adaptive measure of performance, while the stratified baseline (predicting classes in proportion to their prevalence in the dataset) offered a more dynamic and realistic comparison.

4.2 CNN Model Performance

The CNN model was trained over 15 epochs, achieving a testing accuracy of 99% and a validation accuracy of 97%. The model's performance improved consistently across epochs, as shown in Table 1. The training time decreased over successive epochs, suggesting increased computational efficiency. The loss metrics decreased substantially, indicating the model's improved predictive performance over time.

```
Epoch 1/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 58s 457ms/step - accuracy: 0.6403 - loss: 0.8044 - val_accuracy: 0.9337 - val_loss: 0.1622
Epoch 2/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 46s 397ms/step - accuracy: 0.9283 - loss: 0.2046 - val_accuracy: 0.9441 - val_loss: 0.1557
Epoch 3/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 46s 393ms/step - accuracy: 0.9482 - loss: 0.1501 - val_accuracy: 0.9574 - val_loss: 0.1118
Epoch 4/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 44s 379ms/step - accuracy: 0.9563 - loss: 0.1277 - val_accuracy: 0.9441 - val_loss: 0.1379
Epoch 5/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 43s 373ms/step - accuracy: 0.9565 - loss: 0.1240 - val_accuracy: 0.9568 - val_loss: 0.0979
Epoch 6/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 43s 370ms/step - accuracy: 0.9718 - loss: 0.0832 - val_accuracy: 0.9568 - val_loss: 0.1351
Epoch 7/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 43s 376ms/step - accuracy: 0.9558 - loss: 0.1328 - val_accuracy: 0.9508 - val_loss: 0.1224
Epoch 8/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 48s 418ms/step - accuracy: 0.9663 - loss: 0.0959 - val_accuracy: 0.9574 - val_loss: 0.1076
Epoch 9/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 43s 375ms/step - accuracy: 0.9660 - loss: 0.0844 - val_accuracy: 0.9690 - val_loss: 0.0877
Epoch 10/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 45s 386ms/step - accuracy: 0.9748 - loss: 0.0684 - val_accuracy: 0.9714 - val_loss: 0.0885
Epoch 11/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 48s 411ms/step - accuracy: 0.9778 - loss: 0.0590 - val_accuracy: 0.9763 - val_loss: 0.0772
Epoch 12/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 47s 406ms/step - accuracy: 0.9825 - loss: 0.0491 - val_accuracy: 0.9514 - val_loss: 0.1731
Epoch 13/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 49s 429ms/step - accuracy: 0.9799 - loss: 0.0603 - val_accuracy: 0.9702 - val_loss: 0.0816
Epoch 14/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 47s 406ms/step - accuracy: 0.9825 - loss: 0.0577 - val_accuracy: 0.9769 - val_loss: 0.0866
Epoch 15/15
115/115 ━━━━━━━━━━━━━━━━━━━━ 46s 400ms/step - accuracy: 0.9872 - loss: 0.0460 - val_accuracy: 0.9793 - val_loss: 0.0754
```

Figure 1: CNN Model Accuracy and Loss Over Epochs

4.3 MobileNetV2 Model Performance

The MobileNetV2 model, utilizing transfer learning, reached similar testing and validation accuracies in just 7 epochs. The training durations and loss metrics, as reported in Table 2, were higher compared to the CNN model, reflecting the difference in architecture and pre-training influence.[2]

```
Epoch 1/10
115/115 ━━━━━━━━━━━━━━━━━━━━ 84s 635ms/step - accuracy: 0.8654 - loss: 0.3356 - val_accuracy: 0.9544 - val_loss: 0.1213
Epoch 2/10
115/115 ━━━━━━━━━━━━━━━━━━━━ 67s 584ms/step - accuracy: 0.9746 - loss: 0.0705 - val_accuracy: 0.9708 - val_loss: 0.0801
Epoch 3/10
115/115 ━━━━━━━━━━━━━━━━━━━━ 65s 567ms/step - accuracy: 0.9760 - loss: 0.0646 - val_accuracy: 0.9660 - val_loss: 0.0970
Epoch 4/10
115/115 ━━━━━━━━━━━━━━━━━━━━ 62s 542ms/step - accuracy: 0.9868 - loss: 0.0366 - val_accuracy: 0.9562 - val_loss: 0.1200
Epoch 5/10
115/115 ━━━━━━━━━━━━━━━━━━━━ 70s 613ms/step - accuracy: 0.9906 - loss: 0.0268 - val_accuracy: 0.9593 - val_loss: 0.1329
Epoch 6/10
115/115 ━━━━━━━━━━━━━━━━━━━━ 64s 558ms/step - accuracy: 0.9911 - loss: 0.0225 - val_accuracy: 0.9678 - val_loss: 0.0871
Epoch 7/10
115/115 ━━━━━━━━━━━━━━━━━━━━ 64s 556ms/step - accuracy: 0.9943 - loss: 0.0164 - val_accuracy: 0.9726 - val_loss: 0.0838
```

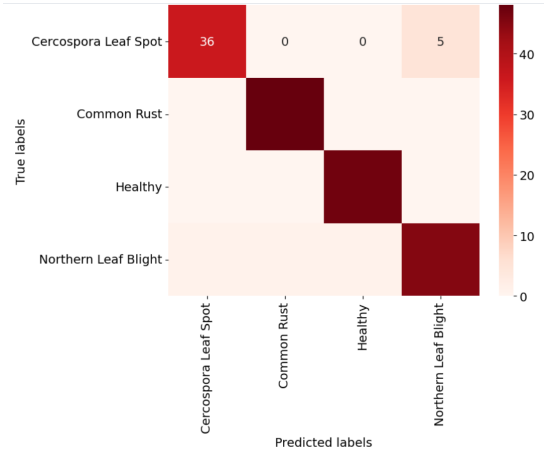
Figure 2: MobileNetV2 Model Accuracy and Loss Over Epochs

4.4 Prediction Confidence

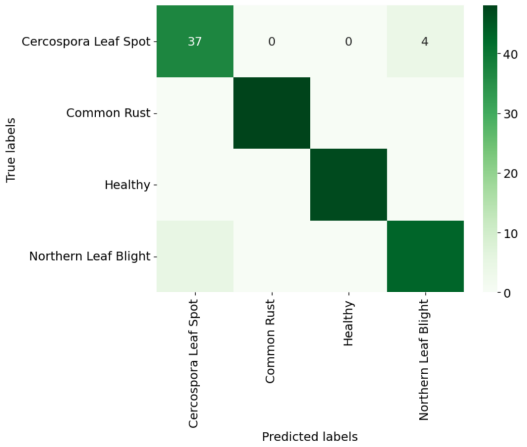
The CNN model consistently made predictions with a 100% confidence level. In contrast, the MobileNetV2 model predictions had varying confidence levels, demonstrating the model's sensitivity to the data it was presented with during testing.

4.5 Confusion Matrices Analysis

The confusion matrices provide a granular view of model performance across classes. As shown in Figure 1 and Figure 2, both models displayed high true positive rates, but with some misclassifications that varied between them.



(a) Confusion Matrix for the CNN model



(b) Confusion matrix for the MobileNetV2 model

4.6 Comparative Baseline Analysis

Compared to the baselines, both the CNN and MobileNetV2 models showed superior performance. The baselines are useful for understanding the chance level of accuracy in our dataset; however, both deep learning models surpassed these benchmarks by significant margins.

```
Baseline Model Accuracy on Validation Set: 0.260790273556231
Baseline Model Accuracy on Test Set: 0.2608695652173913
```

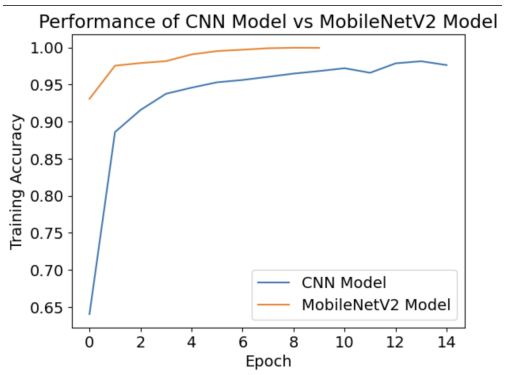
Figure 4: Accuracy purported by our first baseline, most frequent class prediction

```
Stratified Baseline Model Accuracy on Validation Set: 0.24316109422492402
Stratified Baseline Model Accuracy on Test Set: 0.28804347826086957
```

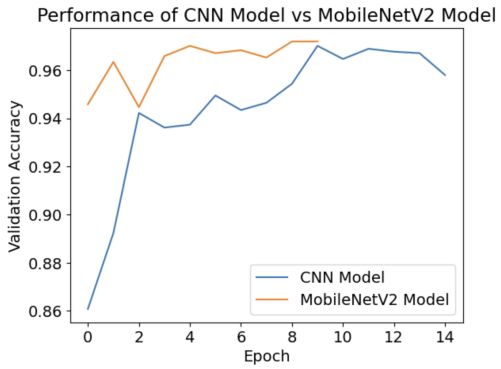
Figure 5: Accuracy purported by our first baseline, stratified class estimation

4.7 MobileNetV2 vs CNN Performance Analysis

The following graphs show the performance of our MobileNetV2 model compared to the CNN model. Note that the general trend involves the CNN Model having an initial jump in training accuracy as it learns after the initial epoch and makes advances after each epoch while the MobileNetV2 model already has a high training accuracy due to its weight and has already been taught through millions of images before continuing on in the scope of our project.



(a) Training accuracy performance for both models



(b) Validation accuracy performance for both models

5 Conclusions

The CNN model our team developed successfully identified various plant diseases found in corn species. Throughout the development of our model, our team adhered to stringent methodology and received positive findings that displayed our model's accuracy. We aimed to develop a CNN that could distinguish between three corn diseases, leveraging a dataset of over 9,000 images. Across key metrics, our CNN model surpassed the benchmark model findings provided by MobileNetV2. Additionally, our model architecture, crafted within Keras, adhered to principles conducive to effective feature extraction and hierarchical representation learning.

Because our CNN performed better than the baseline model, we can confidently conclude that the CNN was successfully trained and used. In Figure 1, the accuracy of the CNN model is shown to reach 99% with a validation accuracy of 97%. However, Figure 2 shows how the baseline model was not able to make decisions with the same confidence as the CNN model. This heightened accuracy highlights the model's ability to correctly classify corn samples into their respective disease categories with remarkable precision. Moreover, the precision of our CNN model was notably superior, indicating its proficiency in minimizing false positives. By effectively identifying diseased corn samples while minimizing misclassifications of healthy ones, the CNN exhibited a precision that surpassed that of the basic baselines. Similarly, the recall of our CNN model surpassed that of the baselines, showcasing its capability to effectively capture true positive instances while mitigating false negatives. This ensured that the CNN could reliably identify diseased corn samples, thus improving its utility in practical disease detection scenarios.

In summary, the consistent outperformance of our CNN model in terms of accuracy, precision, and recall highlights its effectiveness and dependability in classifying diseases. This positions it as a powerful asset in agriculture, capable of reducing crop losses and ensuring food security.

References

[1] Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Al-Dujaili, Ye Duan, Omran Al-Shamma, J. Santamaría, Mohammed A. Fadhel, Muthana Al-Amidie, and Laith Farhan. Review of deep learning: concepts, CNN architectures, challenges, applications, future directions. *Journal of Big Data*, 8(1):53, March 2021.

[2] Ricardo Ribani and Mauricio Marengoni. A survey of transfer learning for convolutional neural networks. 2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T), IEEE, 10 2019.

[3] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks, 2019.