Plant Disease Classification

Joshua Martin, Christian Pensabene

Github : https://github.com/JTM119/DataMining_TermProject

Dataset : https://www.kaggle.com/datasets/asheniranga/augmented-leaf-dataset

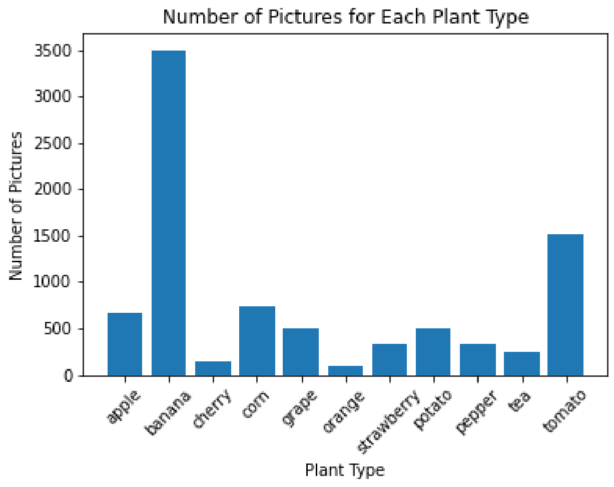
Abstract :

In this paper we describe the computer vision task of image classification on a diseased plant dataset. With 11 classes of plants and 39 classes of diseases. To achieve this task we utilized different CNN model architectures with different levels of dropout, as well as SVM and Random Forest models with the inputs being the output of a CNN intermediate layer. We also experimented with creating random forest and SVM inputs with principal component analysis(PCA). We also experimented with edge detection as a data augmentation method. The best result that we achieved on the dataset was with a Random Forest model in which the inputs were the output of a ResNet101V2 transfer learning CNN model architecture.

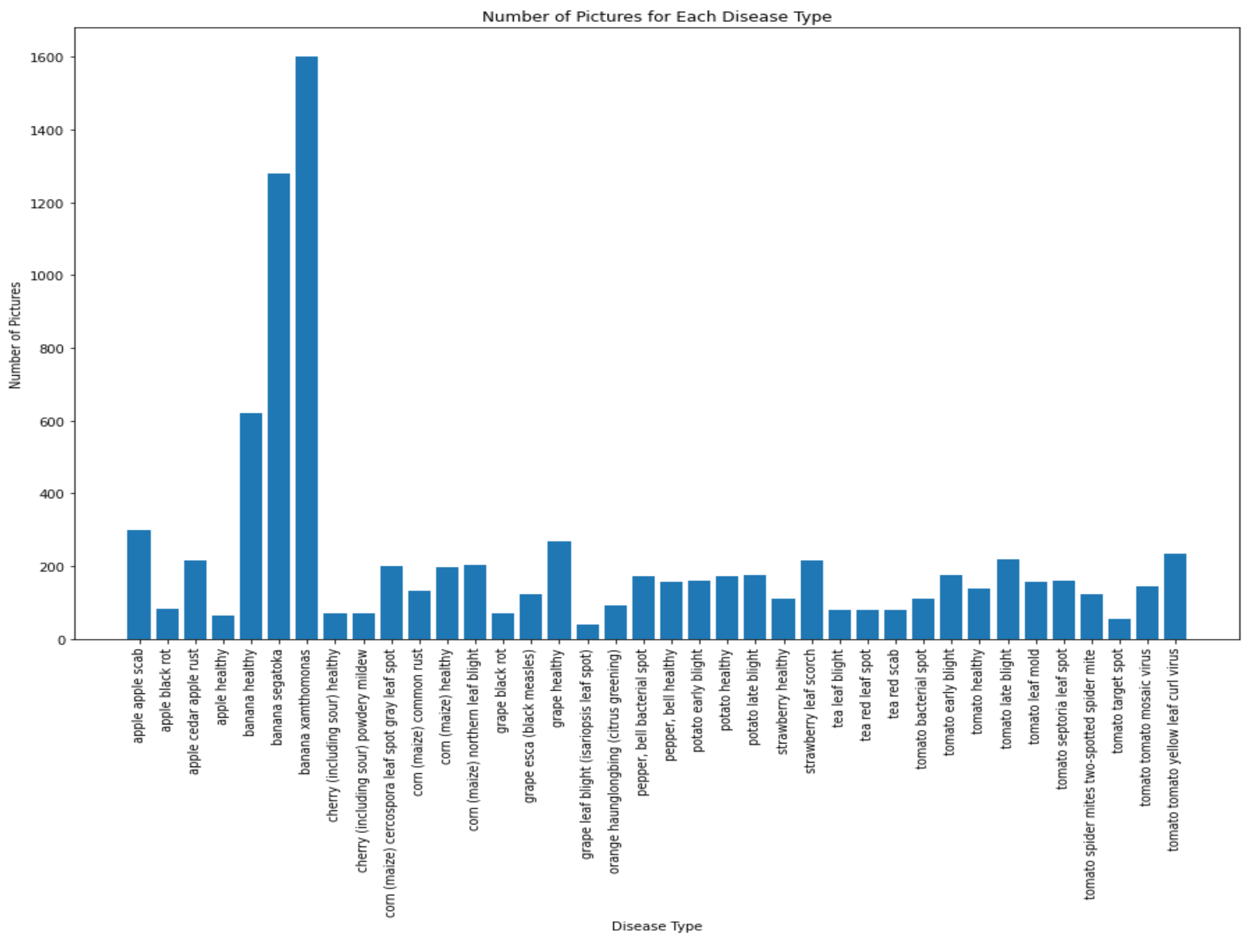
Introduction :

The purpose of this project is to try applying various computer vision algorithms on a plant disease image classification dataset. The motivation behind this project is to serve as a way to detect diseased plants before the disease, fungus, or bacteria gets a chance to spread to other crops. Since the dataset has examples for healthy leaves for most of the plant types this algorithm would be able to make this distinction. The methods used in this project include an exploration of CNN architectures varying from shallow networks, deeper custom networks, and finally utilized transfer learning with other image classification networks. We also experimented with using PCA and intermediate layers of the CNN as an input to a random forest network and an SVM network.

Data Exploration and Analysis:



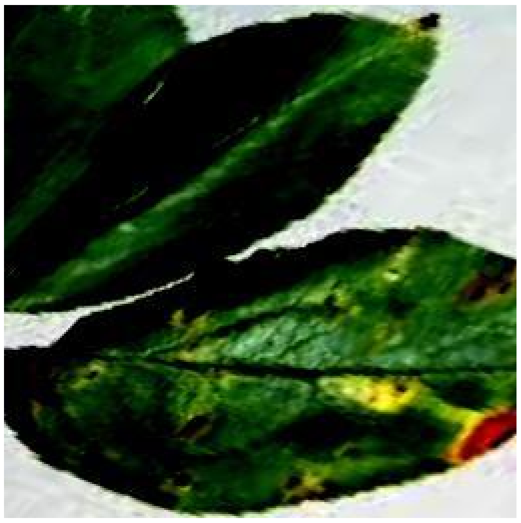
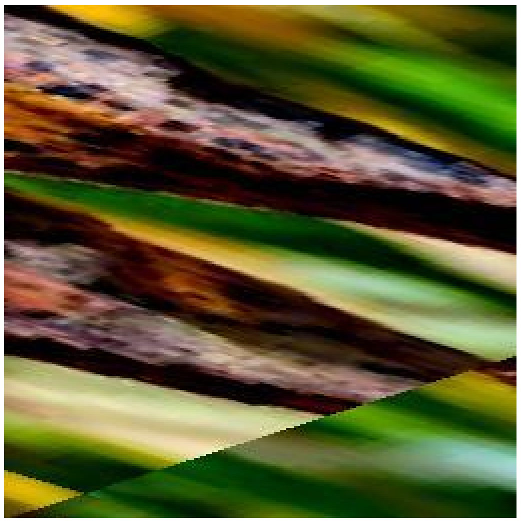
This dataset is an augmented dataset. This is important because the augmentations made to the data expanded the original dataset to create enough data to train with, however the quality of the augmented images was reduced. The image dataset was found on kaggle and features 11 different plant types: apple, banana, potato, corn, tomato, cherry, grape, strawberry, tea, pepper, orange. The dataset also features 38 different disease classes for these plants. This dataset is imbalanced in several different ways. The first is in the quantity of each plant type. The figure to the left is a histogram showing the frequency of each plant type in the



dataset. We can clearly see the dataset is heavily skewed toward a few plants : apple, banana, corn, and tomato. We also see that there are very few instances of other classes like cherry and orange.

Another problem with this dataset is the balance of diseases per plant. Figure 0 in the appendix shows a set of histograms depicting the frequency of disease by plant type. Classes like the orange plant have only one disease type while classes like tomato have 10. Furthermore the quantity of each disease type is also extremely imbalanced. The histogram on the previous page shows the frequency of each disease type in the dataset. This issue of class frequency and representation is a problem that needed to be addressed throughout the experimentation process. The primary means of addressing this issue has been using class weights proportional to the frequency of the class in the dataset.

Another issue in this dataset is consistency between images of the same class. An example of this is shown at the bottom of this page. These two images both belong to the same class. These two images also clearly have very few similar features. This is expected since there is a lot of variability between plants of the same type, but it will also present a challenge for this task if the model struggles to find features identifying a class. One method of dealing with this issue is including additional features as inputs. For example, in order to assist the model we can add an additional channel of binary values outlining the edges detected in the image.

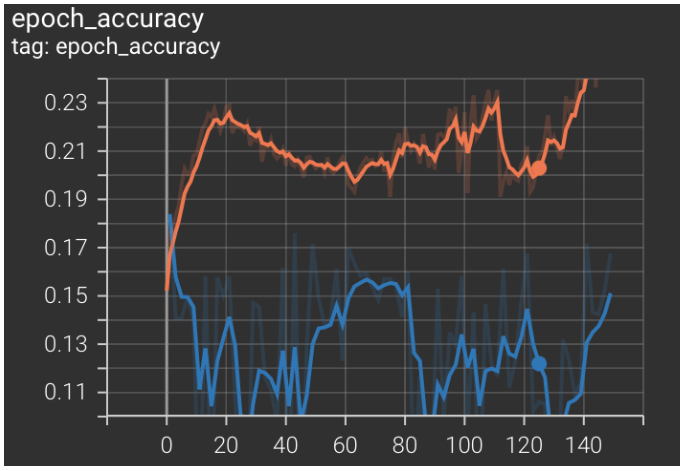
ML Methods and CNN Architectures :

There were three primary models used over the course of experimentation: various CNN architectures, a random forest, and an SVM. Formatting the data for the random forest and the SVM was a challenge in this process because they do not accept image inputs. There were two main methods of formatting the data for these models. The first method was to use principal component analysis for dimensionality reduction. For this we reduced the different color channels down to vectors and concatenated them as an input to the different models. The SVM had an overall accuracy of 14%. The random forest had an accuracy of .21%. Neither of these models performed very well. This is likely because dimensionality reduction on images forces the model to lose information from the input and these models do not capture positional information very well.

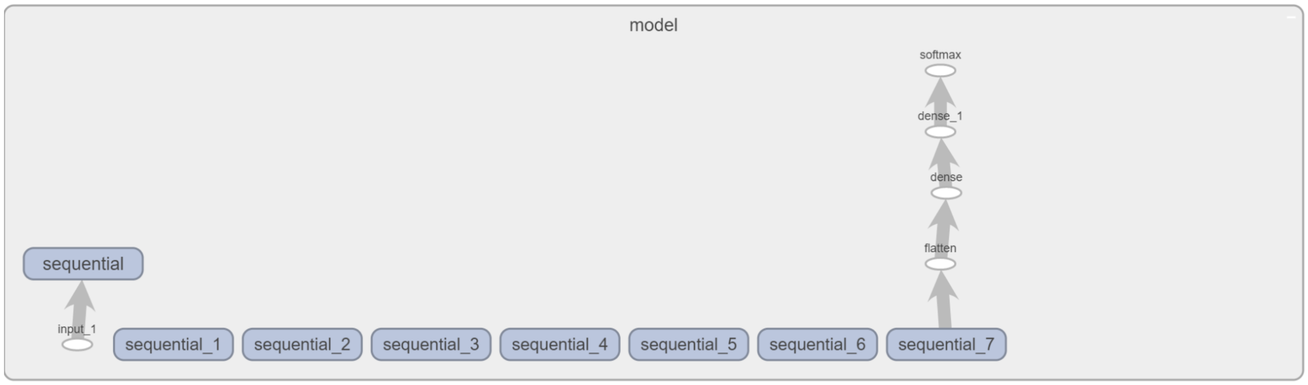
The second method of feeding information into the random forest and the SVM was to take the output of an intermediate layer of a CNN and use that to feed the information into the random forest and SVM. This means that the input vector is created with attention to spatial features which allows the models to better classify the inputs. To do this we built our CNN networks with an additional dense layer to take the output of the CNN and convert it to a vector of length 100 that could be fed into these other models. Once we had trained an optimal CNN network it would be possible to test the performance of this method.

The CNN architectures performed better than the models created using PCA for dimensionality reduction. This was expected because CNNs are designed to extract visual information through the use of filters. These filters allow pattern detection directly from the spatial data. For an image classification system this is particularly useful since it allows the model to consider all color channels and extract many types of useful information.

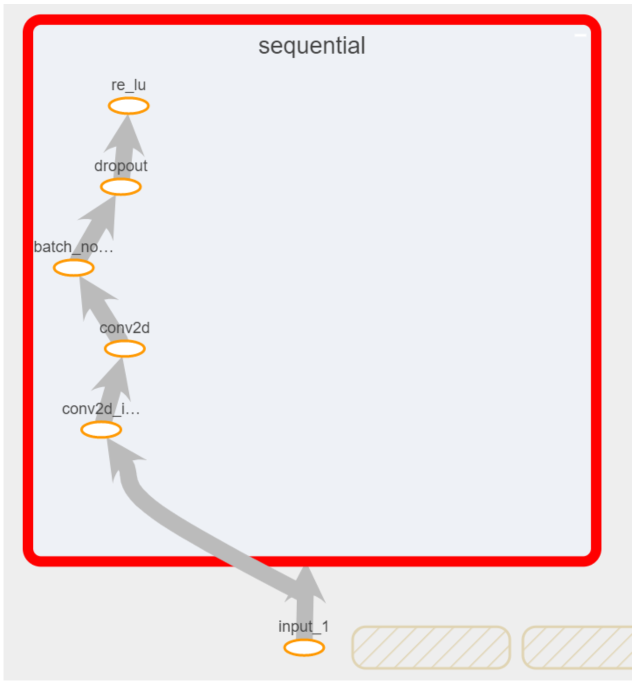
The exploration into CNNs for this project began with a shallow model. The shallow architecture utilizes a single CNN block followed by 2 dense layers. The single CNN block creates a vector of length 256 which is then fed into the dense layers and finally a softmax layer to create an output vector of probabilities. The figure to the left shows the training and validation accuracy for the shallow model as it trained. The blue line is the validation accuracy. We can see that the accuracy was volatile and fluctuated between .1 and .15. Later models adjust to make the training curves smoother by using a smaller learning rate.



Further CNN models used a deeper architecture shown below. The basic architecture is composed of several identical downblocks. Each downblock is composed of a convolutional layer, a batch normalization layer, a dropout layer, followed by a ReLU activation layer. The entire model structure is shown below. The structure of each of the downblocks (sequential layer) in this model is shown at the beginning of the next page.

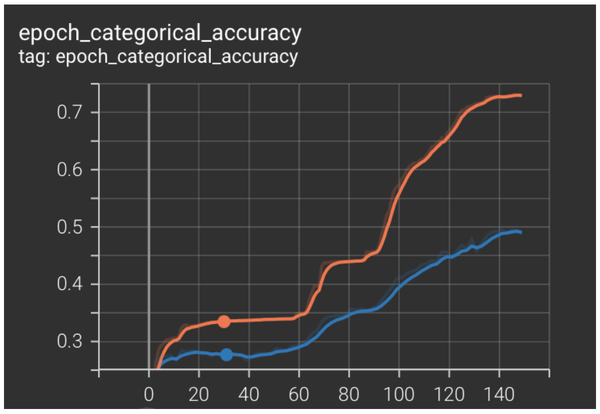


The accuracy of the model with this architecture, referred to as Basic_CNN, did not achieve higher than 20% validation accuracy, but it did reach up to 88% for training accuracy. Because this model had such drastic overfitting, we performed a search for the proper model architecture and dropout values to minimize this issue. This included reducing the size of the model to include fewer downblocks (Small_CNN and Tiny_CNN) each with smaller convolutional layers, including l2 regularization, and dropout. We performed this search for dropout values of 0, .05, .1, and .15. The best performance of this group came from the basic model utilizing l2 regularization with a dropout of .15. Even in this case the validation accuracy did not reach above 20.5%. This model is only slightly better than the other models because it reduced the overfitting on the model.



Overfitting became one of the primary problems with this task. The variety of the images in each class meant that models had difficulty generalizing but quickly overfit because the dataset was small. In order to try and give the model more information to differentiate classes we augmented the dataset by including a separate channel for edges. We performed a search similar to the one done before on this new dataset. The models trained with this new dataset achieved marginally better validation accuracy with values between .22 and .2376. The best model still remained the basic architecture with l2 regularization and .15 dropout, achieving a validation accuracy of .2376. The results of both of these experiments can be seen in figure 1 in the appendix.

After these methods still yielded poor results we decided to attempt transfer learning with standard image classification networks. We used four pretrained models for this task: MobileNetV2, ResNet50V2, ResNet101V2, and InceptionV3. MobileNetV2 and Inception V3 resulted in validation accuracies between 25 and 27%. This is another marginal improvement over the previous models. The two ResNet models performed extremely well. The ResNet50V2 model reached an accuracy of 48% while the ResNet101V2 model reached an accuracy of 49.06%. The training accuracies for ResNet101V2 are shown below.



One final experiment was performed using the model with the best performance. We ran the inputs through our best CNN model which was the ResNet101V2 model. We took the outputs of the dense layer of this model and trained a random forest and a linear svm on them. The accuracy of these models was greater than any of the CNN models. For the random forest we got an accuracy of 69% and for the SVM we got an accuracy of 68%. When exploring the precision, recall, and f1-scores of these models it became clear that the random forest model generalized better across all of the classes. The SVM seemed to mimic the imbalance of the dataset more than the random forest. The random forest performed the best on the banana classes, getting an f1 score of 92, 94, and 92. For corn common rust we got a 100 precision and an 85 recall, and for tea leaf blight we got 100 recall and 86 precision.

Conclusion :

The best method to solve this task was to apply transfer learning to the ResNet101V2 model and extract an intermediate layer to feed into a random forest**.** This resulted in an accuracy of 69%. The SVM model had similar performance to the random forest, but the random forest was more capable of dealing with the imbalanced dataset. The SVM would be more useful if we expected to see the same distributions in the dataset in the real world.

Future Works :

If future work is to be done on this task the primary goal should be to work on improving the dataset. This is important because with the quality of this data it is extremely difficult to be able to reliably predict certain classes. Some variability in the classes is to be expected since plants of the same species will look differently. The datasets will also likely include different times of day and weather conditions which will add some variability. The primary changes to the dataset need to be aimed at improving the quality of the images and balancing out the class distributions. With a better balance of classes, a new method of classification can be tried. This new method involves including the plant type as an input and predicting the disease. This will limit the options for the disease type. The reason this will not work well with the current dataset is because certain plant types have very few diseases and very few instances. These instances will not be enough for training and will result in overfitting. Another future work on this task could be experimenting with alternative loss functions. In particular, categorical focal loss would be useful for this task. This is because it takes into account the balance of the classes in the dataset.

Contributions :

Josh created the CNN model architectures while Christian created the random forest and SVM scripts. Christian also wrote the script that allowed us to extract the output of a layer of the model and feed it into the random forest and SVM. Josh wrote the scripts to apply transfer learning to the pre-built image classification networks. We worked together on the data exploration and analysis. We also worked together to train all of the models to split the time for training in half. Once we had our trained models we both discussed what changes to make to the models and what ideas to try next.

Appendix:

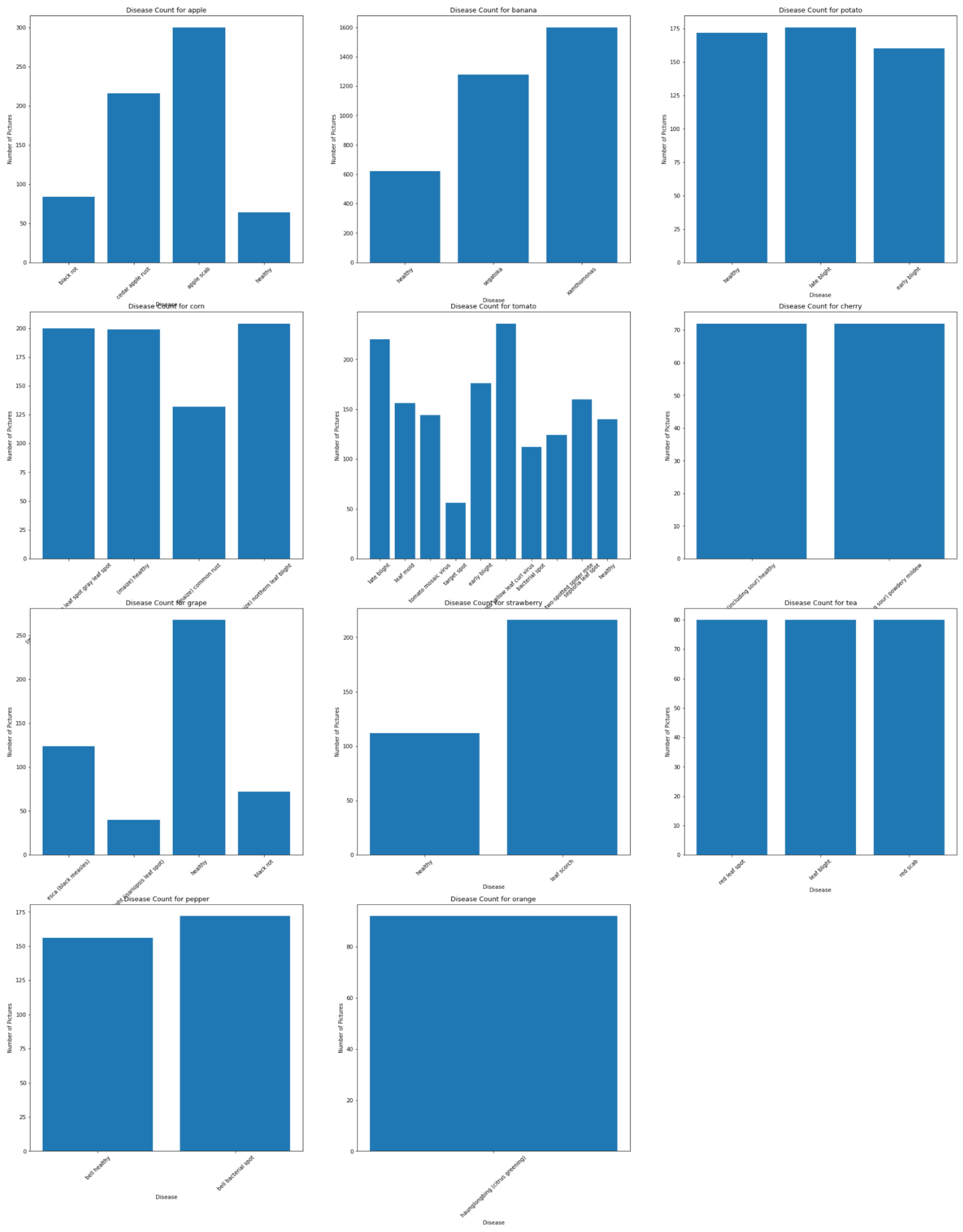


Figure 0 : The figure above shows the breakdown of disease frequency for each plant

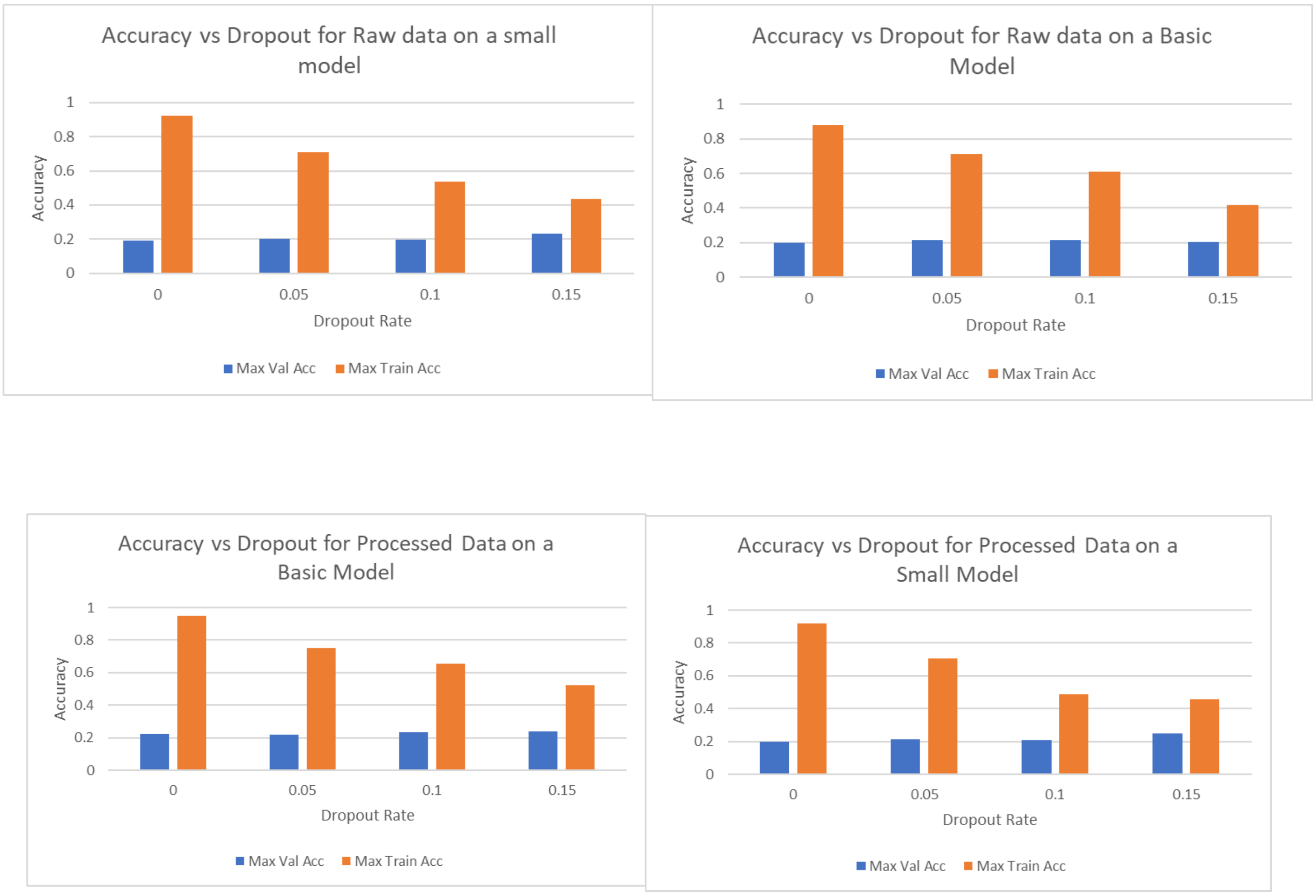


Figure 1 : The figures above show the results of the search for the optimal combination of dataset information, CNN architecture, and dropout