Cassava Leaf Disease Classification

Heath Thompson
College of Engineering
University of Idaho
Moscow, ID
thom2197@vandals.uidaho.edu

***Abstract*—This document discusses the classification of cassava leaf images into five disease categories using both transfer learning and deep learning approaches to develop a neural network architecture. Further discussion details the development and implementation of this predictive model and how to improve upon it in the future.**

I. Introduction

The focus of this project is to accurately predict Cassava leaves' state of health using a predictive model. The different states include four categories of diseases as well as a fifth state that is the state of a healthy leaf or one without noticeable signs of disease. These states of health are predicted using both transfer learning and deep learning approaches within a machine learning model. This model's architecture is of the form of a figuratively six-layer neural network. The approach to the model involves appending a four-layer densely connected neural network to a pre-trained InceptionResNetV2 model implemented by Keras.

II. Model Development and Approach

A. Transfer Learning Approach

Transfer learning is implemented because it greatly reduces the development time needed to produce an effective model. Here, we use transfer learning with a Keras application model; in particular, we used InceptionResNetV2. This pre-trained model was selected due to its accuracy. It, of all the available models provided by Keras, was the most accurate in predicting the right class labels for images. The model, though highly accurate, requires the most time to compute out of this selection of Keras models. This is an understood compromise for the model's effectiveness in accuracy.

B. Outputs Layers Approach

Alongside the implementation of InceptionResNetV2, is a series of densely connected layers. There are four layers with each layer first through last with the node counts of 100, 50, 10, and 5 respectively. These layers further reduce the inputted data into the final five categories of disease for the leaves in the provided dataset. The final layer outputs the decision of the model in the form of a list of 5 values. Whatever list index corresponds to the largest value in the list is the class label, thus, disease for which an image is classified. The number of layers and nodes within this network was determined using experimentation. Using a step-by-step approach, incremental changes were made to these numbers until the best accuracy was found using the training data available.

C. Paramter Settings

A majority of parameters were set to their default values. This is because the default values made the most sense for this application. For the model's loss function, categorical cross-entropy was chosen. This is due to the fact that this project focused on two known ideas: 1: this is a classification problem, 2. there are five class labels. For the model's optimization, the Adam optimizer was chosen. Although there were experiments done with the SGD optimizer, it was determined that the Adam optimizer was a better fit because it adapts by changing its learning rate automatically based on its input. Finally, for the activation functions, Selu was chosen for the hidden layers and Softmax for the output layer. Selu was chosen because it provides a solution to almost all activation function drawbacks known today along with the incorporation of normalization. It is also easier on computer resources to compute than a majority of other activation function options. Softmax was chosen for the output layer for its well-known use as such.

III. Results

A. Model Training and Validation Performance

My model's performance is illustrated in Figure 1 and Figure 2 seen below. The model is not accurate enough to produce any dependable or informative results. As seen in the figures, the model's accuracy rates are at or right under 70%. The model's loss sits at or above 85%. There results are not sufficient enough to gain valuable insight on Cassava leaf classification and therefore shouldn't be used for research purposes.

B. Model Test Performance

Test class label were not produced due to an error that was unsolvable in the time given. The error occurred while loading the testing images dataset. For some reason unknown, after running the test_data_generator, zero images were loaded into memory. Also, when storing the image IDs into test_df, it is found that these IDs are ten-digit long strings whereas they should only be nine-digit. The training dataset is ten digits, but not the testing dataset. This leads to a belief that the path to this data may be corrupted in some way. Again, this problem was unsolvable, therefore there was no predictions made on the testing data. As seen in the program, switching to reading in the test images using a dataframe of their filenames, the images still couldn't be loaded. An assumption to this error may be that those filenames in that dataframe don't exist in the directory provided. It is known the provided directory is correct and it is also believed to be known that the dataframe stores the filesnames in the provided directory. This leads to the current state of delay.

GRAPH ONE

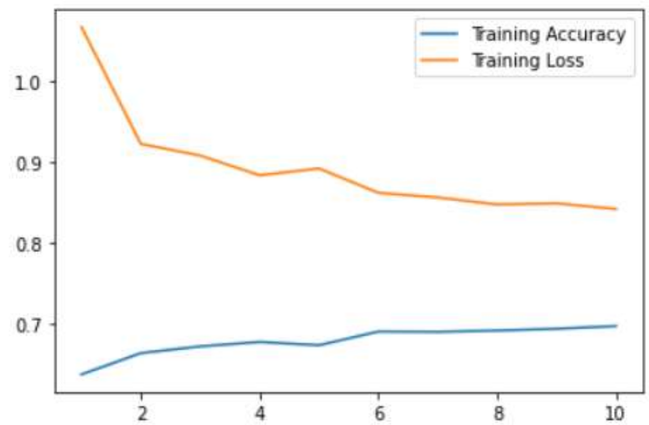

Fig. 1. Training Set Accuracy and Loss

GRAPH TWO

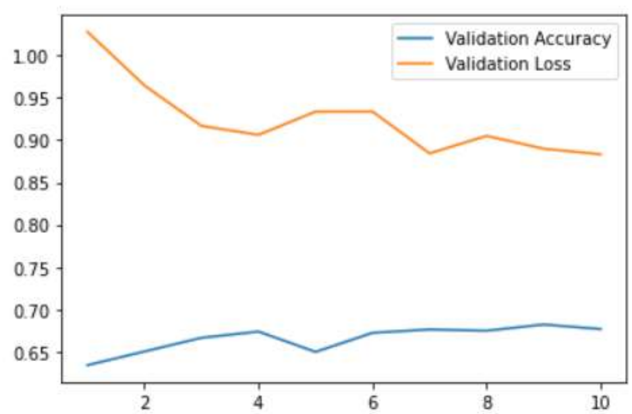

Fig. 2. Validation Set Accuracy and Loss

IV. CONCLUSION

A. *Restrospective*

In conclusion, the predictive model performed subpar. More experimentation and research would provide a better understanding in model architecture development; that is, determining the number of nodes and layers that would provide the most optimal results. Alongside an improvement in model architecture is a better understanding and knowledge of data manipulation and preprocessing. If the data was given to the model in a more effective way, the model would produce optimized results. In further detail, more augmentation may have been done to produce more training data for model development and fitting. Adding more epochs may have also slightly increased the accuracy of the model as well. Lastly, a better understanding of the available tools provided by Python and its machine learning libraries will provide a much quicker development for future projects.

B. *Leaning Outcome*

In developing this project, much was learned about not only model development, but also preprocessing and understanding the results after the model has made its predictions. It was shocking to see how much these pre- and post-processes effect the outcome of a project and whether or not it is successful. Finally, it was learned that things don't go as planned all of the time, so better time management should be put in place to allow for barriers as described previously to be solved.