New Plant Disease Detection

CSML1020 Course Project

Jerry Khidaroo  
 CSML1020 – Group 3  
York University  
Toronto, ON Canada  
[jkhidaroo@gmail.com](mailto:jkhidaroo@gmail.com)

Paul Doucet  
 CSML1020 – Group 3  
York University  
Toronto, ON Canada  
 [pjdoucet@gmail.com](mailto:pjdoucet@gmail.com)

ABSTRACT

This paper will explore the classification of plant images to identify new plant diseases using a machine learning model.

The dataset was obtained from Kaggle and consists of over 87,000 rgb images of healthy and diseased crop leaves labeled by plant and disease type in 38 different classes.

CCS CONCEPTS

• Artificial Intelligence • Machine Learning   • Image Classification

1 Introduction

The project is a supervised multi-class image classification analysis to identify plant diseases by the image of their leaves.

The dataset consists of 38 categories of images which are divided into training and validation folders.

2 Existing Work

|  |  |
| --- | --- |
| Plant Desease Classifictaion-VGG16  <https://www.kaggle.com/wiwidsetiawan/plant-desease-classifictaion-vgg16> | Example of classification using the VGG16 pre trained model |
| ImageNet: A Large-Scale Hierarchical Image Database, 2009.  <https://ieeexplore.ieee.org/document/5206848> | ImageNet database documentation |
| Fork of Plant Diseases Classification Using incep3  <https://www.kaggle.com/vimaladit/fork-of-plant-diseases-classification-using-incep3> | Example of classification using the Inception Version 3 pre trained model |
| Plant Diseases Classification Using AlexNet  <https://www.kaggle.com/vipoooool/plant-diseases-classification-using-alexnet> | Example classification using the AlexNet pre trained model |

3 Methodology

3.1 Data Preparation

The dataset was downloaded from the Kaggle website: <https://www.kaggle.com/vipoooool/new-plant-diseases-dataset>

The image dataset was well prepared and did not require any modification.

The images were divided into training and validation folders: 70295 training images and 17572 validation images.

The folder names were parsed to obtain some data exploration information.

Table 1: Dataset Parsed from Category Folder Names



3.2 Data Exploration

The dataset consists of images of plant leaves in various conditions. The following graphs show the distribution of this data.

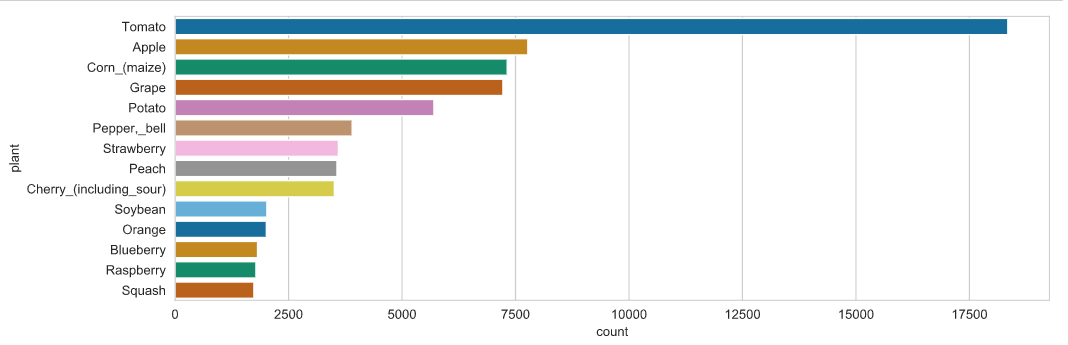


Figure 1: Number of images by plant

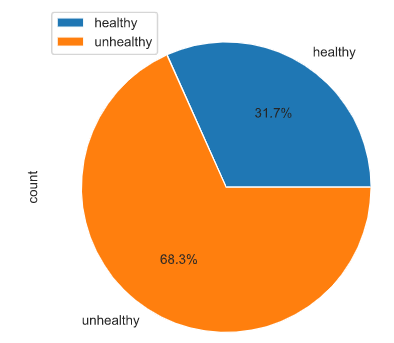


Figure 2: Relative image percentages by health status

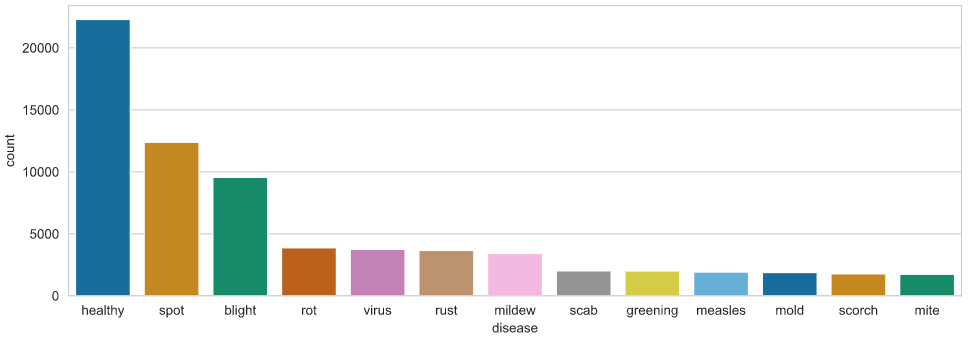


Figure 3: Number of images by disease

3.3 Data Preprocessing

The following data preprocessing methods will be evaluated to determine the best data augmentation for our input layer:

* Random Shear
* Random Brightness
* Random Zoom

The following methods were omitted because the dataset was already pre-processed with those methods:

* Random Horizontal Shift
* Random Vertical Shift
* Random Horizontal Flip
* Random Vertical Flip

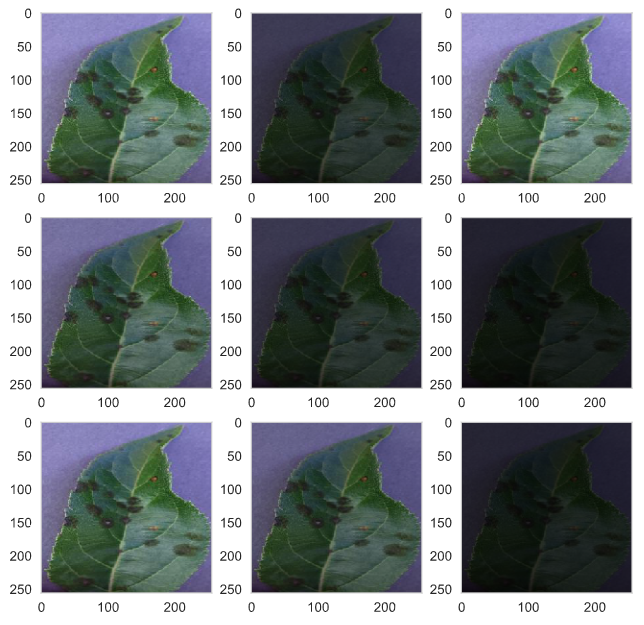


Figure 4: Data Augmentation Visualization for Random Brightness

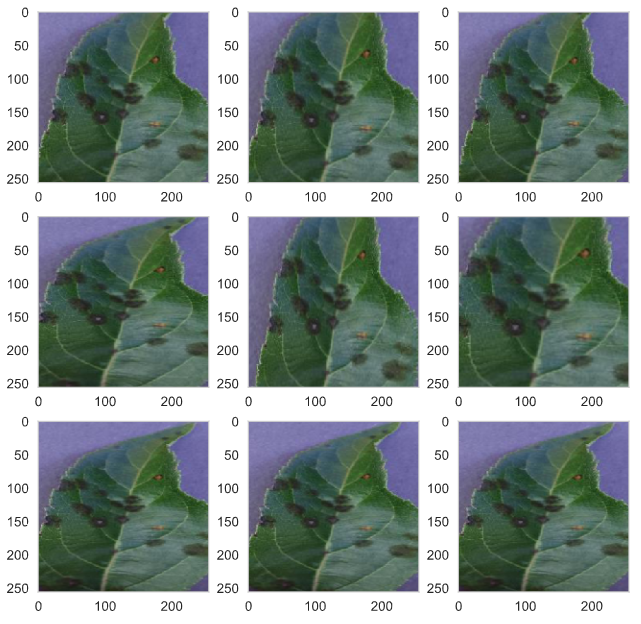


Figure 5: Data Augmentation Visualization for Random Zoom

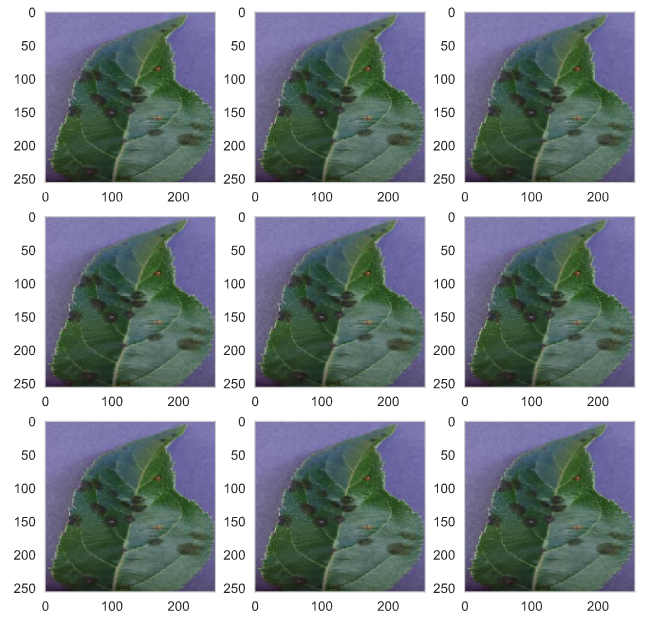
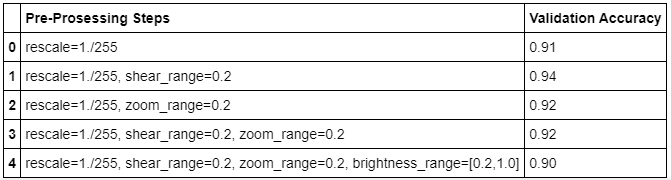


Figure 6: Data Augmentation Visualization for Random Shear

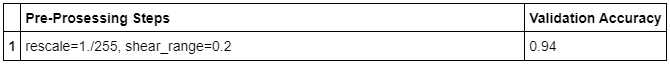
Table 2: Preprocessing Results Based on Baseline VGG16 Model with Transfer Learning



3.4 Data Augmentation Methods Selected

The results obtained for the preprocessing indicate that the shear range method gave the best results and will be used going forward.

Table 3: Data Augmentation Method Selected



3.5 Model Evaluation & Selection

The authors will be evaluating custom defined models, several pre-trained models with transfer learning as well as evaluating the results of tuning the hyper-parameters using the GridSearch function.

3.5.1 Custom Defined Models

All models in this paper are based on the convolutional neural network architecture.

3.5.1.1 Custom Defined Model A

The first iteration of the custom defined model had a convolution layer, max pooling layer, batch normalization layer and a flatten layer.

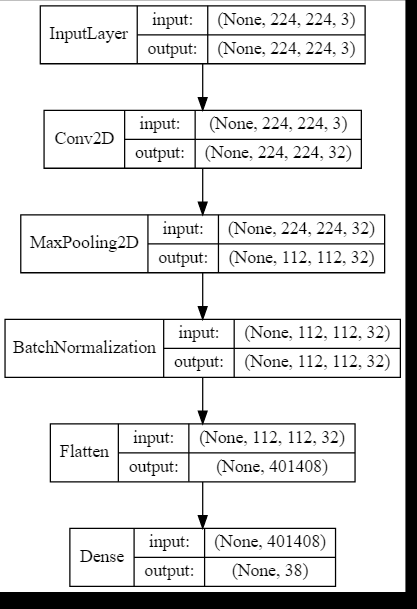


Figure 7: Custom Defined Model A

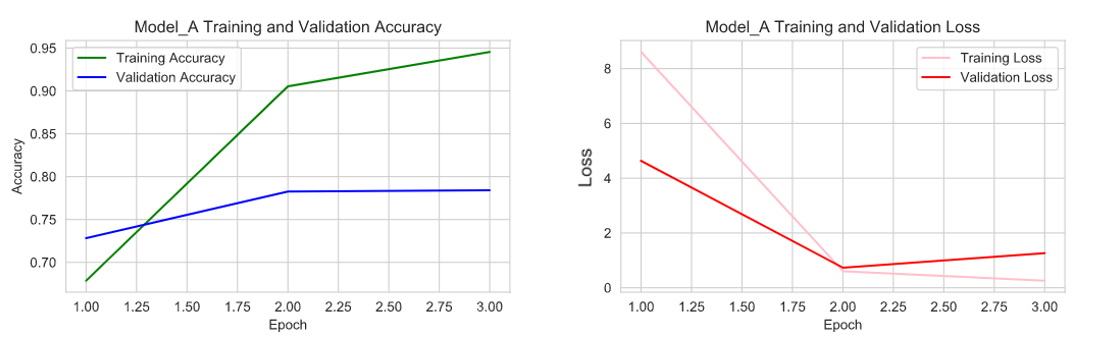


Figure 8: Training/Validation Accuracy and Loss for Model A

3.5.1.2 Custom Defined Model B

The second model tested added a dropout layer and batch normalization before the output layer.

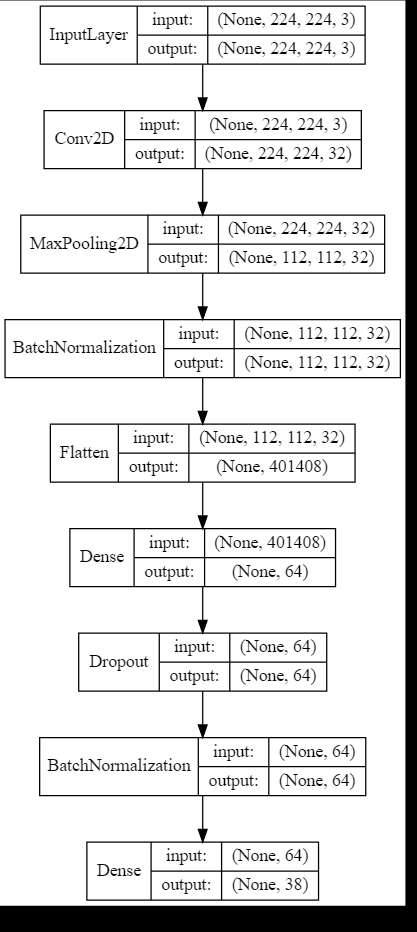


Figure 9: Custom Defined Model B

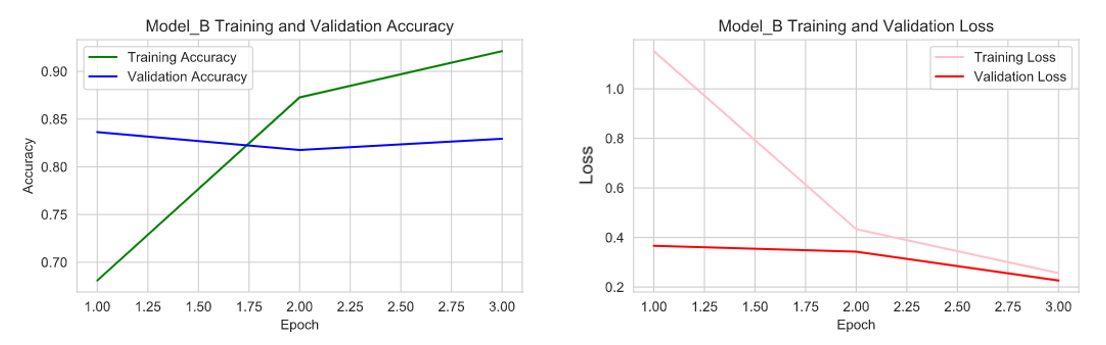


Figure 10: Training/Validation Accuracy and Loss for Model B

3.5.1.3 Custom Defined Model C

The third model trialed saw two additional convolutional layers, with max pooling and batch normalization.

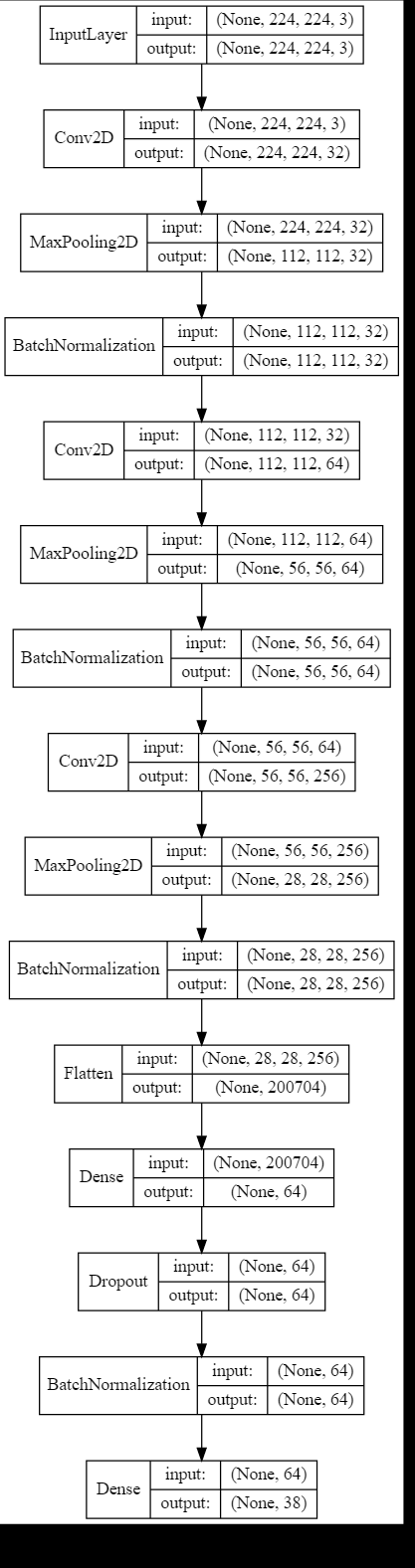


Figure 11: Custom Defined Model C

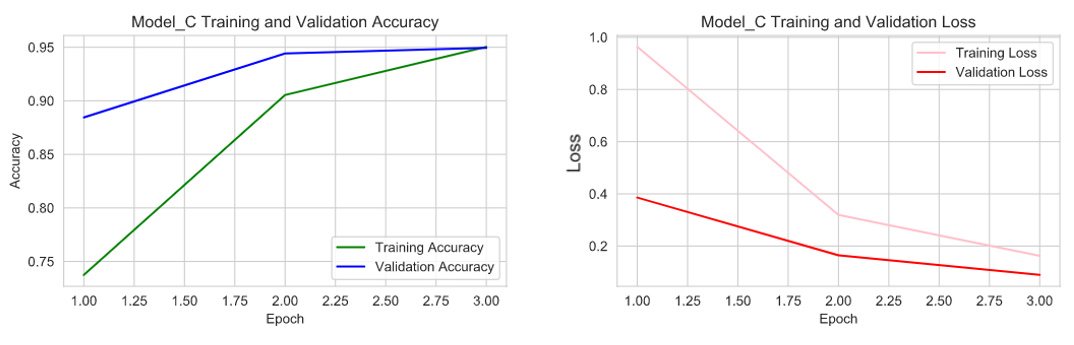


Figure 12: Training/Validation Accuracy and Loss for Model C

3.5.1.4 Custom Defined Model D

For the fourth and final custom defined model, a dense layer, dropout layer and a batch normalization layer were added between the second and third convolutional layers.

For the fourth and final iteration of the custom defined model, a dropout layer was added.

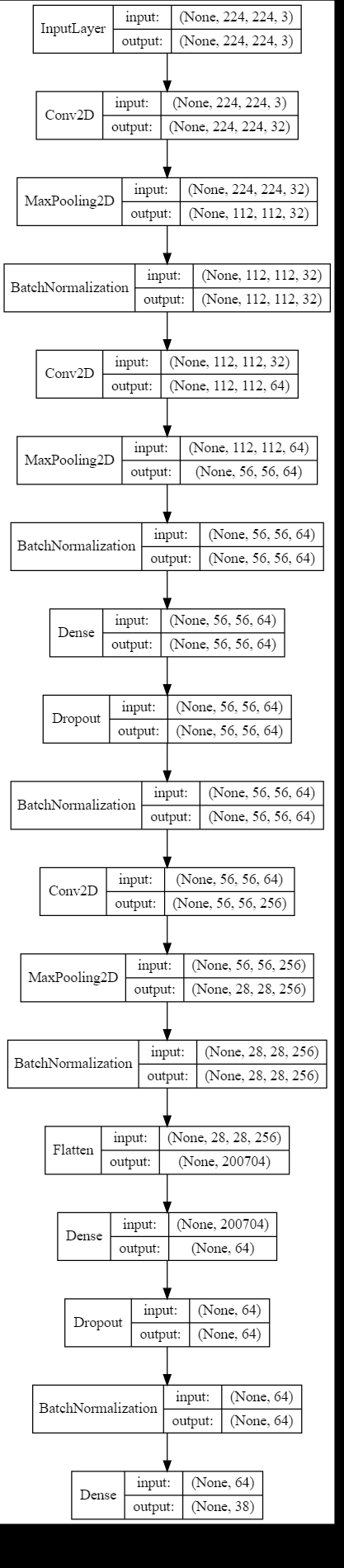


Figure 13: Custom Defined Model D

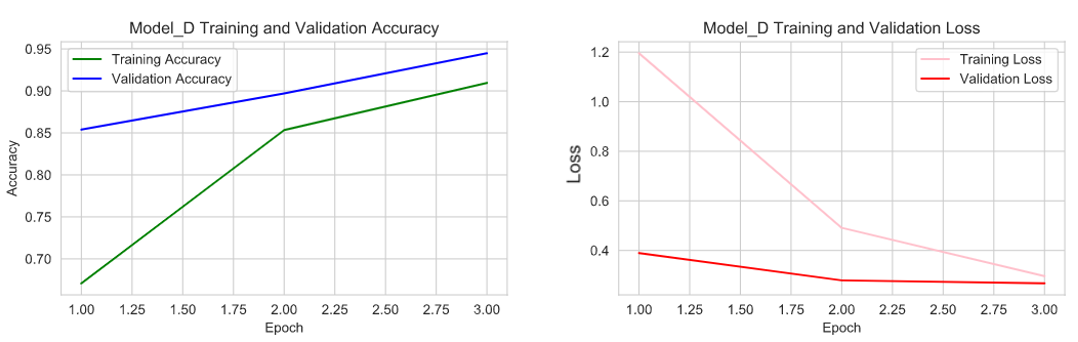
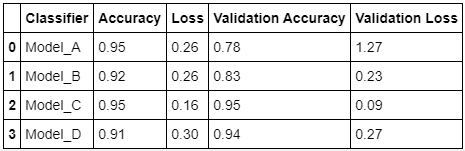


Figure 14: Training/Validation Accuracy and Loss for Model D

3.5.1.5 Custom Defined Model Results Summary



The Model\_C iteration gave the best accuracy for the custom defined classifiers and will be used for further evaluation.

3.5.2 Benchmarks for Pre-Trained Models with Transfer Learning

The accuracy and loss were evaluated for each of the four base pre-trained models: VGG16; ResNet50; InceptionV3; Alexnet .

3.5.2.1 VGG16 Base Model with Transfer Learning

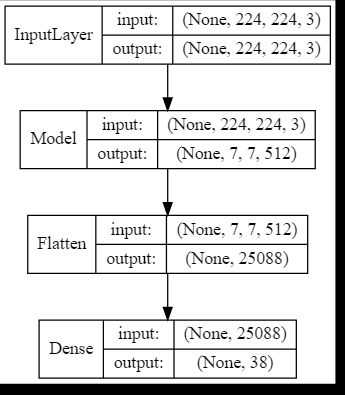


Figure 15: VGG16 Base Model with Transfer Learning

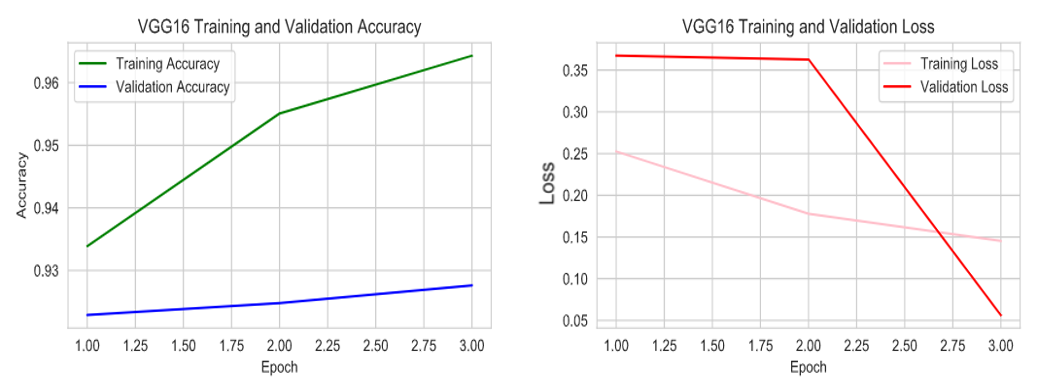


Figure 16: Training/Validation Accuracy and Loss for VGG16 Base Model

3.5.2.2 ResNet50 Base Model with Transfer Learning

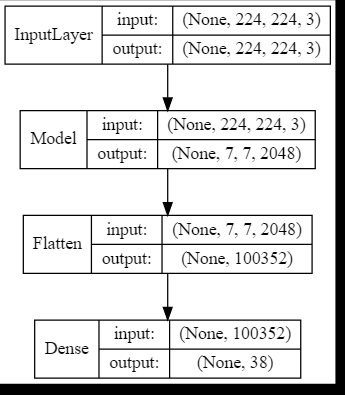


Figure 17: ResNet50 Base Model with Transfer Learning

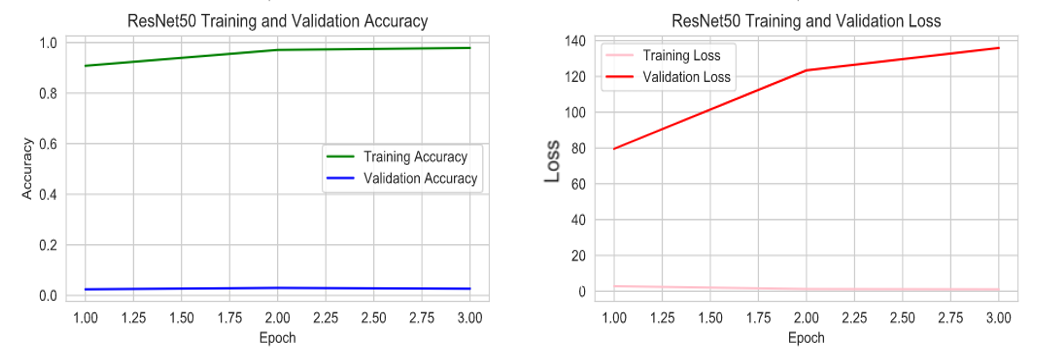


Figure 18: Training/Validation Accuracy and Loss for ResNet50 Base Model

3.5.2.3 InceptionV3 Base Model with Transfer Learning

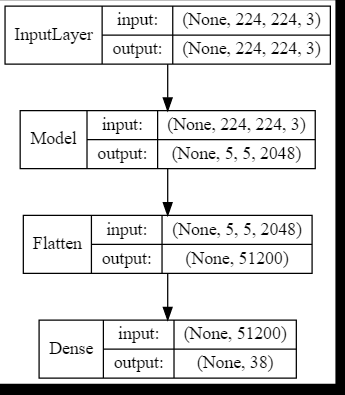


Figure 19: InceptionV3 Base Model with Transfer Learning

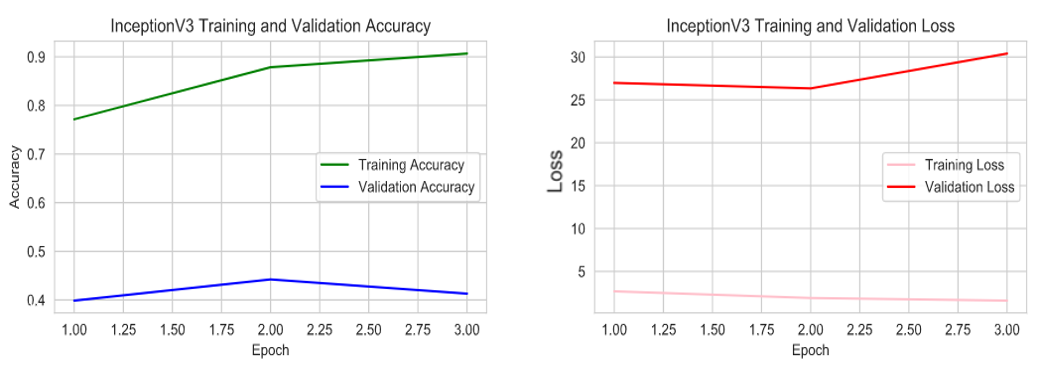


Figure 20: Training/Validation Accuracy and Loss for InceptionV3 Base Model

3.5.2.4 Alexnet Base Model with Transfer Learning

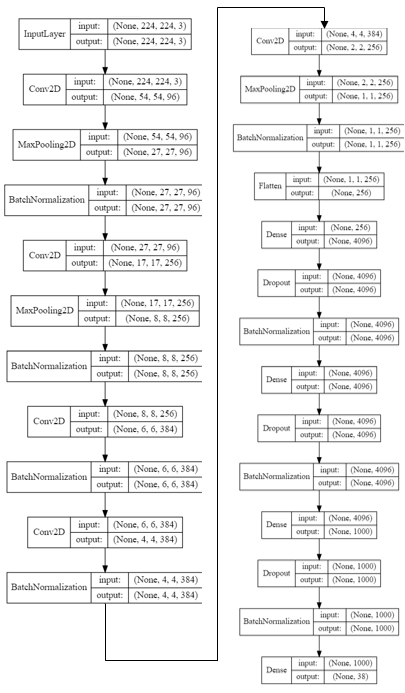


Figure 21: AlexNet Base Model with Transfer Learning

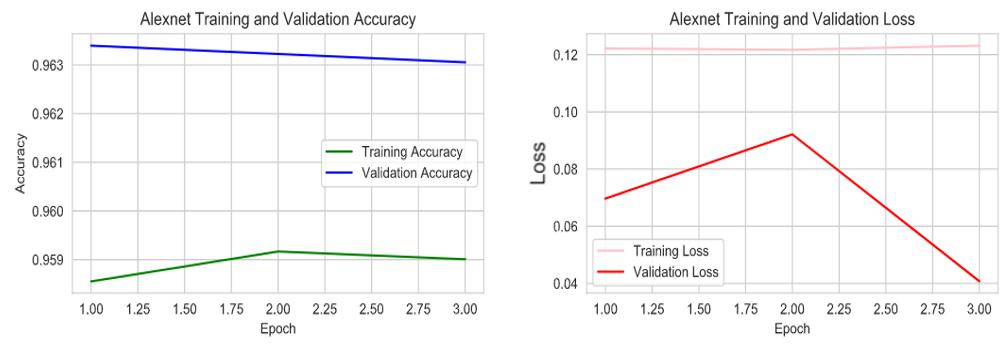
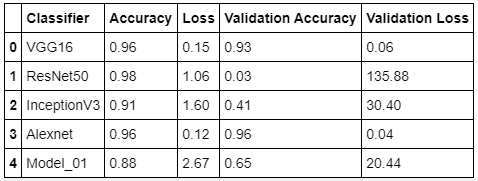


Figure 22: Training/Validation Accuracy and Loss for AlexNet Base Model

3.5.2.5 Pre-Trained Model Results Summary



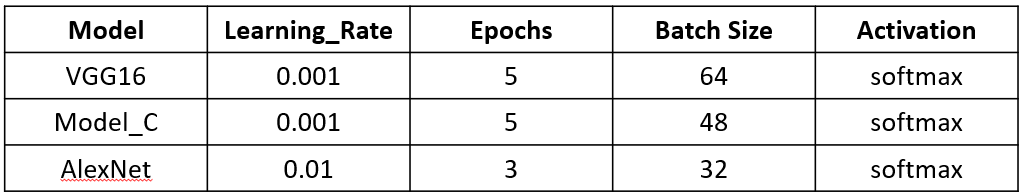
VGG16, Alexnet gave the best validation accuracy and will be used along with Custom Defined Model\_C in moving forward with our evaluation.

3.5.3 Hyperparameter Tuning

The hyperparameters were tuned using the GridSearch function with the following parameter ranges:

* Learning Rate: 0.01, 0.001, 0.0001
* Number of epochs: 3, 5, 10
* Batch size: 32, 48, 64
* Activation: “softmax”

Table 4: Best Hyperparameters



**3.5.3.1 VGG16 Model Using Best Hyperparameters**

The VGG16 pretrained model using the best found hyperparameters using gridsearch gave the following results, shown in the graph below.

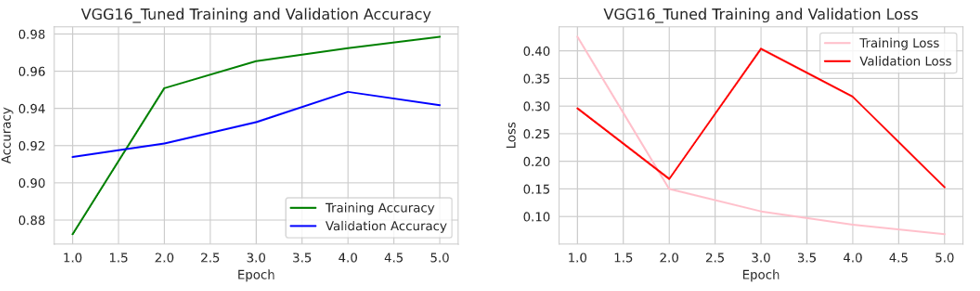


Figure 23: Training/Validation Accuracy and Loss for Tuned VGG16 Model

**3.5.3.2 Custom Defined Model\_C Using Best Hyperparameters**

The custom defined model “C” with the best found hyperparameters using gridsearch gave the following results, shown in the graph below.

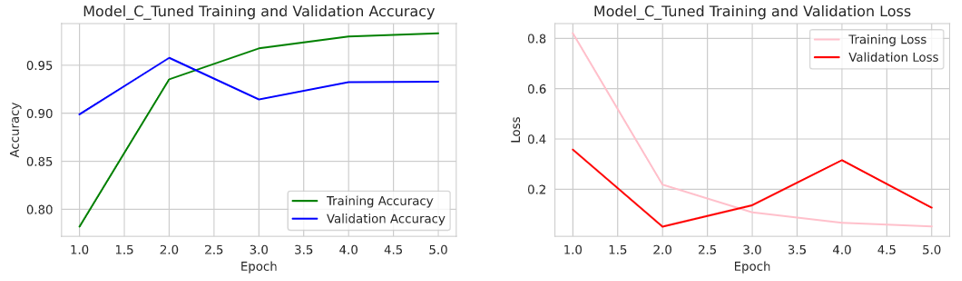
****

Figure 24: Training/Validation Accuracy and Loss for Tuned Custom Defined Model C

**3.5.3.3 AlexNet Model Using Best Hyperparameters**

The AlexNet pretrained model using the best found hyperparameters using gridsearch gave the following results, shown in the graph below.

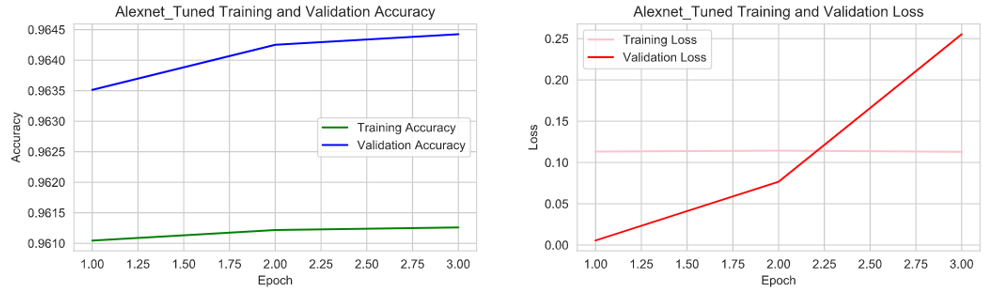
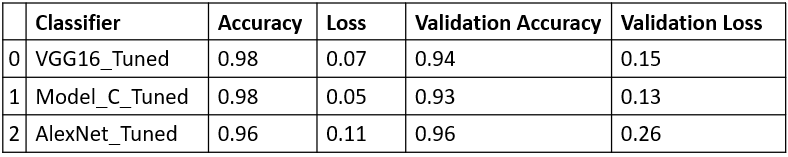
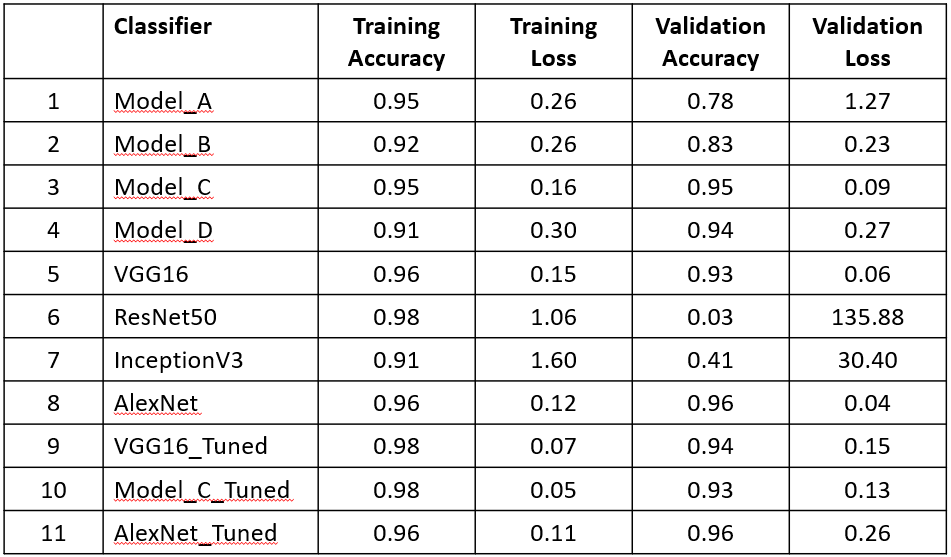


Figure 25: Training/Validation Accuracy and Loss for Tuned AlexNet Model

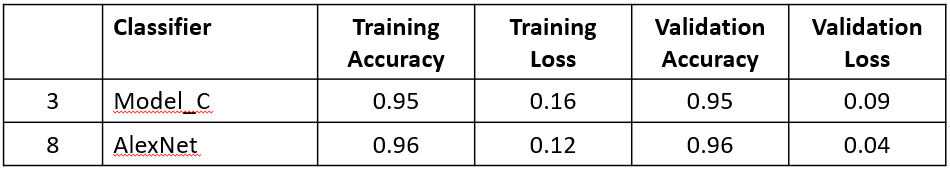
**3.5.3.4 Hyperparameter Tuning Benchmarks**



**3.5.4 Summary of Benchmark Results for All Models**



3.5.5 Models Selected



3.5.6 Example Prediction



Figure 26: Example Class Prediction Using Model C

4 Next Steps

The following will be considered for next steps:

* Ensemble model evaluation
* Deploying a detection application with one of the selected models
* Spark scaling to speed up the processing

5 Conclusion

The paper succeeded in producing two models that performed reasonably well for predictive analysis goals. Further study may be conducted in the future to attempt to further optimize the results.

ACM Reference format:

Jerry Khidaroo, Paul Doucet. 2020. New Plant Disease Detection. In *Proceedings of CSML1020 (Winter’2020). York University, Toronto, ON, Canada,*

ACKNOWLEDGMENTS

We would like to thank Karthik Kuber, PhD, MS for his constructive suggestions during this course and development of this research work.

REFERENCES

[1] Wiwid Setiawan, 2020. T Plant Desease Classifictaion-VGG16 <https://www.kaggle.com/wiwidsetiawan/plant-desease-classifictaion-vgg16>

[2] Vimal Adit. 2019. Fork of Plant Diseases Classification Using incep3 <https://www.kaggle.com/vimaladit/fork-of-plant-diseases-classification-using-incep3>

[3] Jason Brownlee 2019, Introduction to Python Deep Learning with Keras <https://machinelearningmastery.com/introduction-python-deep-learning-library-keras/>

[4] Antrixsh Gupta 2020, How to mount Cloud Storage bucket with GCP compute engine <https://medium.com/@antrixsh/how-to-mount-cloud-storage-bucket-with-gcp-compute-engine-ba7c95ad5349>

[5] Keras Documentation, Keras Applications <https://keras.io/api/applications/>

[6] Jason Brownlee 2019, How to Configure Image Data Augmentation in Keras <https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/>

[7] ACM Website, ACM Master Article Template <https://www.acm.org/publications/proceedings-template>

[8] Somnath Chatterjee, 2020. Transfer learning and ensemble <https://www.kaggle.com/somnath796/transfer-learning-and-ensemble>

[9] Image-Net 2016 Stanford Vision Lab, Imagenet

<http://www.image-net.org/>

[10] Samarth Agrawal, 2019 Hyperparameters in Deep Learning, <https://towardsdatascience.com/hyperparameters-in-deep-learning-927f7b2084dd>

[11] N. Silberman and S. Guadarrama, 2016. TensorFlow-Slim image classification model library, <https://github.com/tensorflow/models/tree/master/research/slim>

[12] Kaiming He and Xiangyu Zhang and Shaoqing Ren and Jian Sun, 2015,

Deep Residual Learning for Image Recognition,arXiv preprint arXiv:1512.03385, <https://github.com/KaimingHe/deep-residual-networks>

[13] Muneeb ul Hassan 2018, AlexNet – ImageNet Classification with Deep Convolutional Neural Networks, <https://neurohive.io/en/popular-networks/alexnet-imagenet-classification-with-deep-convolutional-neural-networks/>

[14] Muneeb ul Hassan 2018, VGG16 – Convolutional Network for Classification and Detection - <https://neurohive.io/en/popular-networks/vgg16/>

[15] Julio Borges, 2019, DeepStack: Ensembles for Deep Learning

<https://github.com/jcborges/DeepStack>