New Plant Disease Detection

CSML1020 Course Project

Jerry Khidaroo
CSML1020 – Group 3
York University
Toronto, ON Canada
jkhidaroo@gmail.com

Paul Doucet
CSML1020 – Group 3
York University
Toronto, ON Canada
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 problem we will examine is a supervised multi-class image classification problem. The goal is to investigate which supervised machine learning models will give the best results in classifying the images from our dataset in the predefined categories.

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 dataset did not need any manipulation as it was previously divided into a useable directory structure for training and validation as well as several test images.

Table 1: Dataset Parsed from Category Folder Names

	plant	condition	count	status	disease
0	Tomato	Target_Spot	50	unhealthy	spot
1	Tomato	Early_blight	50	unhealthy	blight
2	Apple	healthy	50	healthy	healthy
3	Tomato	healthy	50	healthy	healthy
4	Blueberry	healthy	50	healthy	healthy
5	Grape	healthy	50	healthy	healthy
6	Peach	healthy	50	healthy	healthy
7	Cherry_(including_sour)	Powdery_mildew	50	unhealthy	mildew
8	Tomato	Leaf_Mold	50	unhealthy	mold
9	Apple	Black_rot	50	unhealthy	rot
10	Squash	Powdery_mildew	50	unhealthy	mildew
11	Corn_(maize)	Northern_Leaf_Blight	50	unhealthy	blight
12	Strawberry	Leaf_scorch	50	unhealthy	scorch
13	Pepper,_bell	healthy	50	healthy	healthy
14	Orange	Haunglongbing_(Citrus_greening)	50	unhealthy	greening
15	Potato	Late_blight	50	unhealthy	blight
16	Tomato	Late_blight	50	unhealthy	blight
17	Strawberry	healthy	50	healthy	healthy
18	Tomato	Tomato_Yellow_Leaf_Curl_Virus	50	unhealthy	virus
19	Corn_(maize)	Common_rust	50	unhealthy	rust
20	Raspberry	healthy	50	healthy	healthy
21	Tomato	Tomato_mosaic_virus	50	unhealthy	virus
22	Pepper,_bell	Bacterial_spot	50	unhealthy	spot
23	Cherry_(including_sour)	healthy	50	healthy	healthy
24	Tomato	Septoria_leaf_spot	50	unhealthy	spot
25	Peach	Bacterial_spot	50	unhealthy	spot
26	Apple	Cedar_apple_rust	50	unhealthy	rust
27	Tomato	Bacterial_spot	50	unhealthy	spot
28	Grape	Esca_(Black_Measles)	50	unhealthy	measles
29	Grape	Leaf_blight_(Isariopsis_Leaf_Spot)	50	unhealthy	spot
30	Corn_(maize)	Cercospora_leaf_spot_Gray_leaf_spot	50	unhealthy	spot
31	Apple	Apple_scab	50	unhealthy	scab
32	Grape	Black_rot	50	unhealthy	rot
33	Potato	healthy	50	healthy	healthy
34	Corn_(maize)	healthy	50	healthy	healthy
35	Potato	Early_blight	50	unhealthy	blight
36	Soybean	healthy	50	healthy	healthy
37	Tomato	Spider_mites_Two-spotted_spider_mite	50	unhealthy	mite

3.2 Data Exploration

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

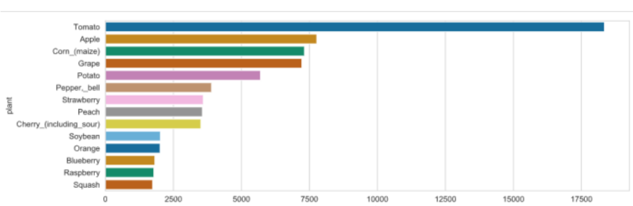


Figure 1: Number of images by plant

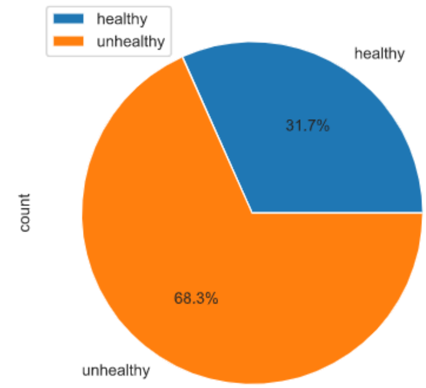


Figure 2: Relative image percentages by health status

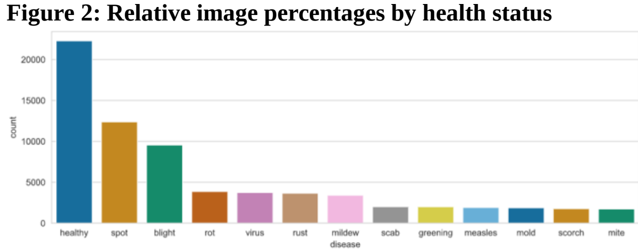


Figure 3: Number of images by disease

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

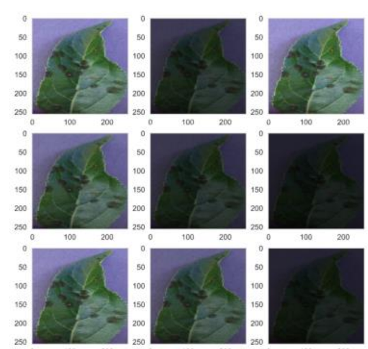


Figure 4: Data Augmentation Visualization for Random Brightness

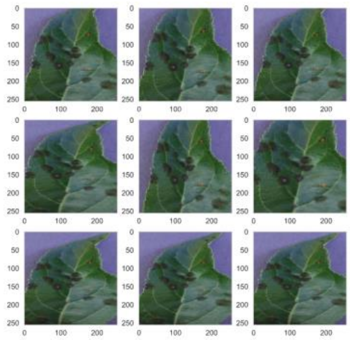


Figure 5: Data Augmentation Visualization for Random Zoom

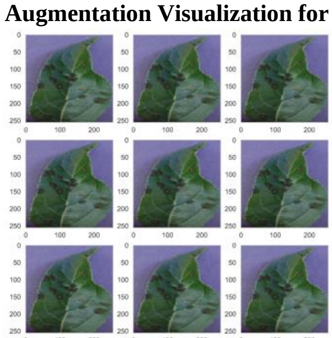


Figure 6: Data Augmentation Visualization for Random Shear

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

	Pre-Prosessing Steps	Validation Accuracy
0	rescale=1./255	0.91
1	rescale=1./255, shear_range=0.2	0.94
2	rescale=1./255, zoom_range=0.2	0.92
3	rescale=1./255, shear_range=0.2, zoom_range=0.2	0.92
4	rescale=1./255, shear_range=0.2, zoom_range=0.2, brightness_range=[0.2,1.0]	0.90

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.

	Pre-Prosessing Steps	Validation Accuracy
1	rescale=1./255, shear_range=0.2	0.94

3.5 Model Evaluation & Selection

We will be evaluating a number of pre-trained models with transfer learning as well as several models that we define.

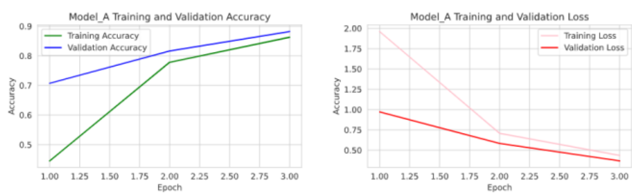
3.5.1 Custom Defined Models

3.5.1.1 Custom Defined Model A

The first iteration of the custom defined model had the following layers

```
def get_model_A():
    classifier = Sequential()
    classifier.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Flatten())
    classifier.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
    classifier.add(Dense(38,activation='softmax'))

    opt = SGD(lr=0.001, momentum=0.9)
    classifier.compile(optimizer=opt, loss='categorical_crossentropy', metrics=['accuracy'])
```



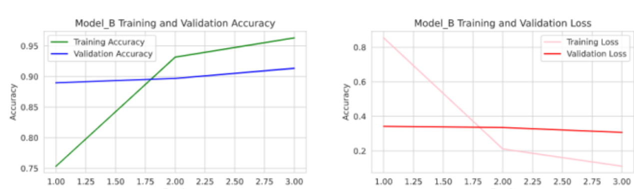
3.5.1.2 Custom Defined Model B

For the second iteration, the optimizer was changed from Stochastic Gradient Descent with a learning rate of 0.001 and momentum of 0.9 to ‘adam’.

```
def get_model_B():
    classifier = Sequential()
    classifier.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Flatten())
    classifier.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
    classifier.add(Dense(38,activation='softmax'))

    classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    return classifier
```



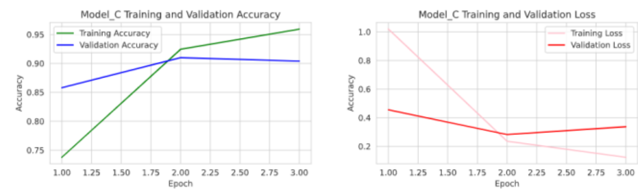
3.5.1.3 Custom Defined Model C

For the third iteration, the filter for the third 2D Convolutional Layer was updated from 128 to 256.

```
def get_model_C():
    classifier = Sequential()
    classifier.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Flatten())
    classifier.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
    classifier.add(Dense(38,activation='softmax'))

    classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    return classifier
```



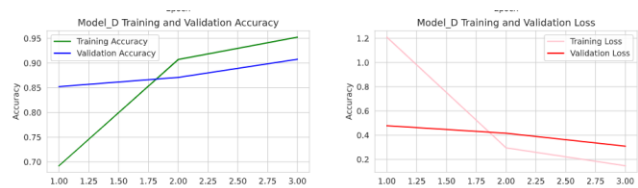
3.5.1.4 Custom Defined Model D

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

```
def get_model_D():
    classifier = Sequential()
    classifier.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Dropout(0.2))
    classifier.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform'))
    classifier.add(MaxPooling2D((2, 2)))
    classifier.add(Flatten())
    classifier.add(Dense(128, activation='relu', kernel_initializer='he_uniform'))
    classifier.add(Dense(38,activation='softmax'))

    classifier.compile(optimizer='adam', loss='categorical_crossentropy', metrics=['accuracy'])

    return classifier
```



	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	Model_A	0.86	0.43	0.88	0.37
1	Model_B	0.96	0.11	0.91	0.31
2	Model_C	0.96	0.12	0.90	0.34
3	Model_D	0.95	0.15	0.91	0.31

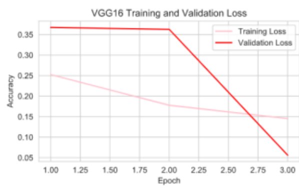
The Model_D iteration gave the best accuracy for the custom defined classifiers and will be used for further evaluation.

3.5.2 Benchmarks for Pre-Trained Models with Transfer Learning

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

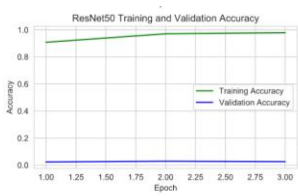
3.5.2.1 VGG16 Base Model with Transfer Learning

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg16 (Model)                (None, 7, 7, 512)         14714688
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense (Dense)                (None, 38)                953382
=================================================================
Total params: 15,668,070
Trainable params: 953,382
Non-trainable params: 14,714,688
_________________________________________________________________
```



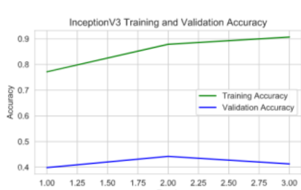
3.5.2.2 ResNet50 Base Model with Transfer Learning

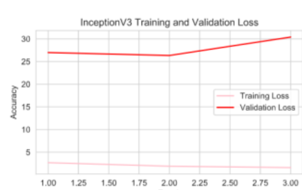
```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
resnet50 (Model)             (None, 7, 7, 2048)        23587712
_________________________________________________________________
flatten_1 (Flatten)          (None, 100352)            0
_________________________________________________________________
dense_1 (Dense)              (None, 38)                3813414
=================================================================
Total params: 27,401,126
Trainable params: 3,813,414
Non-trainable params: 23,587,712
_________________________________________________________________
```



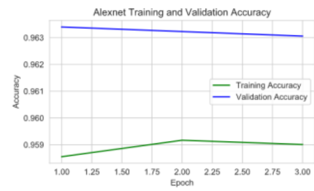
3.5.2.3 InceptionV3 Base Model with Transfer Learning

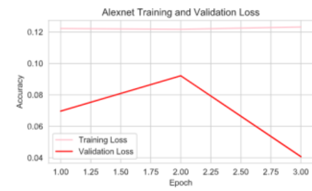
```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
inception_v3 (Model)         (None, 5, 5, 2048)        21802784
_________________________________________________________________
flatten_2 (Flatten)          (None, 51200)             0
_________________________________________________________________
dense_2 (Dense)              (None, 38)                1945638
=================================================================
Total params: 23,748,422
Trainable params: 1,945,638
Non-trainable params: 21,802,784
_________________________________________________________________
```





3.5.2.4 Alexnet Base Model with Transfer Learning





	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	VGG16	0.96	0.15	0.93	0.06
1	ResNet50	0.98	1.06	0.03	135.88
2	InceptionV3	0.91	1.60	0.41	30.40
3	Alexnet	0.96	0.12	0.96	0.04

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

3.5.4 Hyperparameter Tuning

VGG16

Best Params: {'learning_rate': 0.0001, 'epochs': 3, 'batch_size': 32, 'activation': 'softmax'}

Alexnet

Custom Defined Model D

Best Params: {'learning_rate': 0.001, 'epochs': 10, 'batch_size': 32, 'activation': 'softmax'}

	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	VGG16	0.96	0.15	0.93	0.06
1	ResNet50	0.98	1.06	0.03	135.88
2	InceptionV3	0.91	1.60	0.41	30.40
3	Alexnet	0.96	0.12	0.96	0.04
4	Model_01	0.88	2.67	0.65	20.44
5	VGG16 HP Tuned	0.95	0.19	0.94	0.27

Best Paramaters: {'learning_rate': 0.0001, 'epochs': 3, 'batch_size': 32, 'activation': 'softmax'}

3.5.3 Final Model and Predictions

To Complete

4 Discussion

To Complete

5 Conclusion

To Complete

ACM Reference format:

Jerry Khidaroo, Paul Doucet. 2020. New Plant Disease Detection. In *Proceedings of CSML1020 (Winter'2020). York University, Toronto, ON, Canada, 5 pages*.

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]Acknowledgments https://elc.polyu.edu.hk/FYP/html/ack.htm#:~:text=A%20page%20of%20acknowledgements%20is,in%20carrying%20out%20the%20research.
[9] Jason Brownlee 2019, Introduction to Python Deep Learning with Keras https://machinelearningmastery.com/introduction-python-deep-learning-library-keras/