Project: New Plant Disease Detection

CSML1020 – Group 3: Jerry Khidaroo, Paul Doucet
Instructor: Karthik Kuber, PhD, MS

school of continuing studies | YORK U UNIVERSITÉ UNIVERSITY

CSML1020 – Final Project

- Introduction
- Dataset: New Plant Diseases Dataset
- Data Preparation
- Data Exploration
- Data Augmentation
- Custom Defined Models
- Pre-Trained Models with Transfer Learning
- Hyperparameter Tuning
- Results
- Model Selection
- Example Predictions
- Next Steps
- References

Introduction

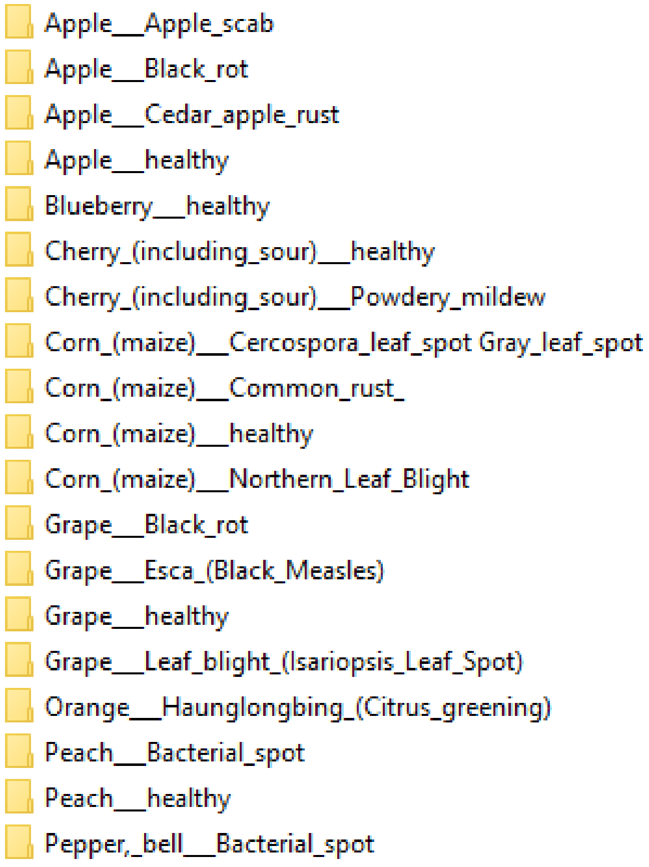
Project Definition:

- 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

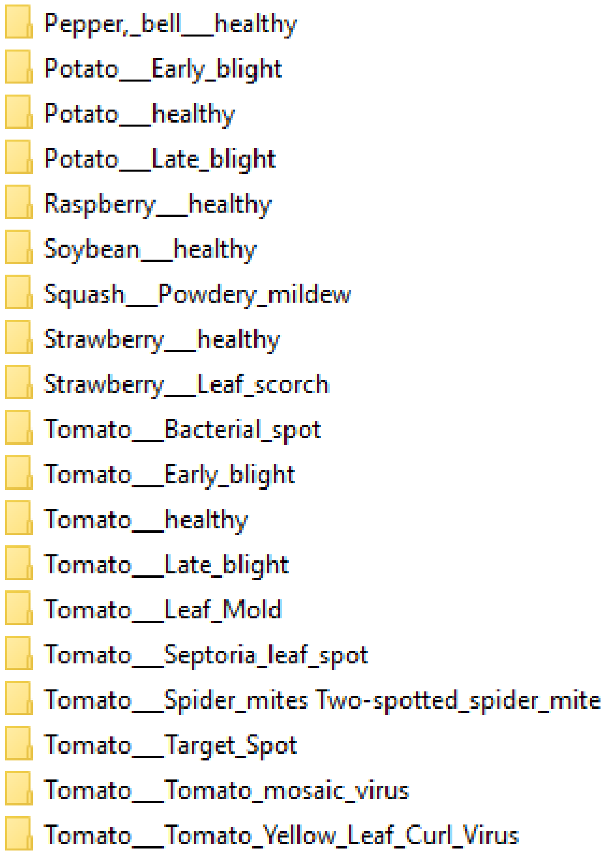
Dataset: New Plant Diseases Dataset from Kaggle

- The data can be obtained from the following location:
  - https://www.kaggle.com/vipoooool/new-plant-diseases-dataset

38 Categories of Images



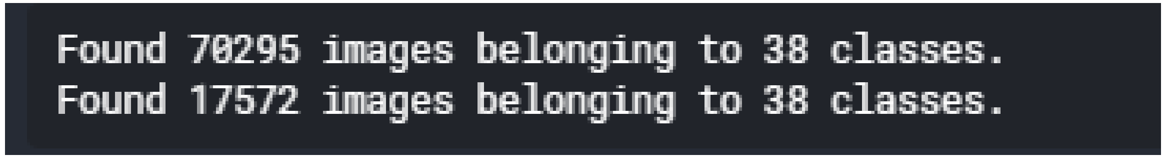
Apple___Apple_scab
Apple___Black_rot
Apple___Cedar_apple_rust
Apple___healthy
Blueberry___healthy
Cherry_(including_sour)___healthy
Cherry_(including_sour)___Powdery_mildew
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot
Corn_(maize)___Common_rust_
Corn_(maize)___healthy
Corn_(maize)___Northern_Leaf_Blight
Grape___Black_rot
Grape___Esca_(Black_Measles)
Grape___healthy
Grape___Leaf_blight_(Isariopsis_Leaf_Spot)
Orange___Haunglongbing_(Citrus_greening)
Peach___Bacterial_spot
Peach___healthy
Pepper,_bell___Bacterial_spot



Pepper,_bell___healthy
Potato___Early_blight
Potato___healthy
Potato___Late_blight
Raspberry___healthy
Soybean___healthy
Squash___Powdery_mildew
Strawberry___healthy
Strawberry___Leaf_scorch
Tomato___Bacterial_spot
Tomato___Early_blight
Tomato___healthy
Tomato___Late_blight
Tomato___Leaf_Mold
Tomato___Septoria_leaf_spot
Tomato___Spider_mites Two-spotted_spider_mite
Tomato___Target_Spot
Tomato___Tomato_mosaic_virus
Tomato___Tomato_Yellow_Leaf_Curl_Virus

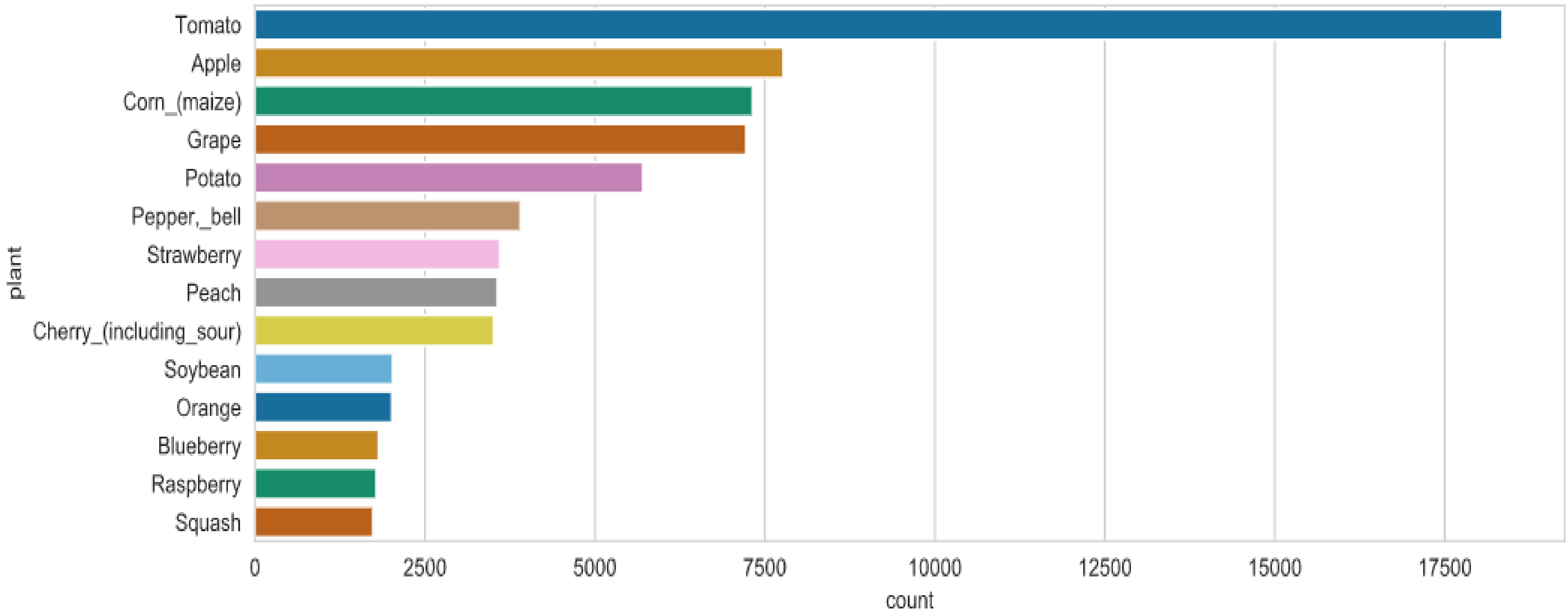
Data Preparation

- The image dataset was well prepared and did not require any modification
- The images were divided into training and validation folders:
    - 70295 training images
    - 17572 validation images
- The folder names were parsed to obtain some data exploration data



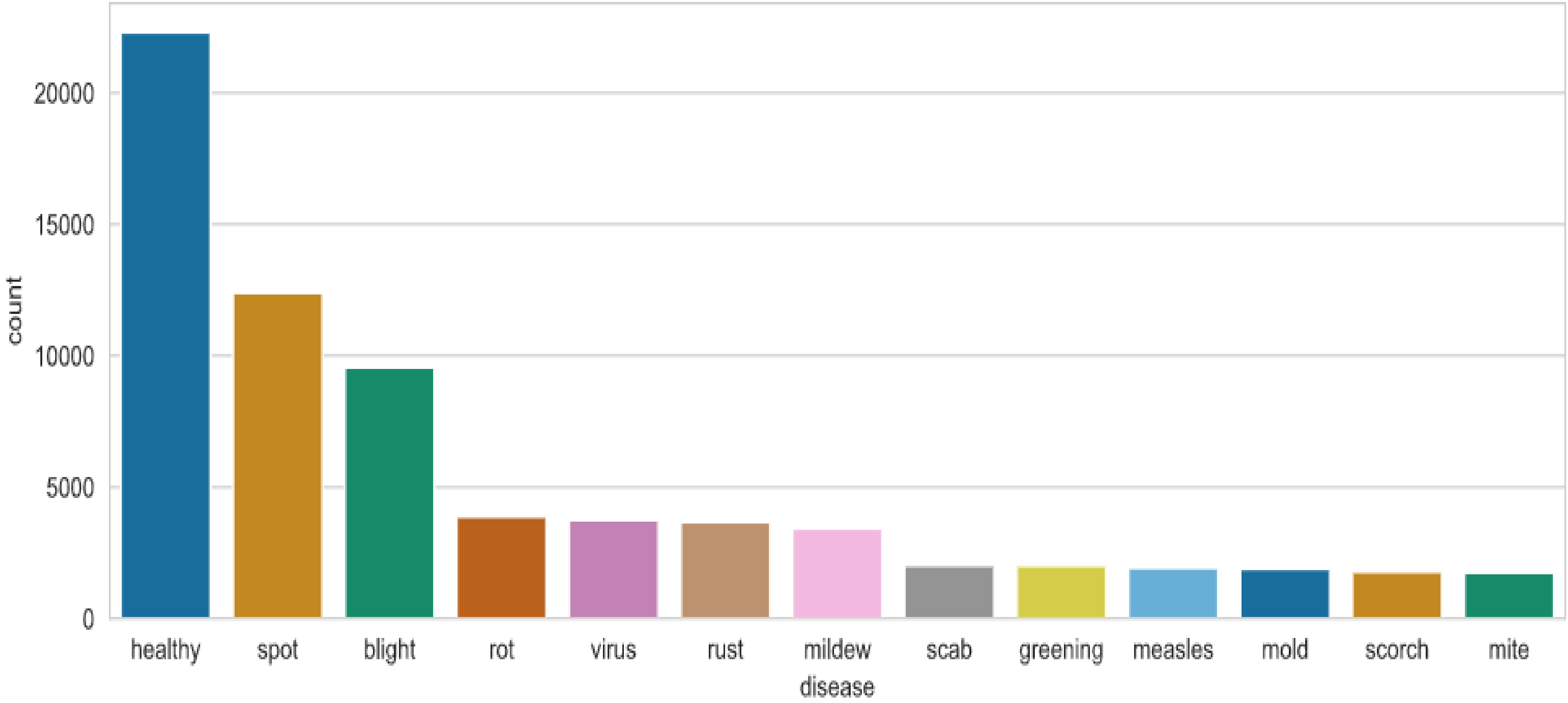
```
Found 70295 images belonging to 38 classes.
Found 17572 images belonging to 38 classes.
```

Data Exploration



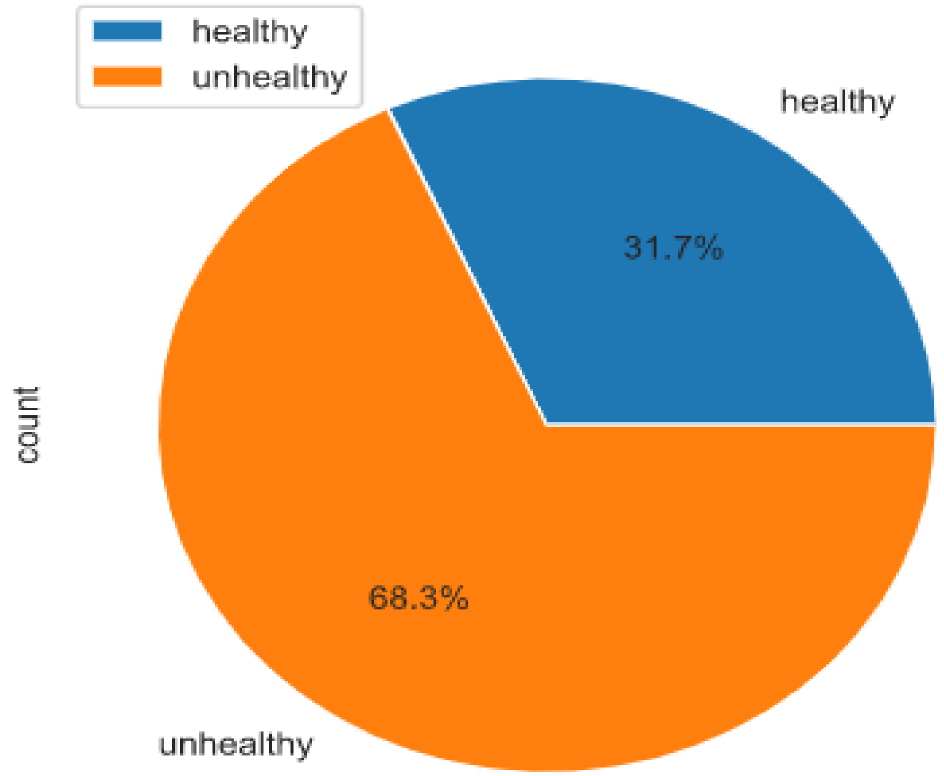
Number of Images by Plant

Data Exploration



Number of Images by Disease

Data Exploration



Relative Percentages by Health Status

Data Preprocessing

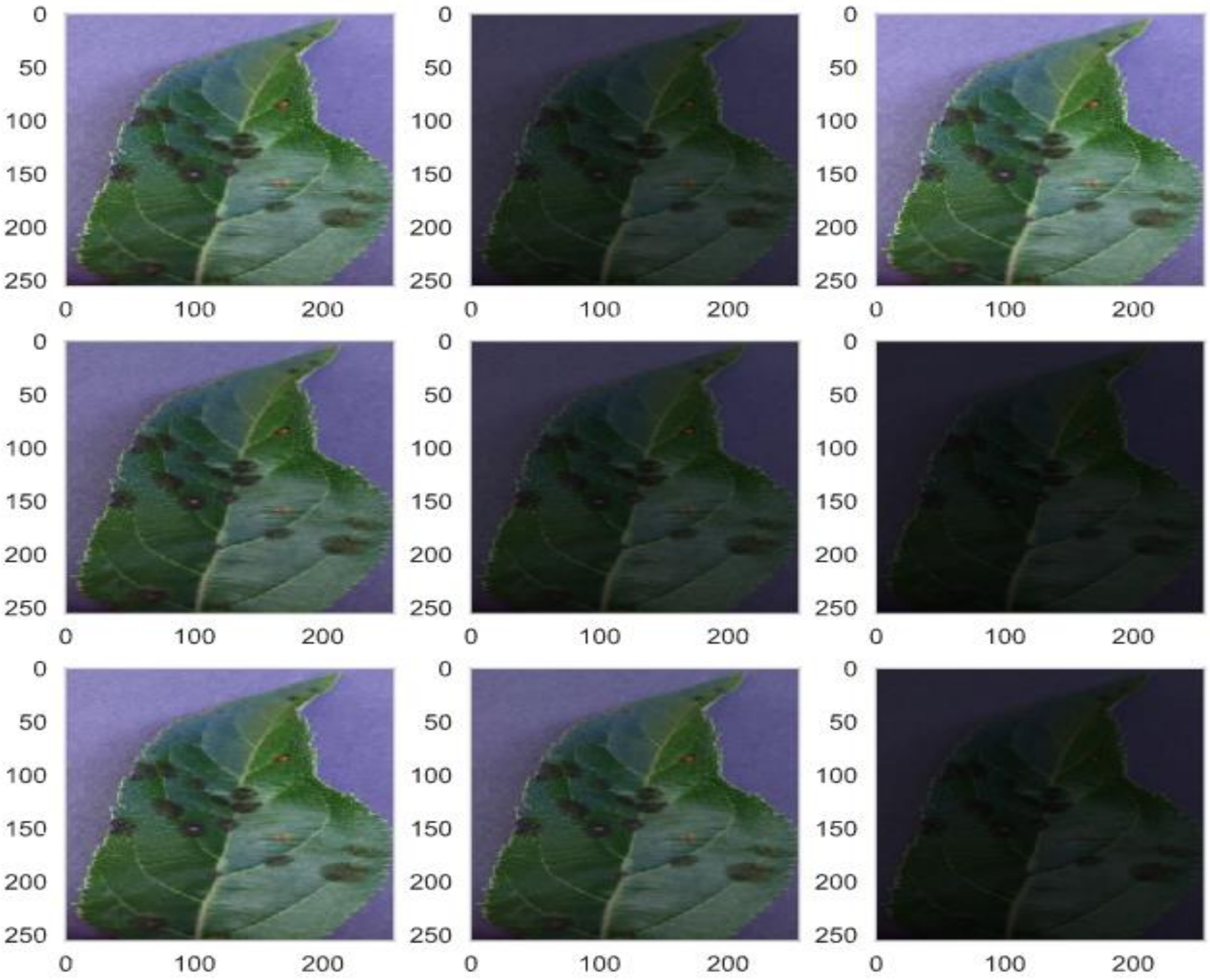
The following data preprocessing methods will be evaluated to determine the best data augmentation for our input layer:

- Random Shear
- Random Brightness
- Random Zoom

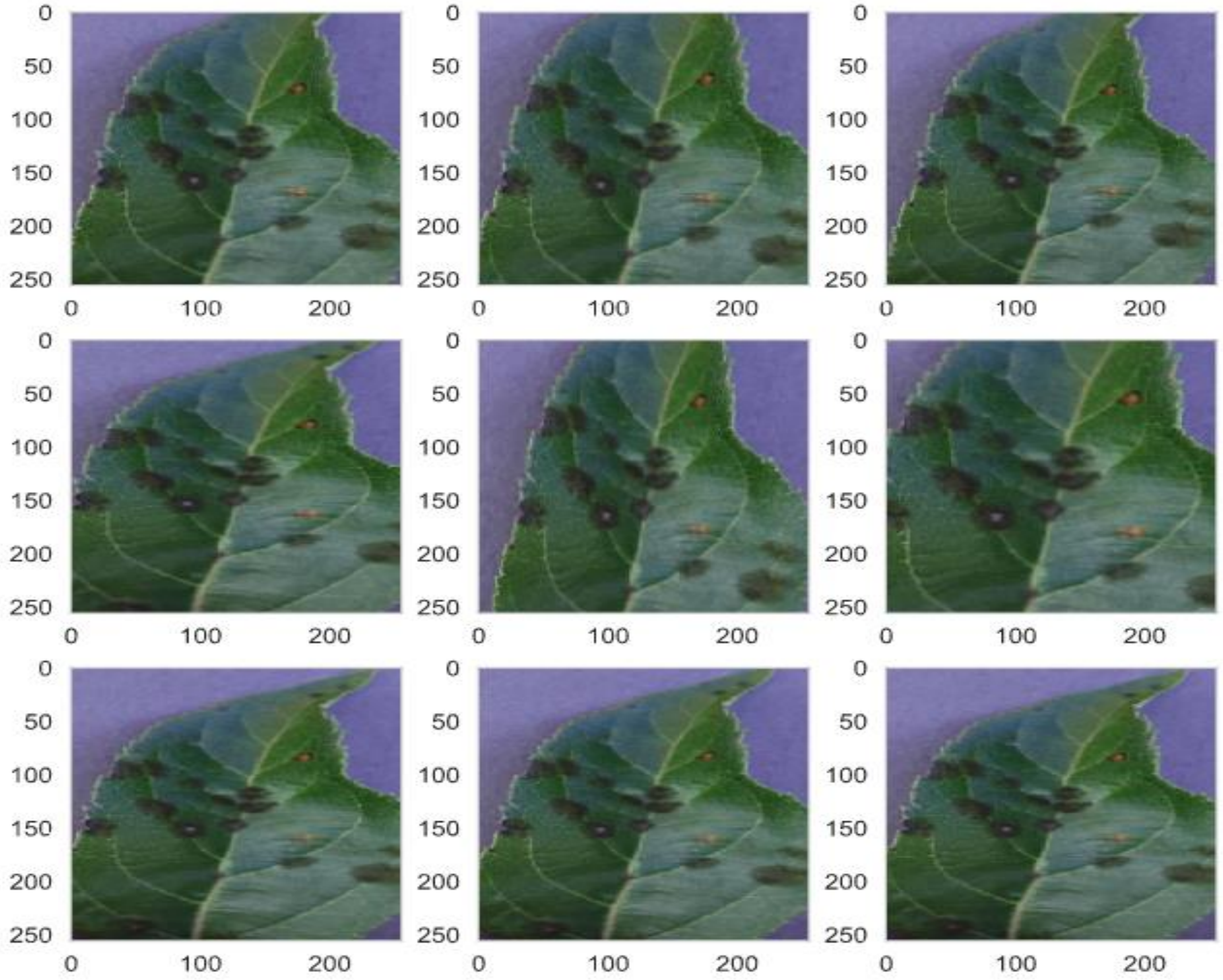
The following methods were omitted because the dataset was already pre-processed with those methods:

- Random Horizontal Shift
- Random Vertical Shift
- Random Horizontal Flip
- Random Vertical Flip

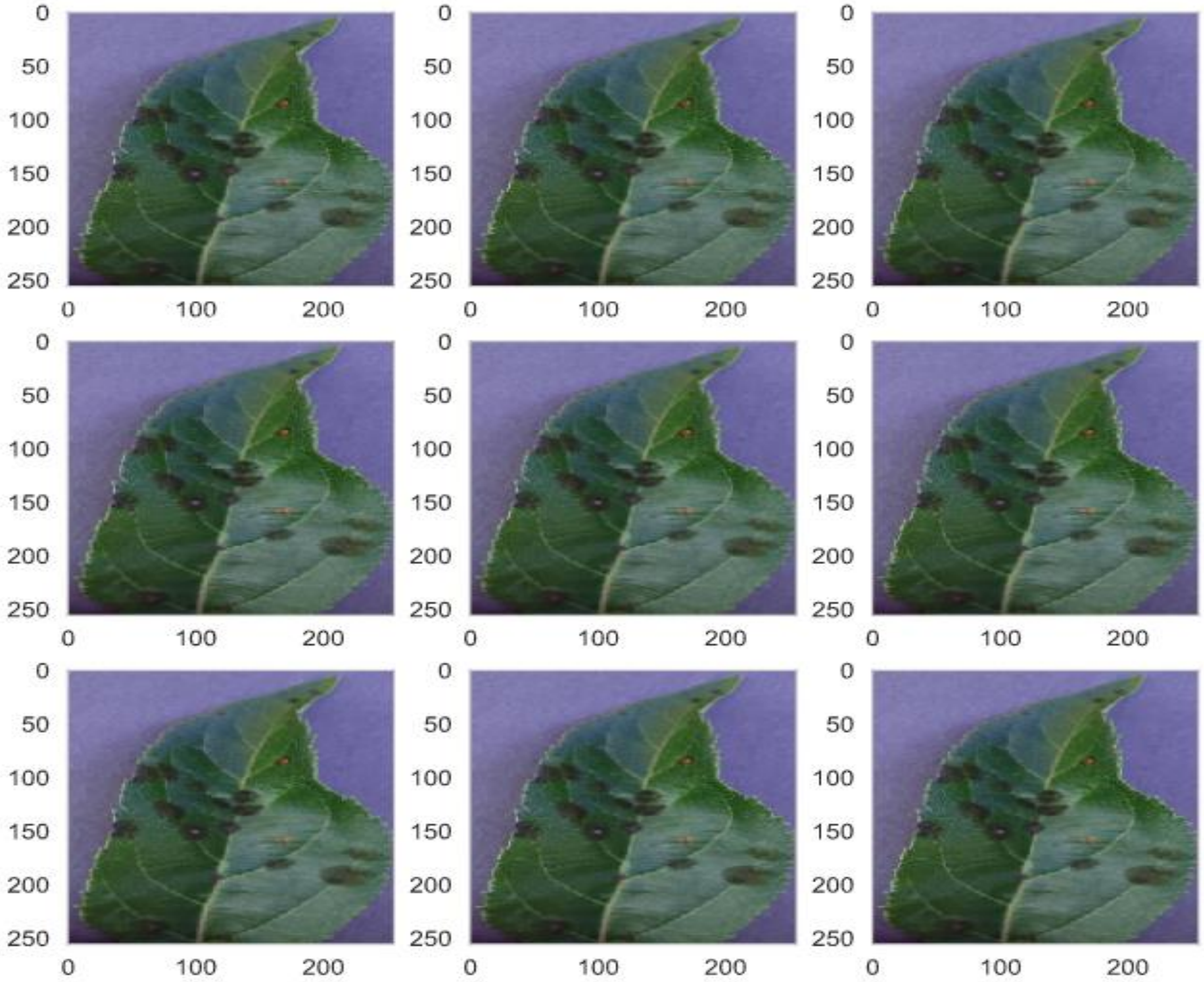
Data Augmentation Visualization For Random Brightness



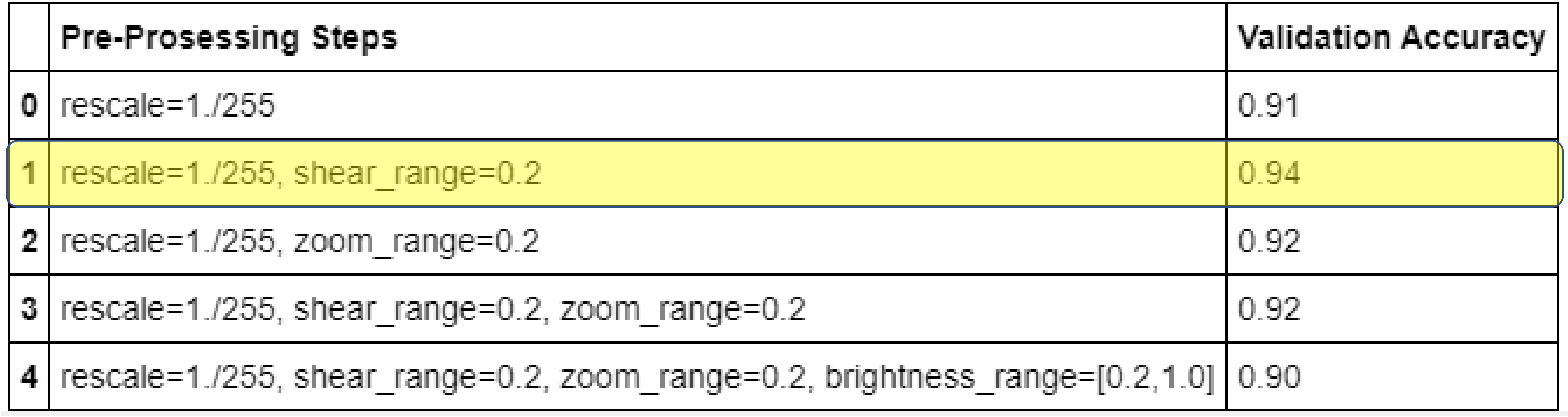
Data Augmentation Visualization For Random Zoom



Data Augmentation Visualization For Random Shear



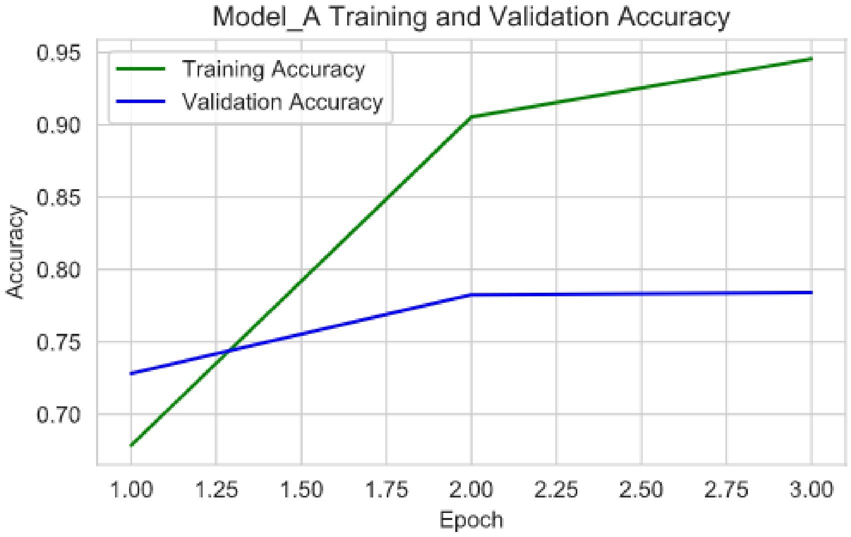
Preprocessing Results

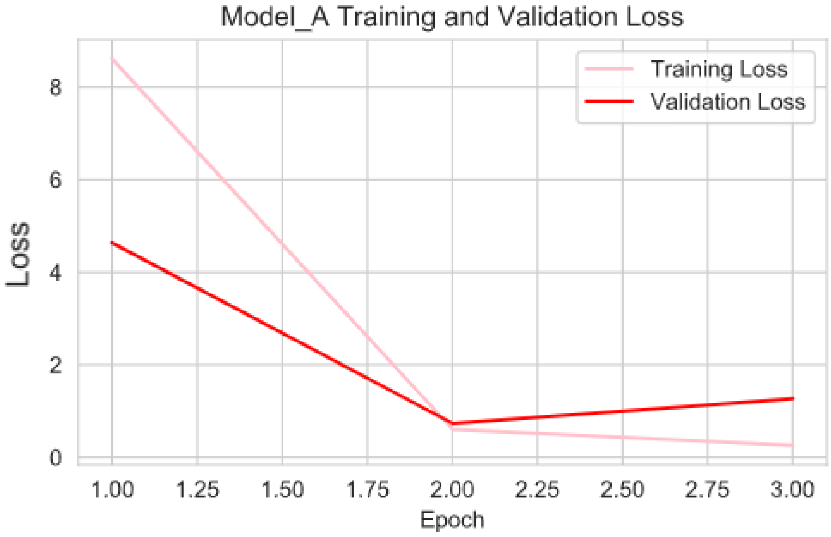


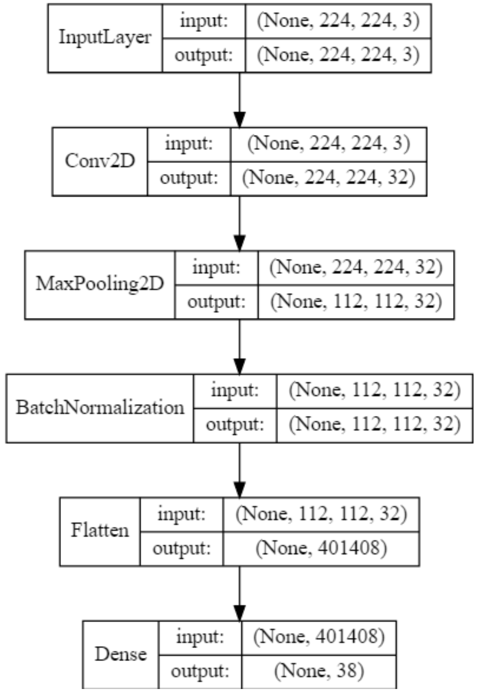
	Pre-Prosessing Steps	Validation Accuracy
0	rescale=1./255	0.91
1	rescale=1./255, shear_range=0.2	0.94
2	rescale=1./255, zoom_range=0.2	0.92
3	rescale=1./255, shear_range=0.2, zoom_range=0.2	0.92
4	rescale=1./255, shear_range=0.2, zoom_range=0.2, brightness_range=[0.2,1.0]	0.90

Selected Preprocessing Method

Model A

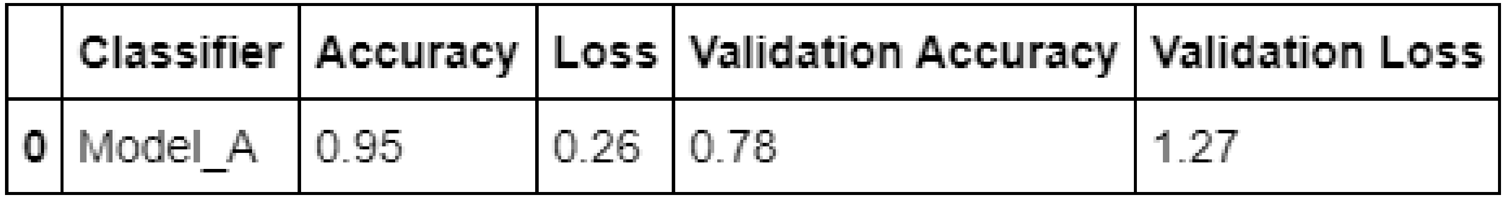

Model_A Training and Validation Accuracy


Model_A Training and Validation Loss



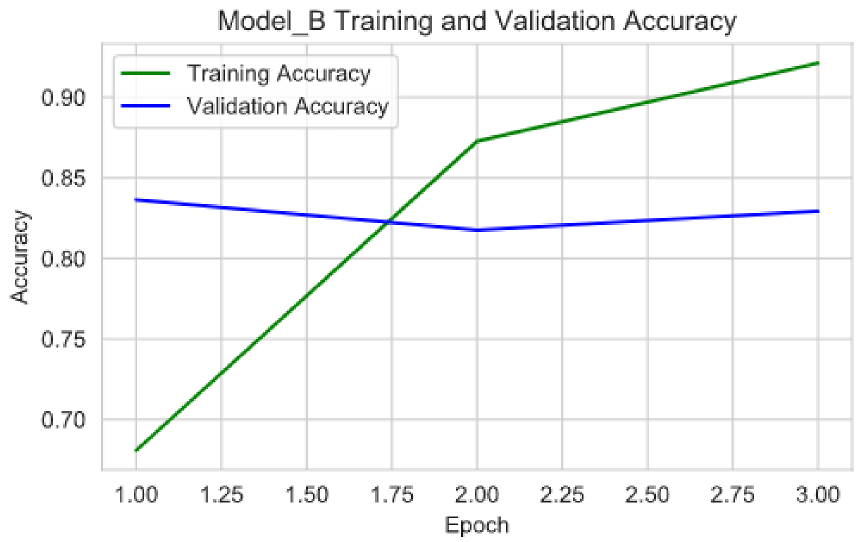
InputLayer	input:	(None, 224, 224, 3)
	output:	(None, 224, 224, 3)
Conv2D	input:	(None, 224, 224, 3)
	output:	(None, 224, 224, 32)
MaxPooling2D	input:	(None, 224, 224, 32)
	output:	(None, 112, 112, 32)
BatchNormalization	input:	(None, 112, 112, 32)
	output:	(None, 112, 112, 32)
Flatten	input:	(None, 112, 112, 32)
	output:	(None, 401408)
Dense	input:	(None, 401408)
	output:	(None, 38)

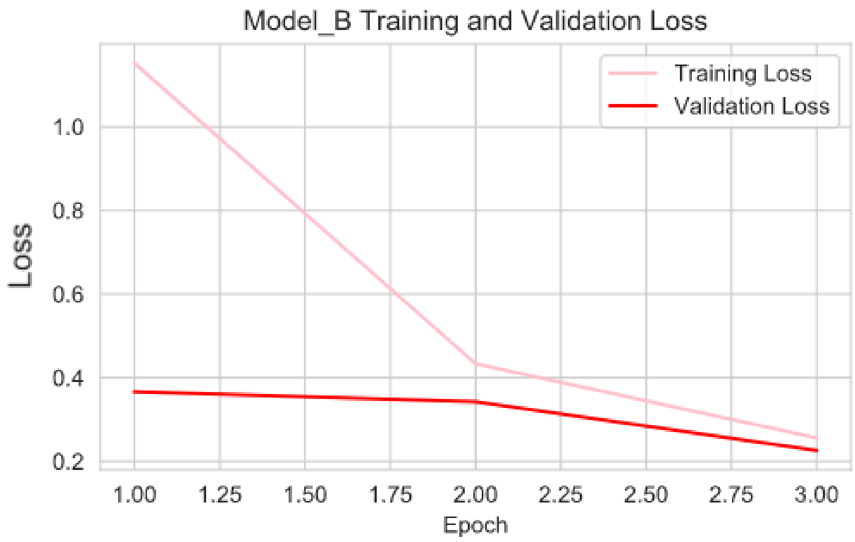
Model with one convolution layer, max pooling layer, batch normalization layer and a flatten layer.

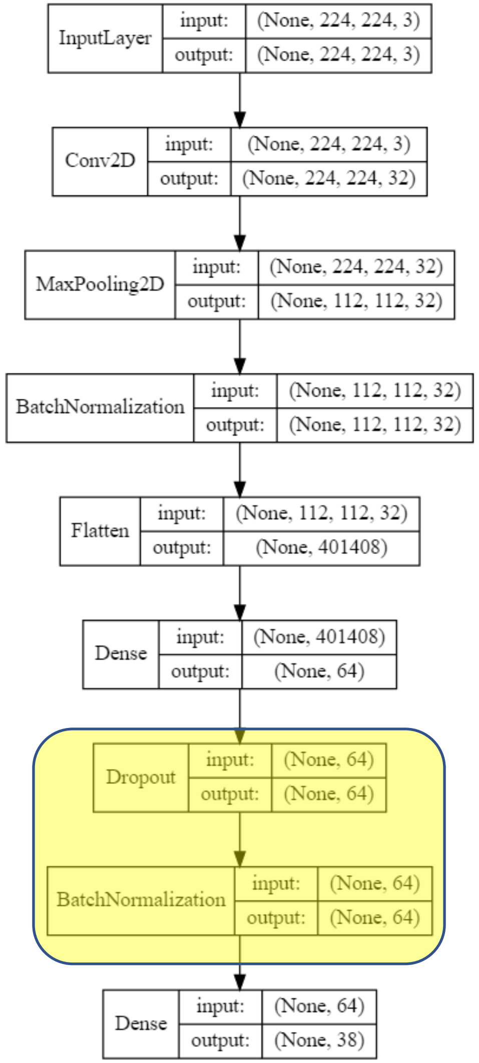


	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	Model_A	0.95	0.26	0.78	1.27

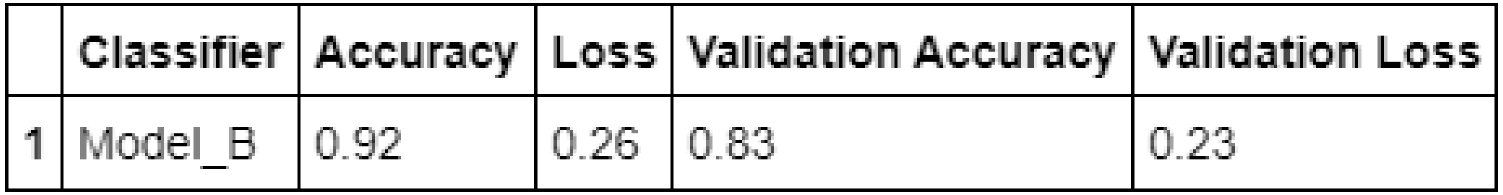
Model B





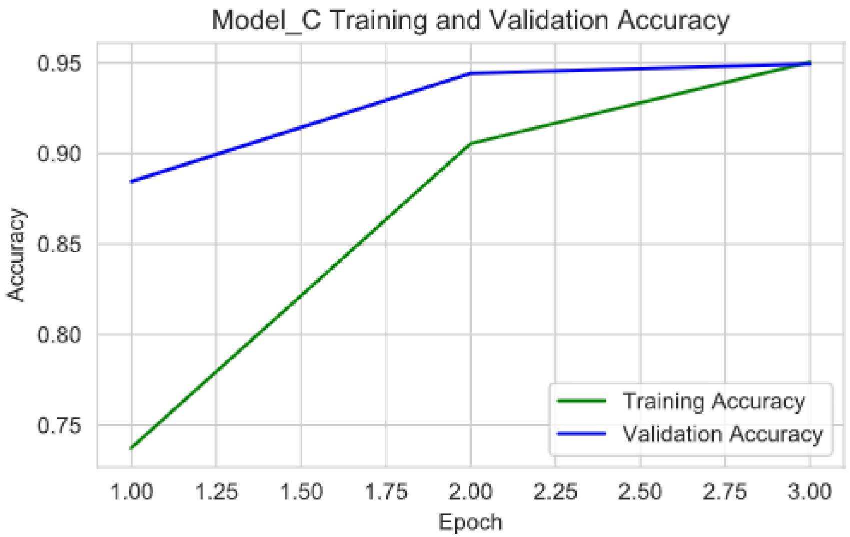


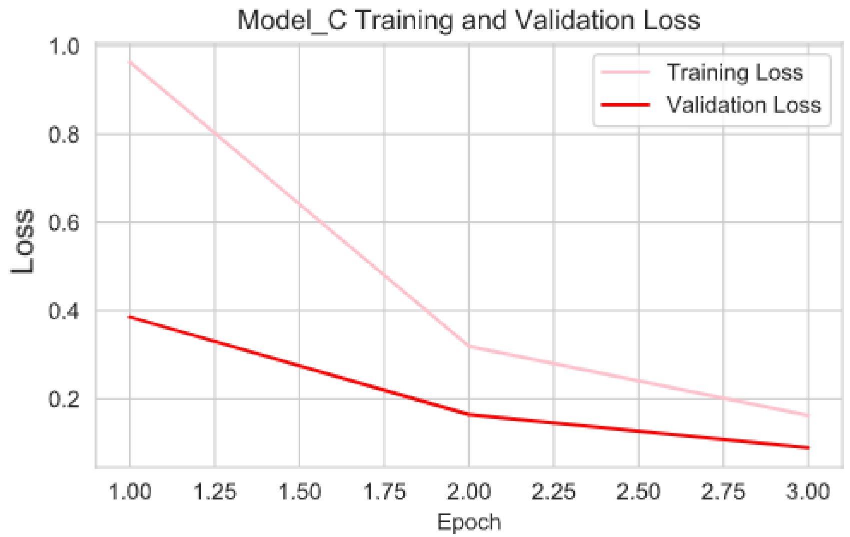
Second model, with dropout layer and batch normalization before the output layer.

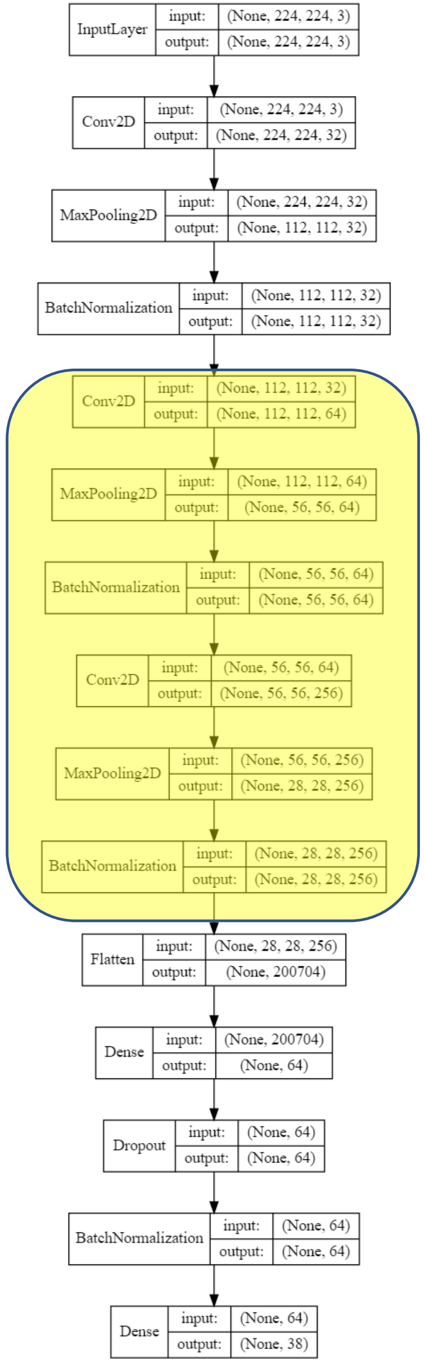


	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
1	Model_B	0.92	0.26	0.83	0.23

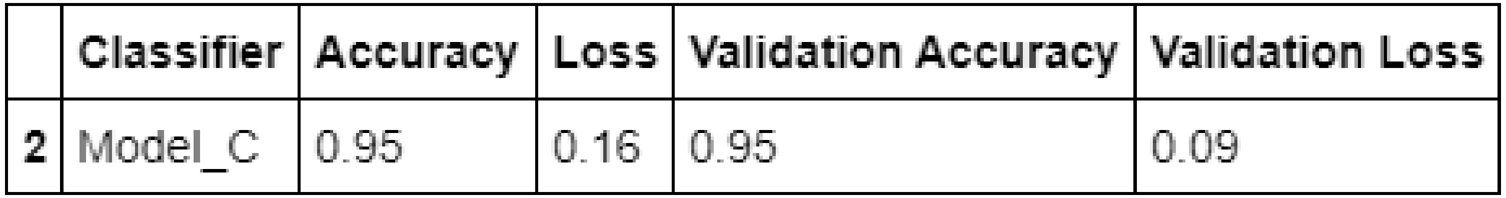
Model C





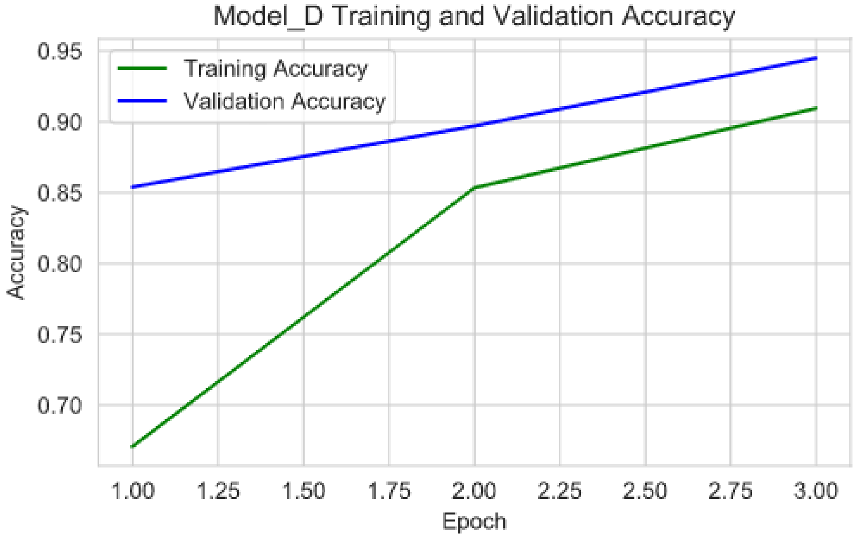

InputLayer input: (None, 224, 224, 3) output: (None, 224, 224, 3)
Conv2D input: (None, 224, 224, 3) output: (None, 224, 224, 32)
MaxPooling2D input: (None, 224, 224, 32) output: (None, 112, 112, 32)
BatchNormalization input: (None, 112, 112, 32) output: (None, 112, 112, 32)
Conv2D input: (None, 112, 112, 32) output: (None, 112, 112, 64)
MaxPooling2D input: (None, 112, 112, 64) output: (None, 56, 56, 64)
BatchNormalization input: (None, 56, 56, 64) output: (None, 56, 56, 64)
Conv2D input: (None, 56, 56, 64) output: (None, 56, 56, 256)
MaxPooling2D input: (None, 56, 56, 256) output: (None, 28, 28, 256)
BatchNormalization input: (None, 28, 28, 256) output: (None, 28, 28, 256)
Flatten input: (None, 28, 28, 256) output: (None, 200704)
Dense input: (None, 200704) output: (None, 64)
Dropout input: (None, 64) output: (None, 64)
BatchNormalization input: (None, 64) output: (None, 64)
Dense input: (None, 64) output: (None, 38)

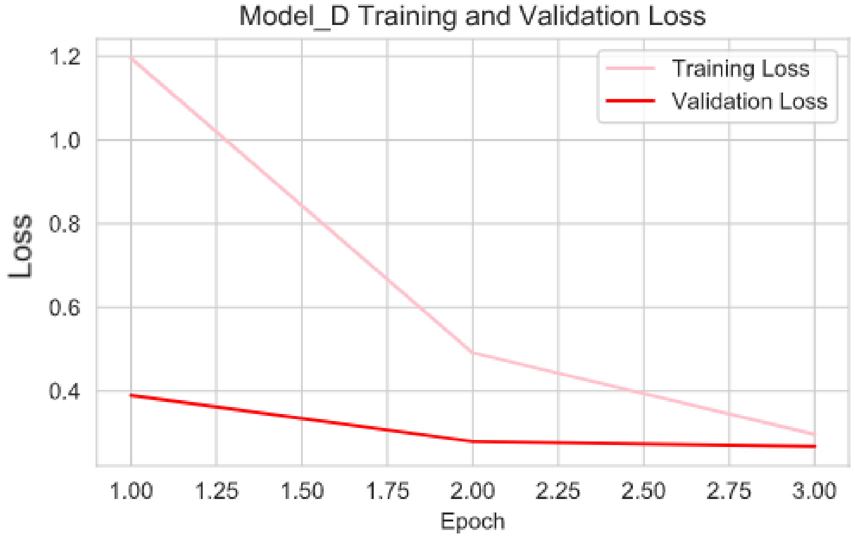
Third model, added two additional convolutional layers, with max pooling and batch normalization.



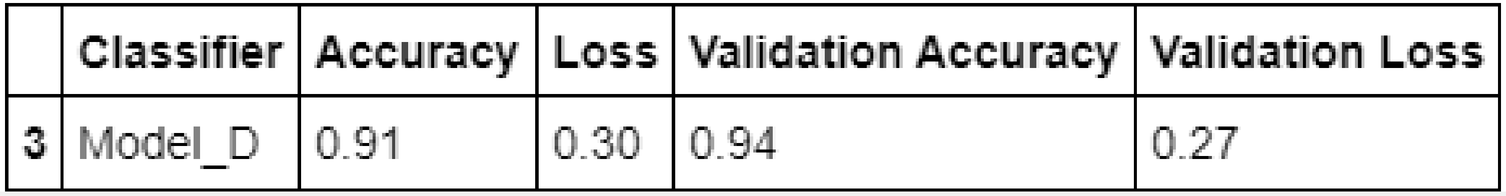
	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
2	Model_C	0.95	0.16	0.95	0.09

Model D

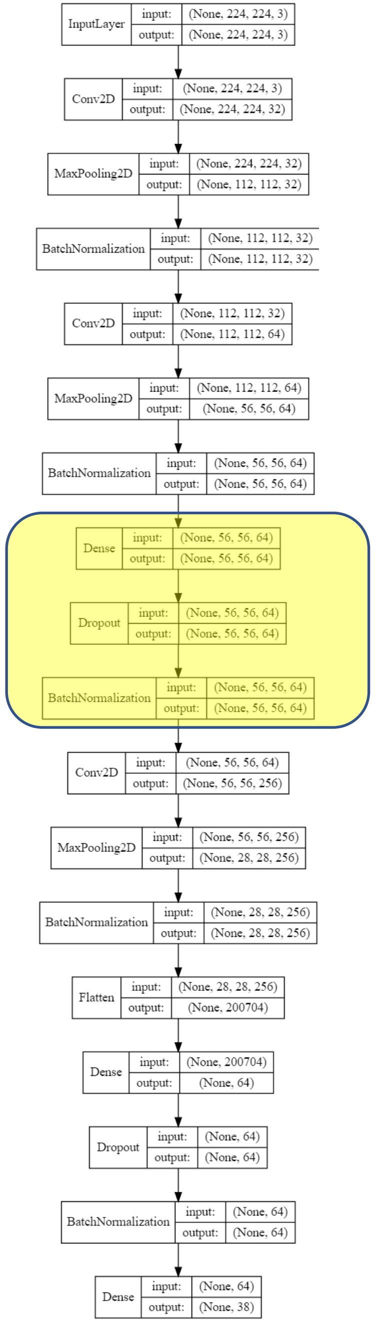




Fourth model, added dense layer, dropout layer and a batch normalization layer in between the second and third convolutional layers.



	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
3	Model_D	0.91	0.30	0.94	0.27



InputLayer input: (None, 224, 224, 3) output: (None, 224, 224, 3)

Conv2D input: (None, 224, 224, 3) output: (None, 224, 224, 32)

MaxPooling2D input: (None, 224, 224, 32) output: (None, 112, 112, 32)

BatchNormalization input: (None, 112, 112, 32) output: (None, 112, 112, 32)

Conv2D input: (None, 112, 112, 32) output: (None, 112, 112, 64)

MaxPooling2D input: (None, 112, 112, 64) output: (None, 56, 56, 64)

BatchNormalization input: (None, 56, 56, 64) output: (None, 56, 56, 64)

Dense input: (None, 56, 56, 64) output: (None, 56, 56, 64)

Dropout input: (None, 56, 56, 64) output: (None, 56, 56, 64)

BatchNormalization input: (None, 56, 56, 64) output: (None, 56, 56, 64)

Conv2D input: (None, 56, 56, 64) output: (None, 56, 56, 256)

MaxPooling2D input: (None, 56, 56, 256) output: (None, 28, 28, 256)

BatchNormalization input: (None, 28, 28, 256) output: (None, 28, 28, 256)

Flatten input: (None, 28, 28, 256) output: (None, 200704)

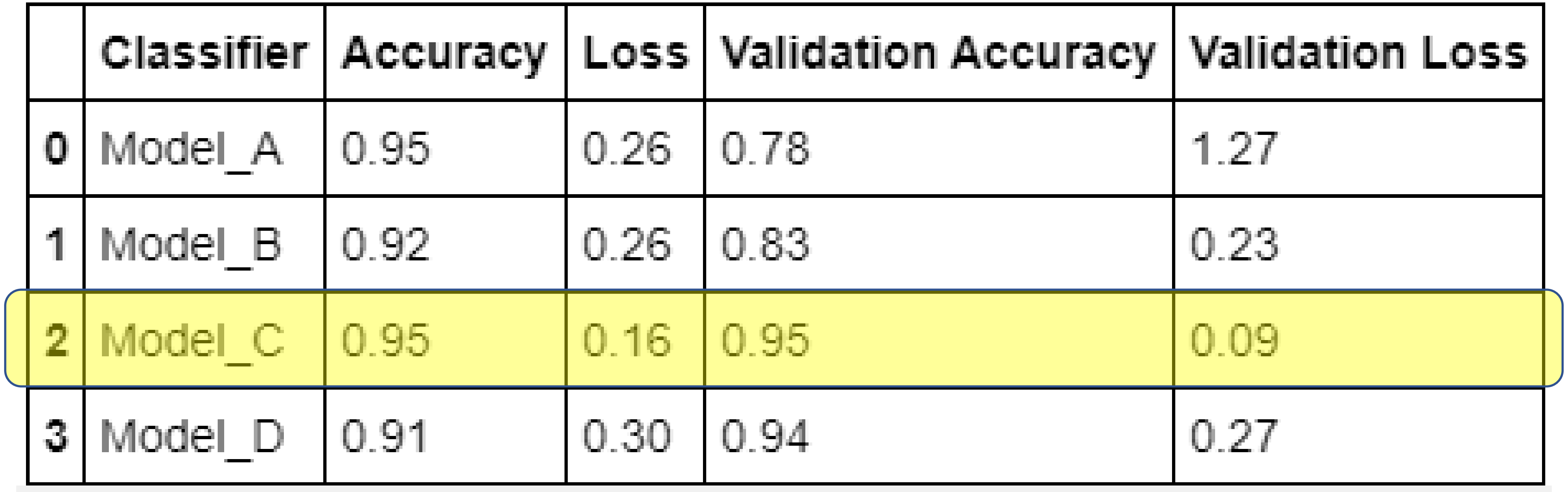
Dense input: (None, 200704) output: (None, 64)

Dropout input: (None, 64) output: (None, 64)

BatchNormalization input: (None, 64) output: (None, 64)

Dense input: (None, 64) output: (None, 38)

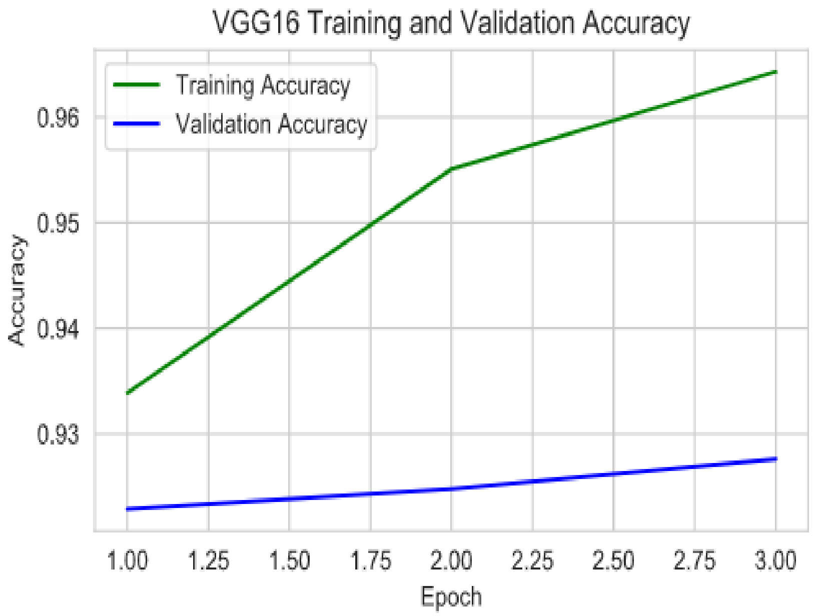
Benchmarks for Custom Defined Models



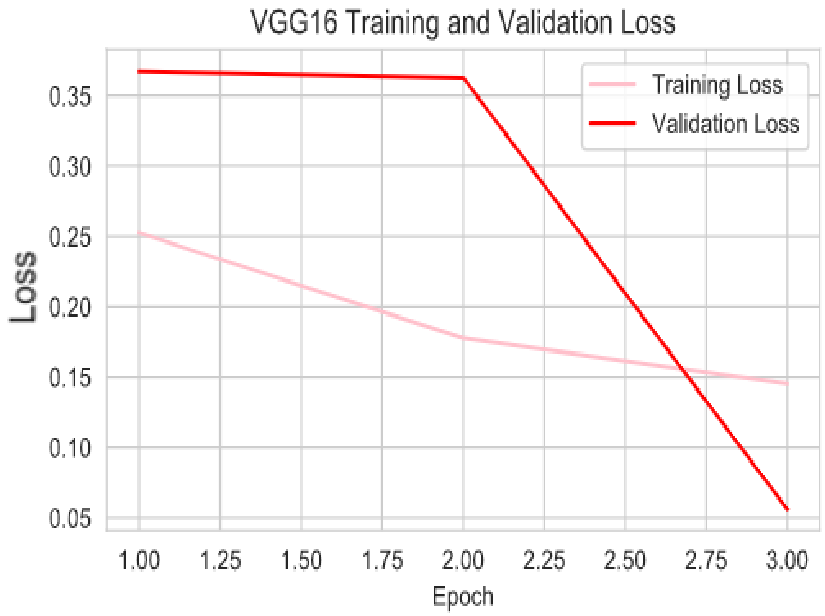
	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	Model_A	0.95	0.26	0.78	1.27
1	Model_B	0.92	0.26	0.83	0.23
2	Model_C	0.95	0.16	0.95	0.09
3	Model_D	0.91	0.30	0.94	0.27

Model_C Selected

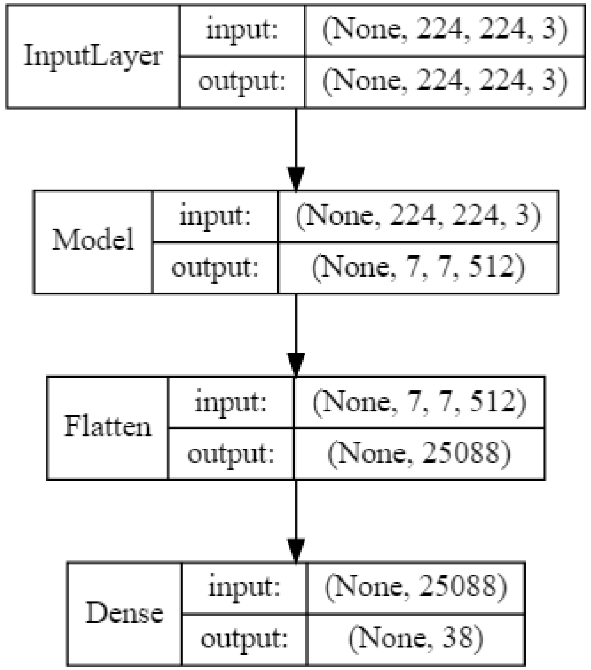
VGG16



VGG16 Training and Validation Accuracy

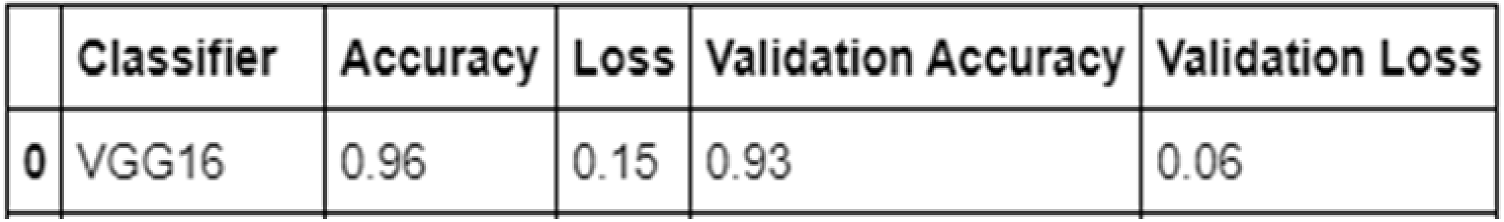


VGG16 Training and Validation Loss



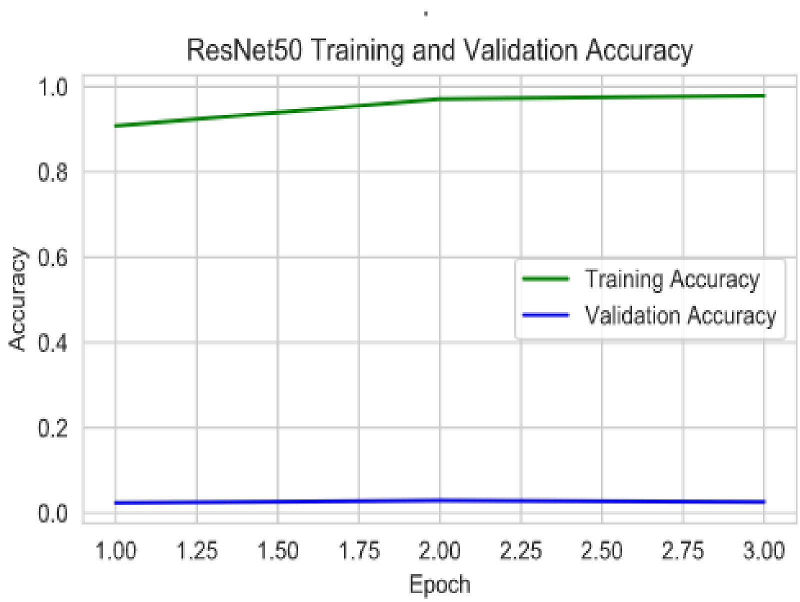
InputLayer	input:	(None, 224, 224, 3)
	output:	(None, 224, 224, 3)
Model	input:	(None, 224, 224, 3)
	output:	(None, 7, 7, 512)
Flatten	input:	(None, 7, 7, 512)
	output:	(None, 25088)
Dense	input:	(None, 25088)
	output:	(None, 38)

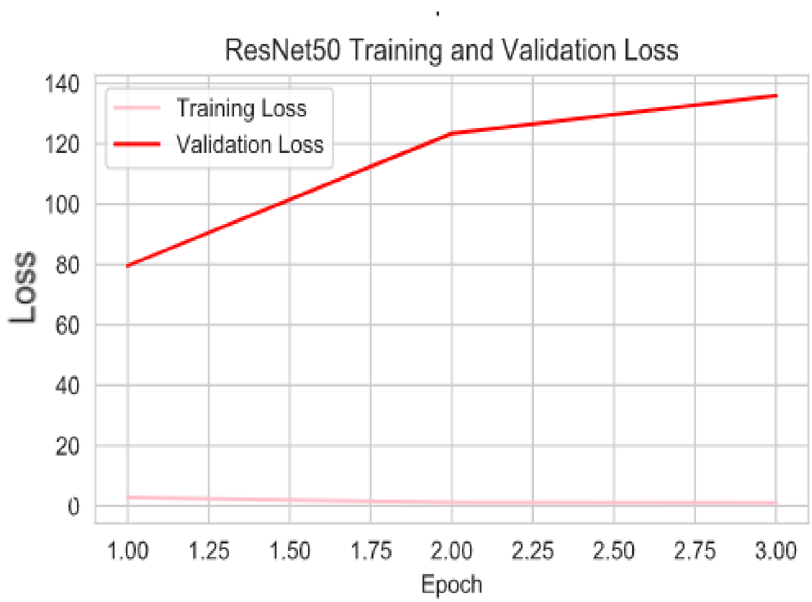
Pre-trained model using VGG16 with default weights

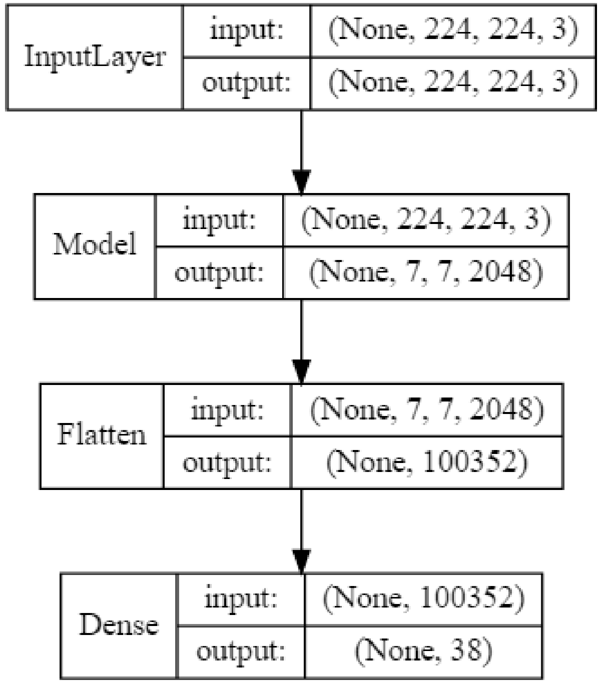


	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	VGG16	0.96	0.15	0.93	0.06

ResNet50


ResNet50 Training and Validation Accuracy


ResNet50 Training and Validation Loss


InputLayer | input: (None, 224, 224, 3) | output: (None, 224, 224, 3)

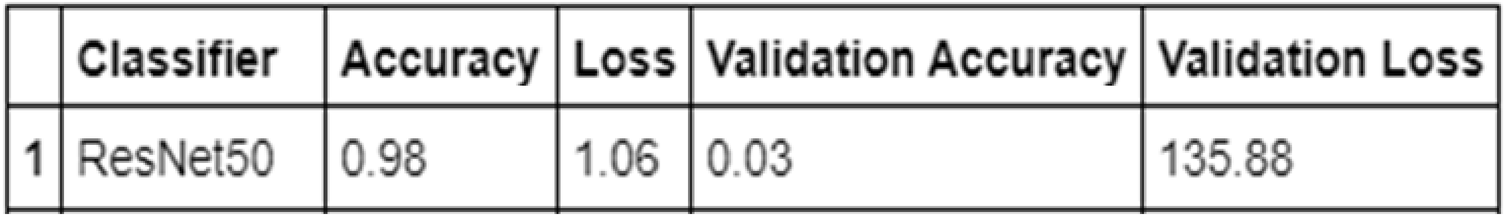
Model | input: (None, 224, 224, 3) | output: (None, 7, 7, 2048)

Flatten | input: (None, 7, 7, 2048) | output: (None, 100352)

Dense | input: (None, 100352) | output: (None, 38)

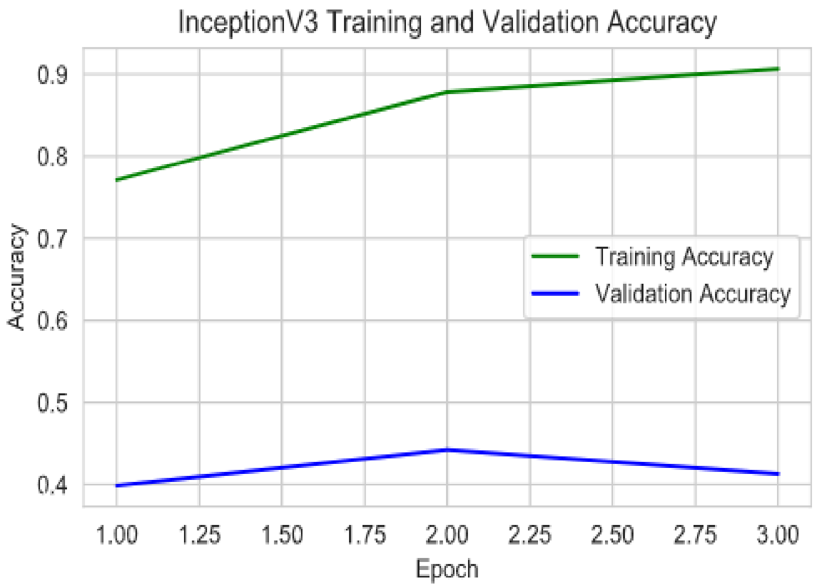
Pre-trained model using ResNet50 with default weights

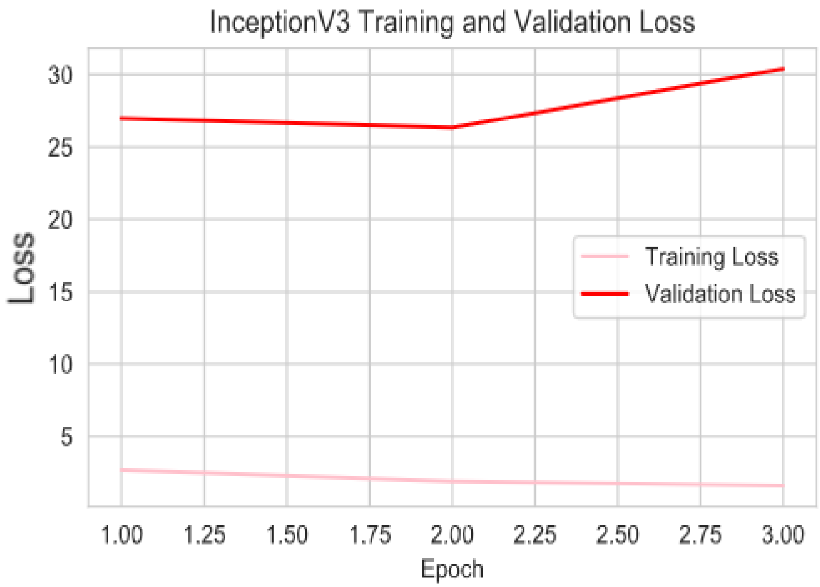
Note: Model is severely over-fitting

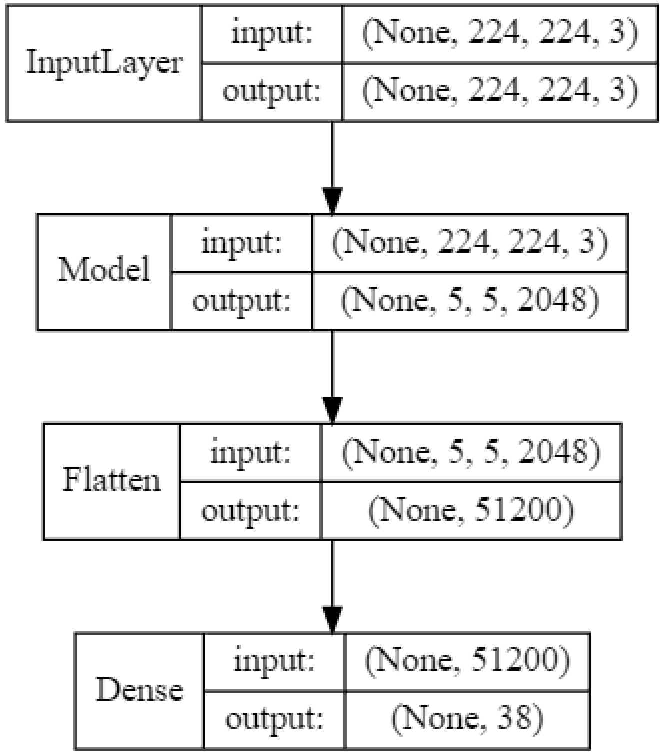


	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
1	ResNet50	0.98	1.06	0.03	135.88

InceptionV3







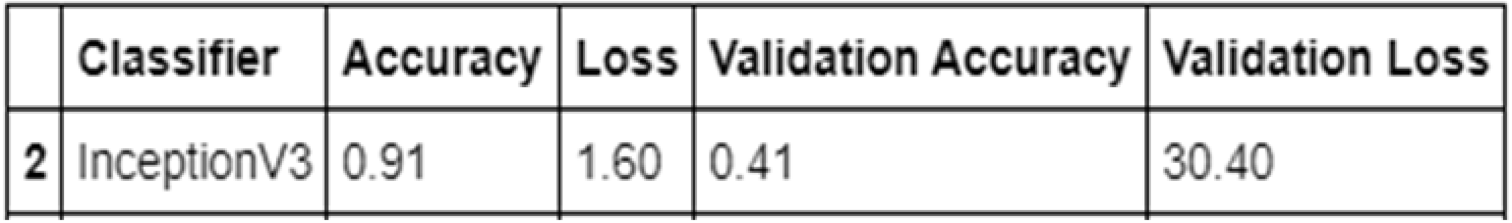
InputLayer | input: | (None, 224, 224, 3)
output: | (None, 224, 224, 3)

Model | input: | (None, 224, 224, 3)
output: | (None, 5, 5, 2048)

Flatten | input: | (None, 5, 5, 2048)
output: | (None, 51200)

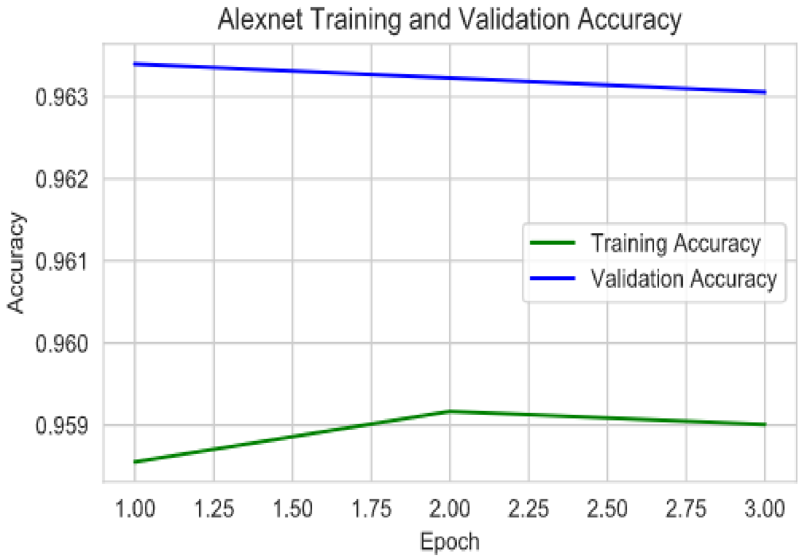
Dense | input: | (None, 51200)
output: | (None, 38)

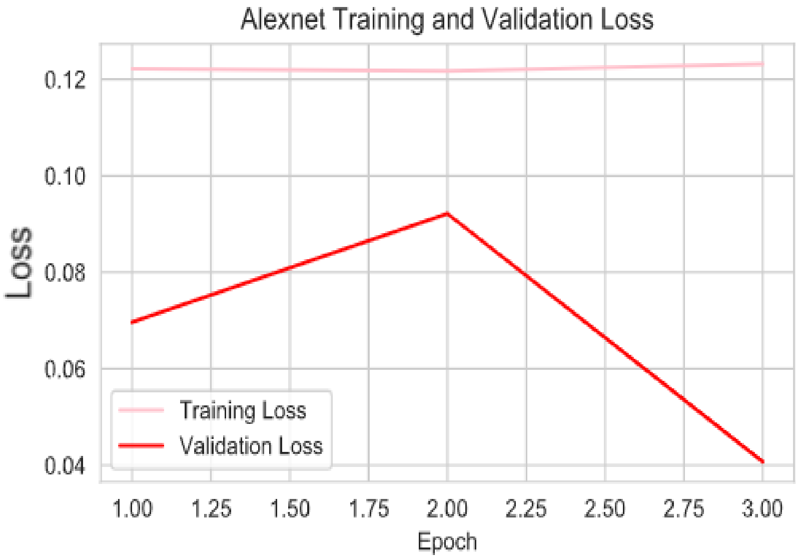
Pre-trained model using InceptionV3 with default weights

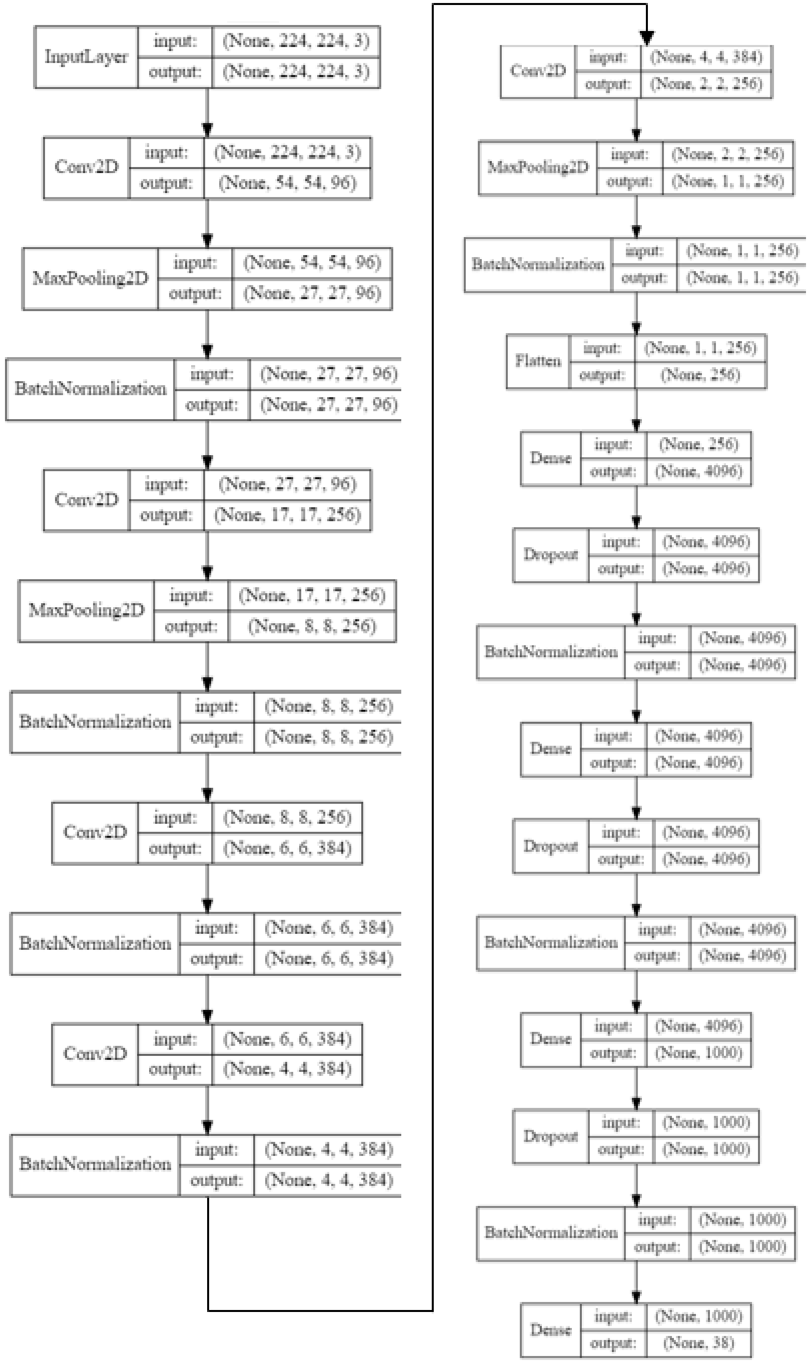


	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
2	InceptionV3	0.91	1.60	0.41	30.40

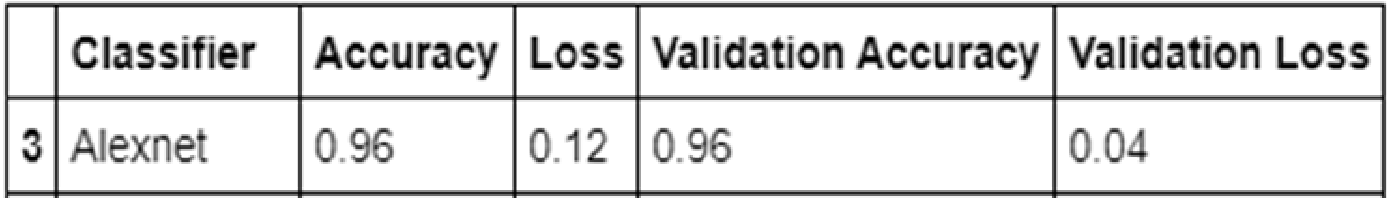
AlexNet





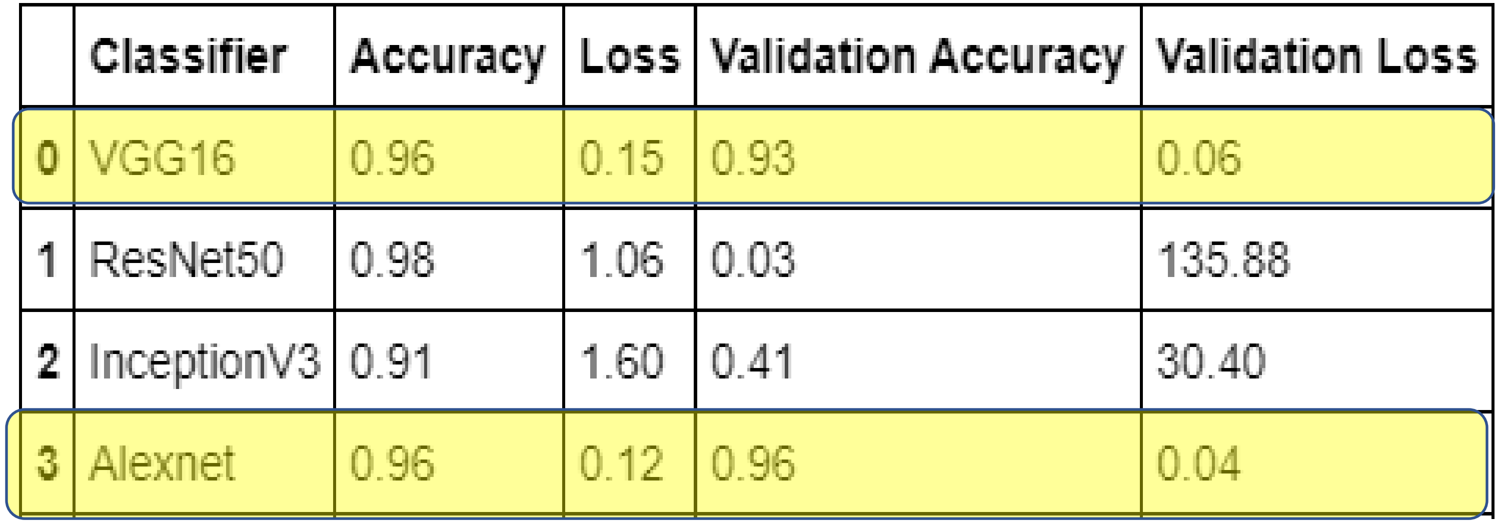

InputLayer | input: | (None, 224, 224, 3)
output: | (None, 224, 224, 3)
Conv2D | input: | (None, 224, 224, 3)
output: | (None, 54, 54, 96)
MaxPooling2D | input: | (None, 54, 54, 96)
output: | (None, 27, 27, 96)
BatchNormalization | input: | (None, 27, 27, 96)
output: | (None, 27, 27, 96)
Conv2D | input: | (None, 27, 27, 96)
output: | (None, 17, 17, 256)
MaxPooling2D | input: | (None, 17, 17, 256)
output: | (None, 8, 8, 256)
BatchNormalization | input: | (None, 8, 8, 256)
output: | (None, 8, 8, 256)
Conv2D | input: | (None, 8, 8, 256)
output: | (None, 6, 6, 384)
BatchNormalization | input: | (None, 6, 6, 384)
output: | (None, 6, 6, 384)
Conv2D | input: | (None, 6, 6, 384)
output: | (None, 4, 4, 384)
BatchNormalization | input: | (None, 4, 4, 384)
output: | (None, 4, 4, 384)
Conv2D | input: | (None, 4, 4, 384)
output: | (None, 2, 2, 256)
MaxPooling2D | input: | (None, 2, 2, 256)
output: | (None, 1, 1, 256)
BatchNormalization | input: | (None, 1, 1, 256)
output: | (None, 1, 1, 256)
Flatten | input: | (None, 1, 1, 256)
output: | (None, 256)
Dense | input: | (None, 256)
output: | (None, 4096)
Dropout | input: | (None, 4096)
output: | (None, 4096)
BatchNormalization | input: | (None, 4096)
output: | (None, 4096)
Dense | input: | (None, 4096)
output: | (None, 4096)
Dropout | input: | (None, 4096)
output: | (None, 4096)
BatchNormalization | input: | (None, 4096)
output: | (None, 4096)
Dense | input: | (None, 4096)
output: | (None, 1000)
Dropout | input: | (None, 1000)
output: | (None, 1000)
BatchNormalization | input: | (None, 1000)
output: | (None, 1000)
Dense | input: | (None, 1000)
output: | (None, 38)

Pre-trained model using AlexNet initialized with best weights from reference example from: https://www.kaggle.com/vipoooool/plant-diseases-classification-using-alexnet



	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
3	Alexnet	0.96	0.12	0.96	0.04

Benchmarks for Pre-Trained Models 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 and AlexNet Selected

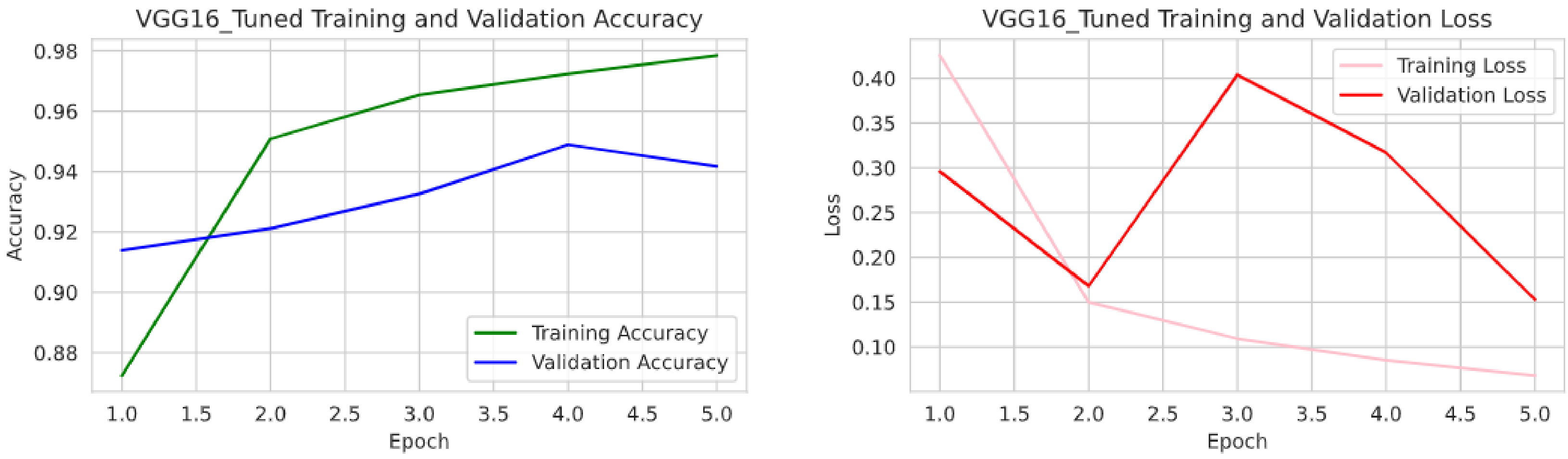
Hyperparameter Tuning with GridSearch

Model	Learning_Rate	Epochs	Batch Size	Activation
VGG16	0.001	5	64	softmax
Model_C	0.001	5	48	softmax
AlexNet	0.01	3	32	softmax

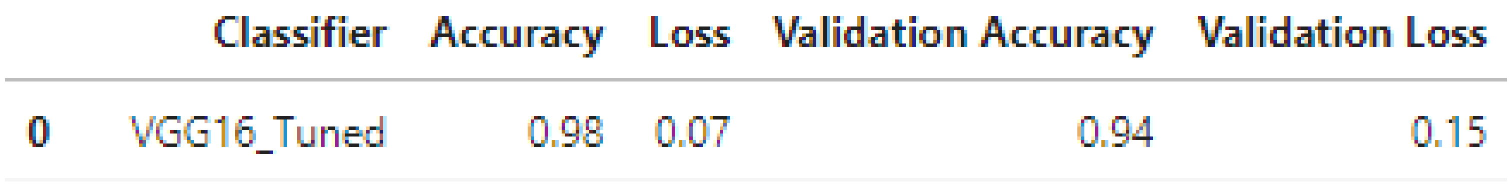
Best Parameters

VGG16 Using Best Hyper-Parameters

Model	Learning_Rate	Epochs	Batch Size	Activation
VGG16	0.001	5	64	softmax



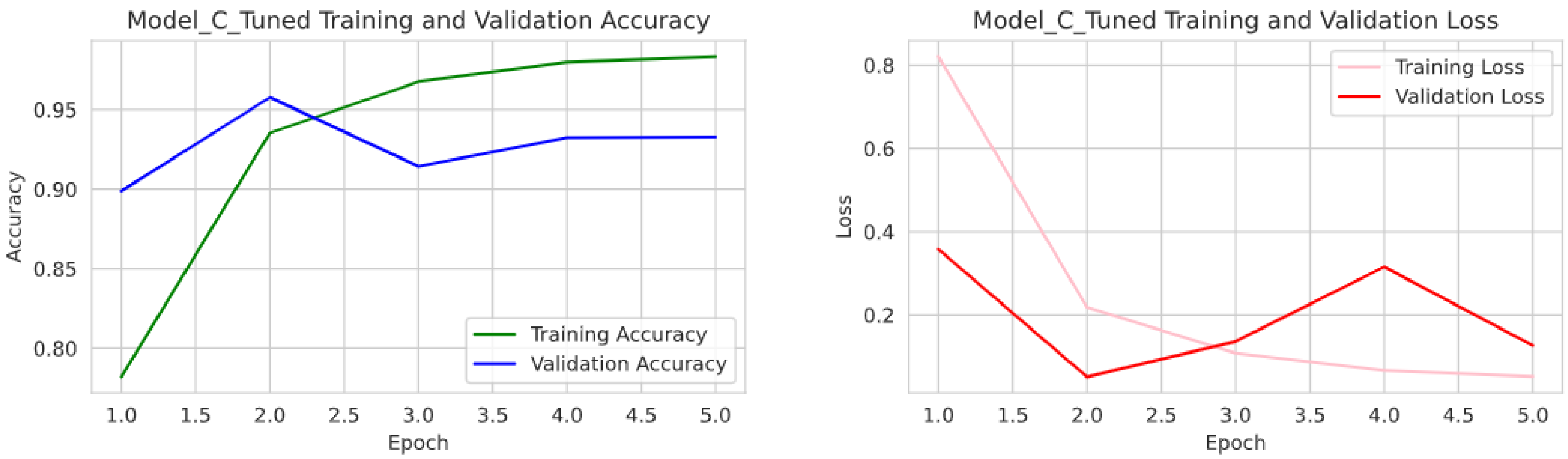
VGG16 Pre-trained model using best found hyperparameters using gridsearch



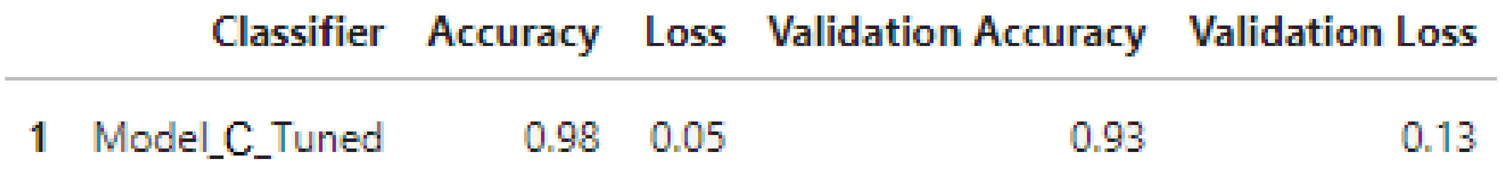
	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	VGG16_Tuned	0.98	0.07	0.94	0.15

Model_C Using Best Hyper-Parameters

Model	Learning_Rate	Epochs	Batch Size	Activation
Model_C	0.001	5	48	softmax



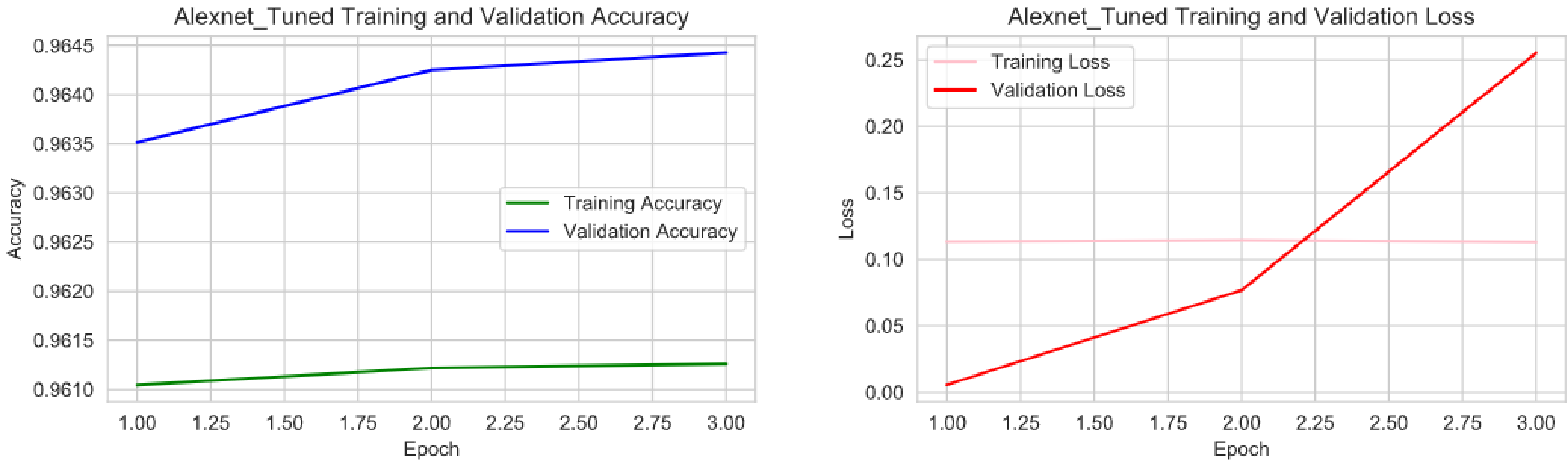
Model_C Pre-trained model using best found hyperparameters using gridsearch



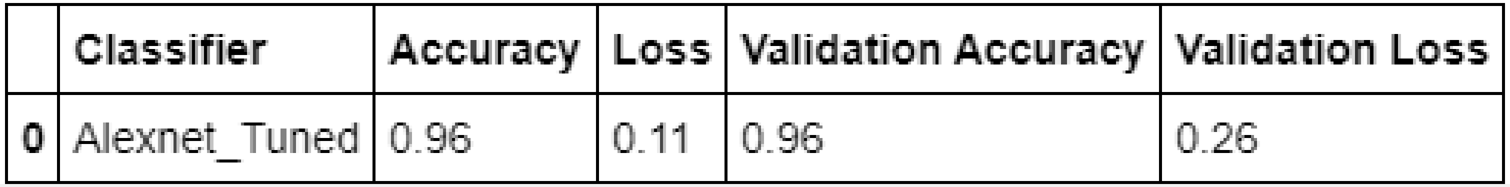
	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
1	Model_C_Tuned	0.98	0.05	0.93	0.13

AlexNet Using Best Hyper-Parameters

Model	Learning_Rate	Epochs	Batch Size	Activation
AlexNet	0.01	3	32	softmax



AlexNet Pre-trained model using best found hyperparameters using gridsearch



	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	Alexnet_Tuned	0.96	0.11	0.96	0.26

Hyperparameter Tuning Benchmarks

	Classifier	Accuracy	Loss	Validation Accuracy	Validation Loss
0	VGG16_Tuned	0.98	0.07	0.94	0.15
1	Model_C_Tuned	0.98	0.05	0.93	0.13
2	AlexNet_Tuned	0.96	0.11	0.96	0.26

Summary of Benchmark Results for all Models

	Classifier	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
1	Model_A	0.95	0.26	0.78	1.27
2	Model_B	0.92	0.26	0.83	0.23
3	**Model_C**	**0.95**	**0.16**	**0.95**	**0.09**
4	Model_D	0.91	0.30	0.94	0.27
5	VGG16	0.96	0.15	0.93	0.06
6	ResNet50	0.98	1.06	0.03	135.88
7	InceptionV3	0.91	1.60	0.41	30.40
8	**AlexNet**	**0.96**	**0.12**	**0.96**	**0.04**
9	VGG16_Tuned	0.98	0.07	0.94	0.15
10	Model_C_Tuned	0.98	0.05	0.93	0.13
11	AlexNet_Tuned	0.96	0.11	0.96	0.26

Models Selected

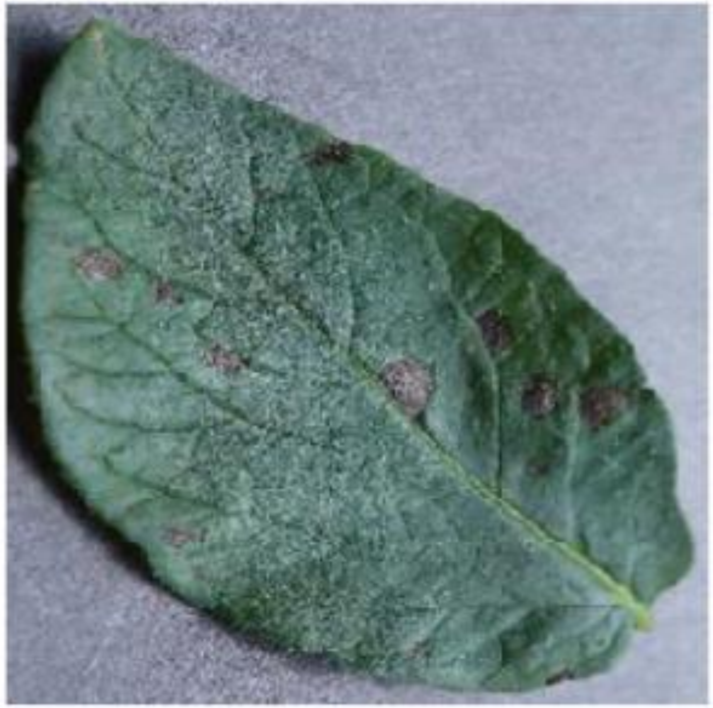
	Classifier	Training Accuracy	Training Loss	Validation Accuracy	Validation Loss
3	Model_C	0.95	0.16	0.95	0.09
8	AlexNet	0.96	0.12	0.96	0.04

Example Predictions

```
image_path = "/home/jupyter/test/test/PotatoEarlyBlight3.JPG"
```

```
Following is our prediction:
```

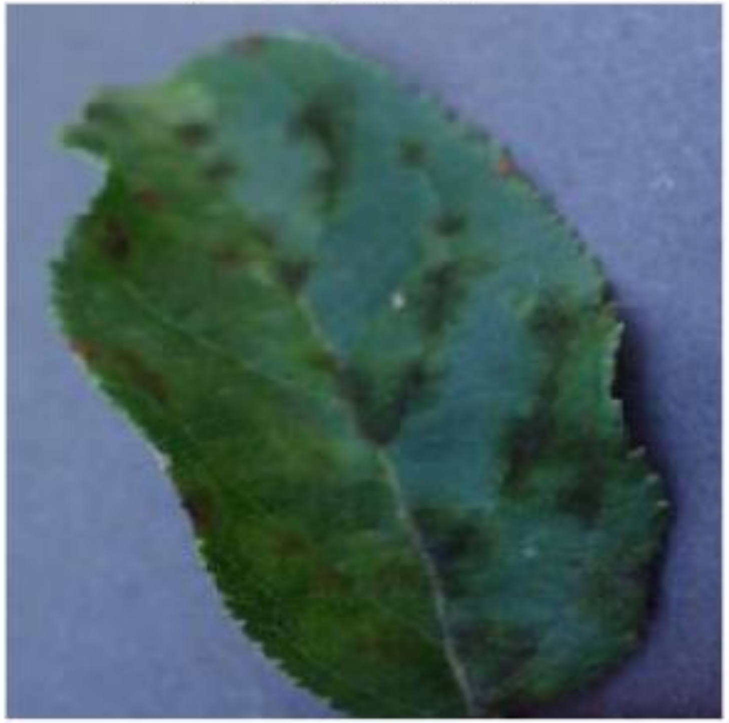
Potato___Early_blight



```
image_path = "/home/jupyter/test/test/AppleScab2.JPG"
```

```
Following is our prediction:
```

Apple___Apple_scab



Next Steps

- Ensemble Models
- Deploy a Detection Application
- Spark Scaling to speed up the processing

Conclusion

- Thank you for listening to our presentation
- Questions

Contact Information

- jkhidaroo@gmail.com
- pjdoucet@gmail.com

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.