Classifying Crop Types in Rwanda Using Drone Imagery

By: Kristin Chang and Jenna Epstein

MUSA650 Final Project

Context

- **12.3 million people** live in Rwanda (2018)
- **⅕** of the population is **food insecure**
- **89%** of the country's economy is **small-scale agriculture**
- **3 essential food secure crop types:**
    - Banana
    - Maize
    - Legumes

Image Source: USAID

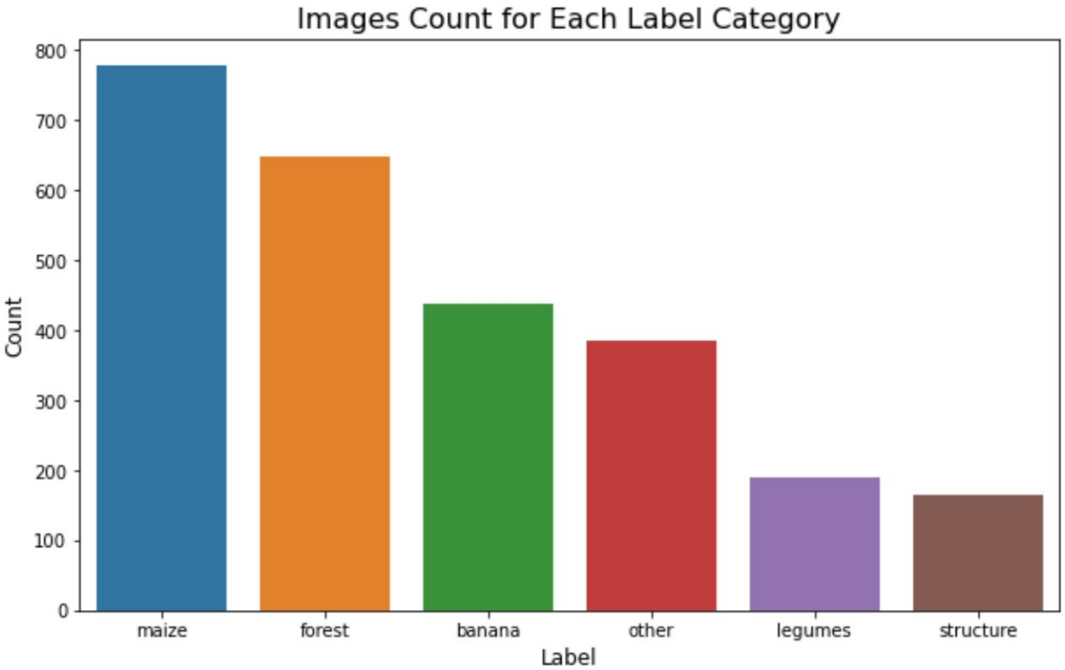
Objective

Develop an accurate yet generalizable machine learning algorithm for identifying crops, with a particular focus on bananas, maize, and legumes*, in Rwanda.

* *These three "staple crops" are key for combating food insecurity in Rwanda.*

The Dataset

2,606 images collected from drones flown across five acro-ecological regions of **Rwanda**. Images fall in **six classes**.



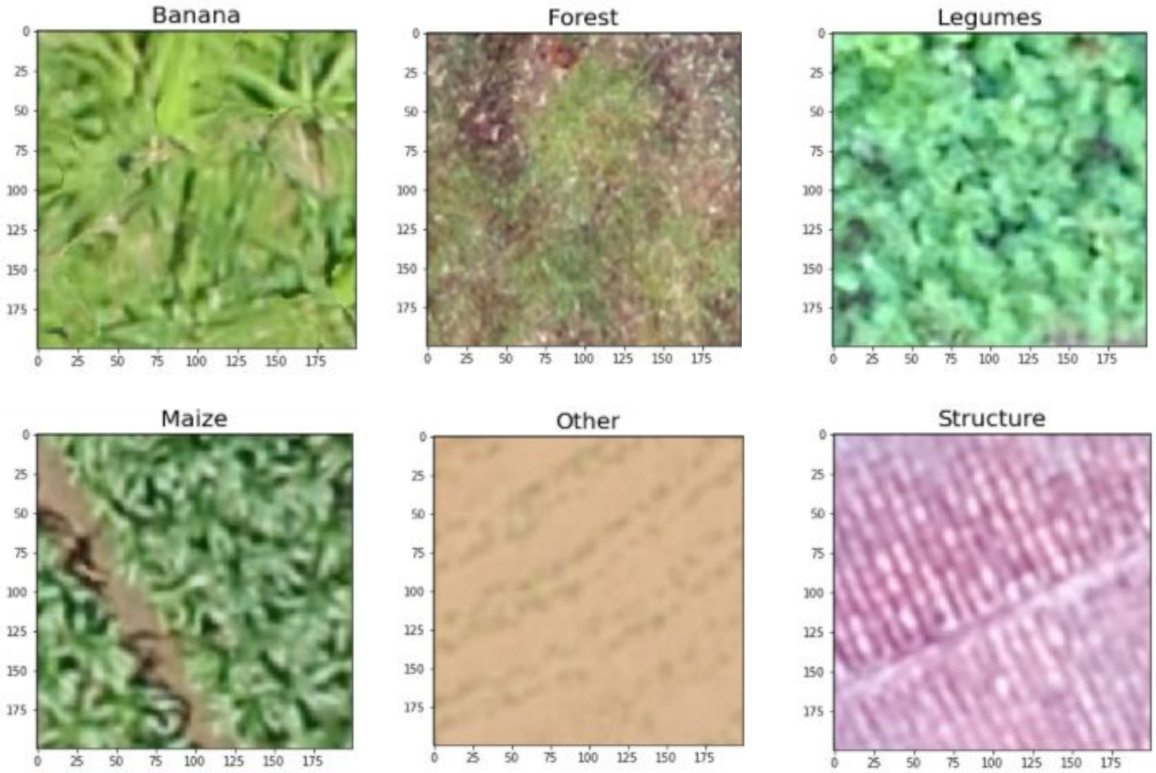
Data source: https://mlhub.earth/data/rti_rwanda_crop_type

Radiant Earth Foundation
EARTH IMAGERY FOR IMPACT

RTI INTERNATIONAL

Image Samples

Each image is 200 x 200 pixels with 3 color channels (RGB)



Two Approaches

Supervised Machine Learning:

1. K-Nearest Neighbors
2. Random Forest

80% training set,
20% testing set
+
stratify

Deep Learning:

1. Simple CNN
2. VGG16
3. MobileNetV2

50% training set,
50% testing set
+
stratify

Data augmentation
+
80% training set,
20% testing set
+
shuffle

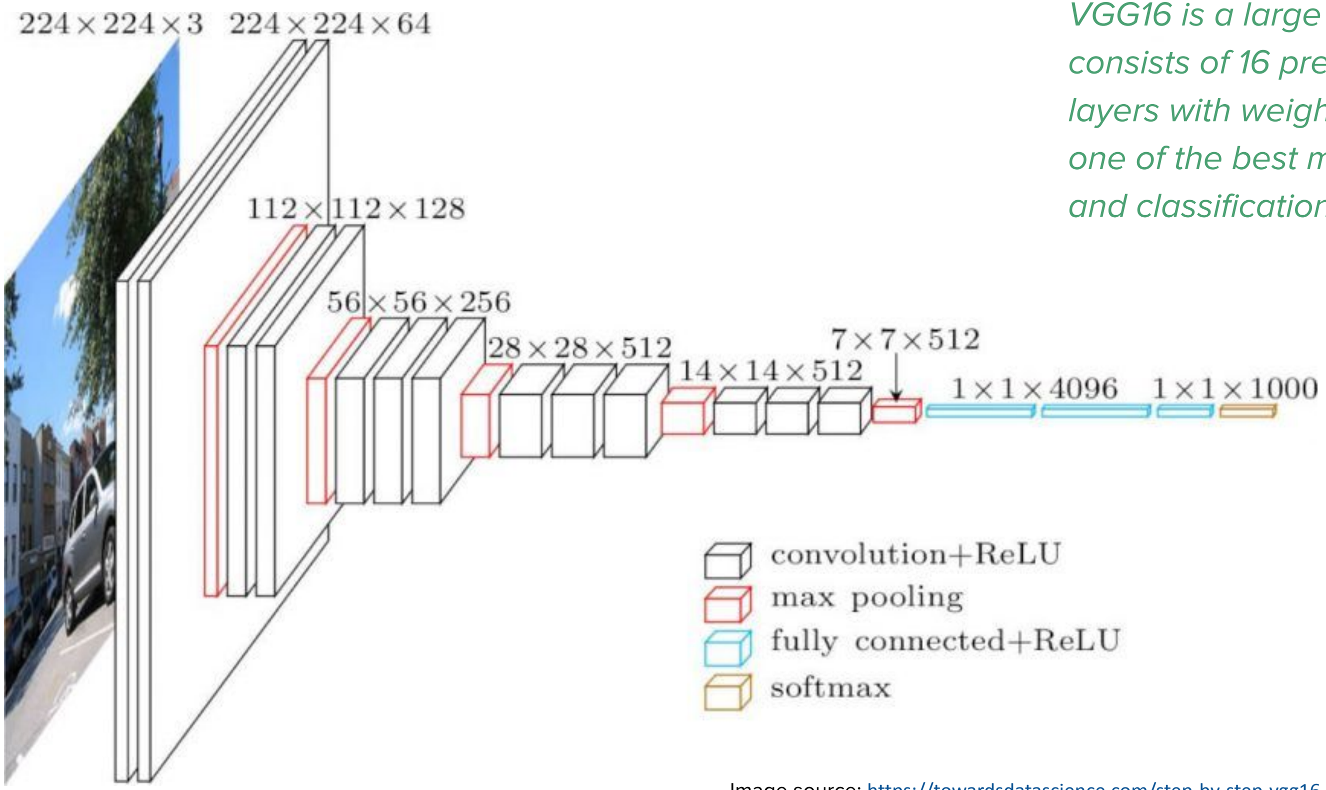
Comparing Testing Accuracy

KNN	Random Forest	Simple CNN	VGG16	MobileNetV2
0.42	0.62	0.45	0.78	0.79

Our two best models

VGG16 and MobileNetV2

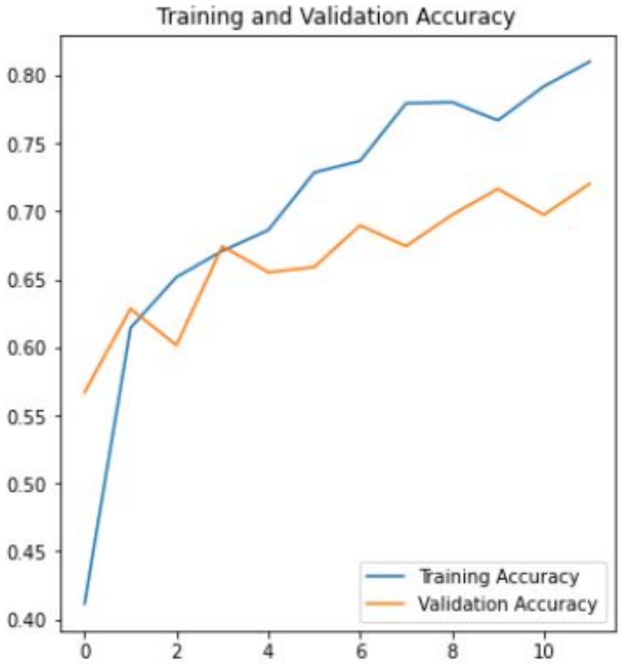
1) VGG16: The Model Architecture

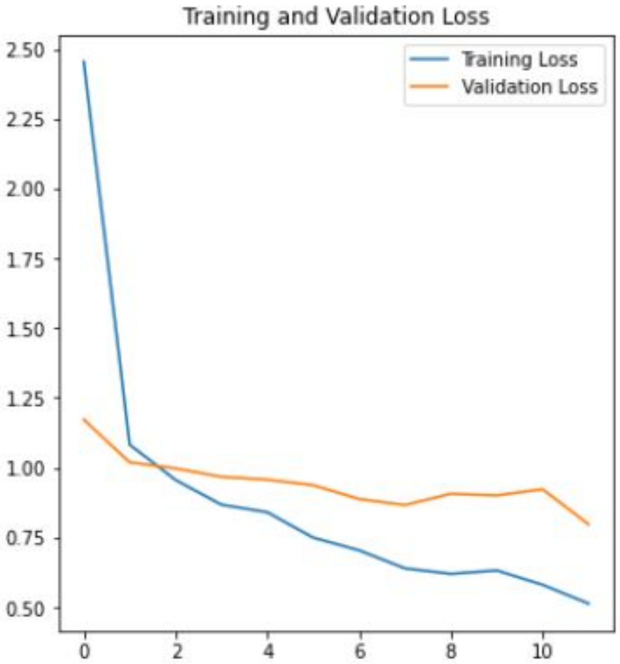


VGG16 is a large neural network that consists of 16 pre-trained convolution layers with weights and is considered to be one of the best models for object detection and classification to date.

Image source: https://towardsdatascience.com/step-by-step-vgg16-implementation-in-keras-for-beginners-a833c686ae6c

1) VGG16: Training the Model



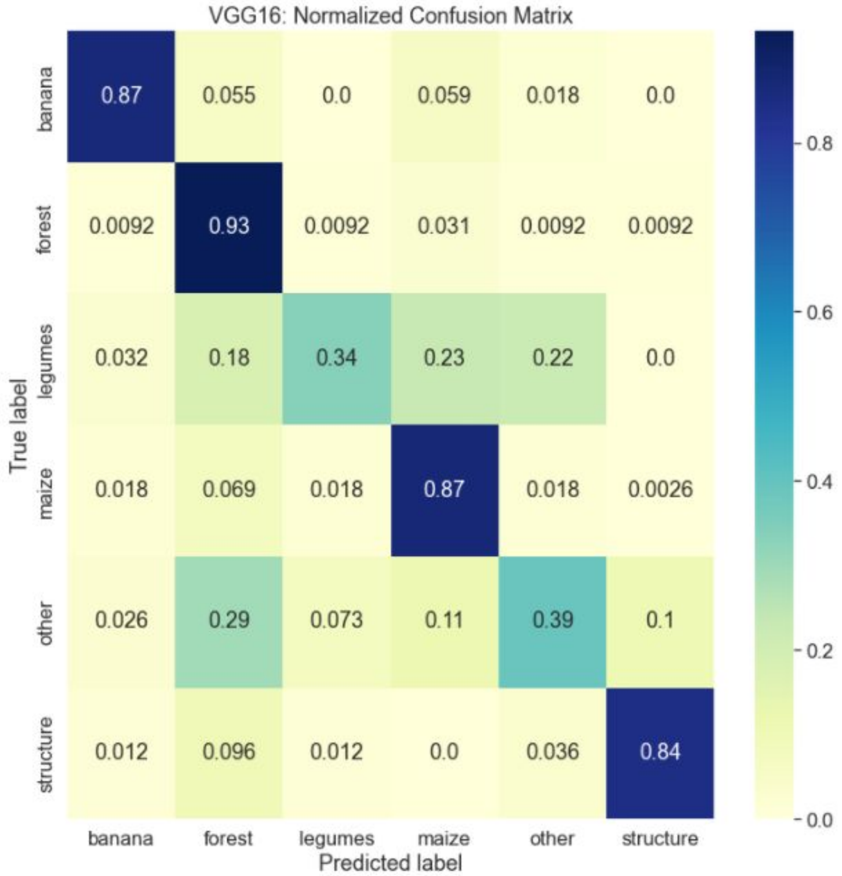



```
Model: "model_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_2 (InputLayer)         [(None, 200, 200, 3)]     0
_________________________________________________________________
block1_conv1 (Conv2D)        (None, 200, 200, 64)      1792
_________________________________________________________________
block1_conv2 (Conv2D)        (None, 200, 200, 64)      36928
_________________________________________________________________
block1_pool (MaxPooling2D)   (None, 100, 100, 64)      0
_________________________________________________________________
block2_conv1 (Conv2D)        (None, 100, 100, 128)     73856
_________________________________________________________________
block2_conv2 (Conv2D)        (None, 100, 100, 128)     147584
_________________________________________________________________
block2_pool (MaxPooling2D)   (None, 50, 50, 128)       0
_________________________________________________________________
block3_conv1 (Conv2D)        (None, 50, 50, 256)       295168
_________________________________________________________________
block3_conv2 (Conv2D)        (None, 50, 50, 256)       590080
_________________________________________________________________
block3_conv3 (Conv2D)        (None, 50, 50, 256)       590080
_________________________________________________________________
block3_pool (MaxPooling2D)   (None, 25, 25, 256)       0
_________________________________________________________________
block4_conv1 (Conv2D)        (None, 25, 25, 512)       1180160
_________________________________________________________________
block4_conv2 (Conv2D)        (None, 25, 25, 512)       2359808
_________________________________________________________________
block4_conv3 (Conv2D)        (None, 25, 25, 512)       2359808
_________________________________________________________________
block4_pool (MaxPooling2D)   (None, 12, 12, 512)       0
_________________________________________________________________
block5_conv1 (Conv2D)        (None, 12, 12, 512)       2359808
_________________________________________________________________
block5_conv2 (Conv2D)        (None, 12, 12, 512)       2359808
_________________________________________________________________
block5_conv3 (Conv2D)        (None, 12, 12, 512)       2359808
_________________________________________________________________
block5_pool (MaxPooling2D)   (None, 6, 6, 512)         0
_________________________________________________________________
flatten_6 (Flatten)          (None, 18432)             0
_________________________________________________________________
dense_9 (Dense)              (None, 256)               4718848
_________________________________________________________________
dropout_8 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_10 (Dense)             (None, 6)                 1542
=================================================================
Total params: 19,435,078
Trainable params: 4,720,390
Non-trainable params: 14,714,688
_________________________________________________________________
```

Test Accuracy: 0.78

1) VGG16: Results

LABEL	PRECISION	RECALL	F1-SCORE	SUPPORT
0 (banana)	0.91	0.87	0.89	219
1 (forest)	0.72	0.93	0.81	325
2 (legumes)	0.56	0.34	0.42	95
3 (maize)	0.84	0.87	0.85	389
4 (other)	0.66	0.39	0.49	192
5 (structure)	0.74	0.84	0.79	83
ACCURACY			0.78	1303
MACRO AVG	0.74	0.71	0.71	1303
WEIGHTED AVG	0.77	0.78	0.76	1303

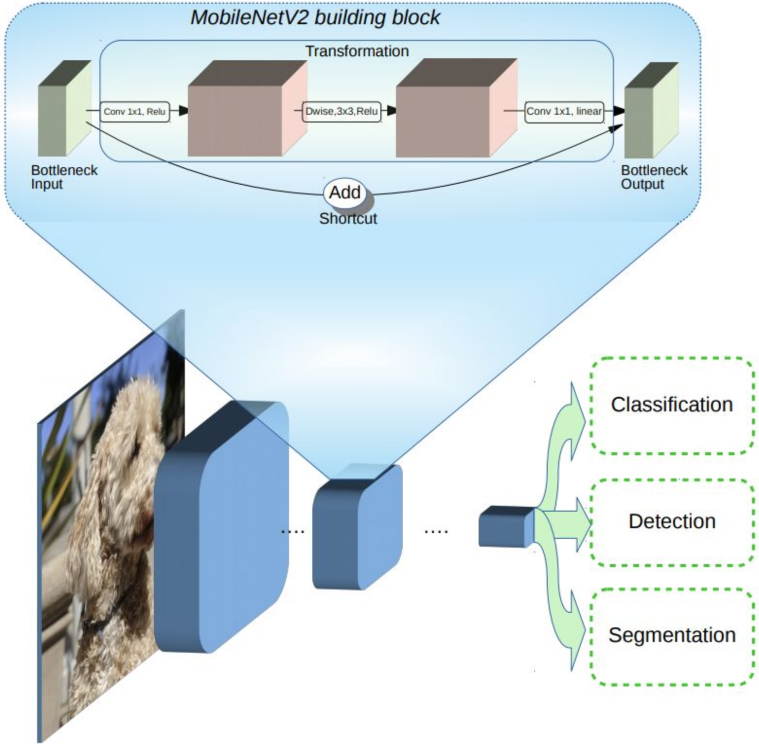


2) MobileNetV2: The Model Architecture

MobileNetV2 is a very effective feature extractor for object detection, segmentation, and classification. MobileNetV2 differs from the original MobileNet by using inverted residual blocks with bottlenecking features, and has a lower parameter count.

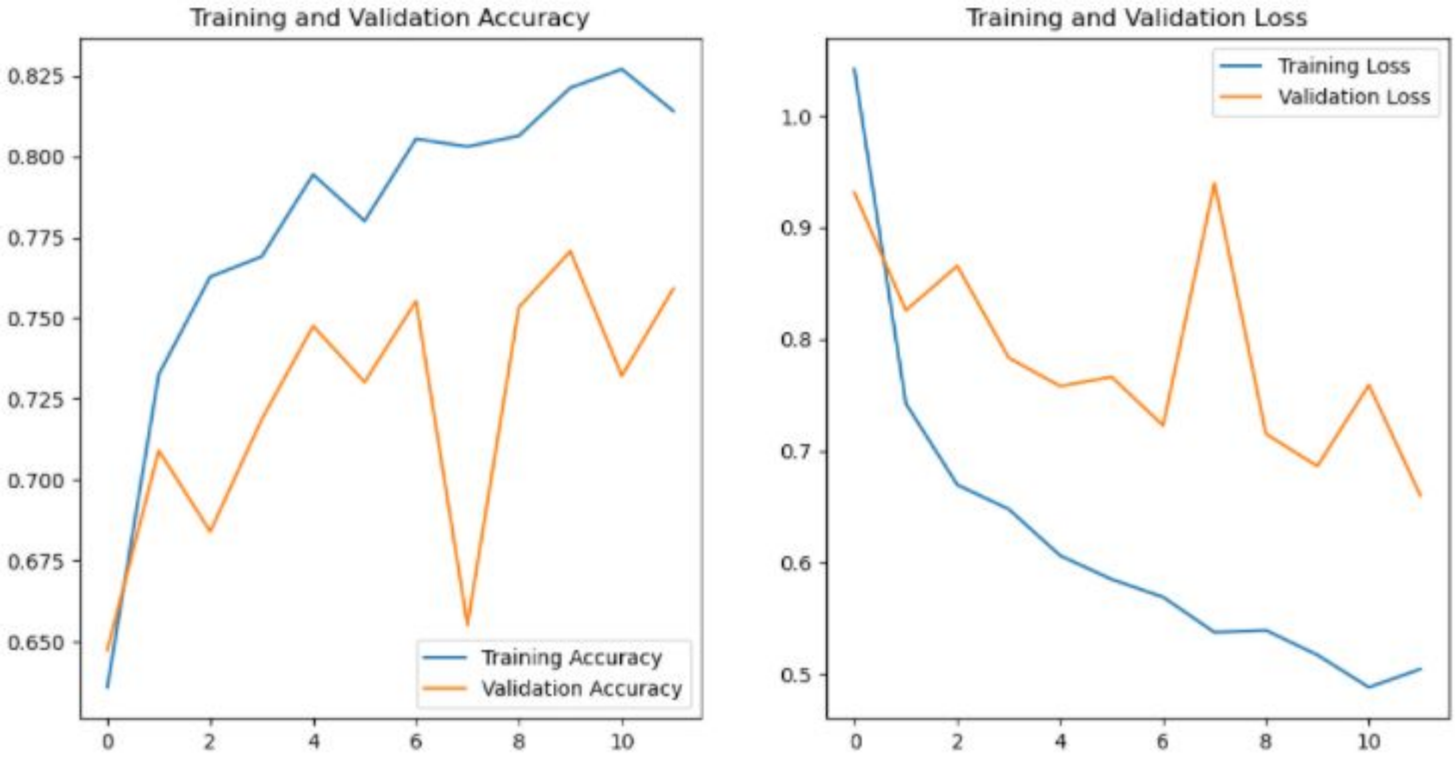
Input	Operator	t	c	n	s
$224^2 \times 3$	conv2d	-	32	1	2
$112^2 \times 32$	bottleneck	1	16	1	1
$112^2 \times 16$	bottleneck	6	24	2	2
$56^2 \times 24$	bottleneck	6	32	3	2
$28^2 \times 32$	bottleneck	6	64	4	2
$14^2 \times 64$	bottleneck	6	96	3	1
$14^2 \times 96$	bottleneck	6	160	3	2
$7^2 \times 160$	bottleneck	6	320	1	1
$7^2 \times 320$	conv2d 1x1	-	1280	1	1
$7^2 \times 1280$	avgpool 7x7	-	-	1	-
$1 \times 1 \times 1280$	conv2d 1x1	-	k	-	

Basic model architecture



Sources: [MobileNetV2: The Next Generation of On-Device Computer Vision Networks](#) (right)
[MobileNetV2: Inverted Residuals and Linear Bottlenecks](#) (left)

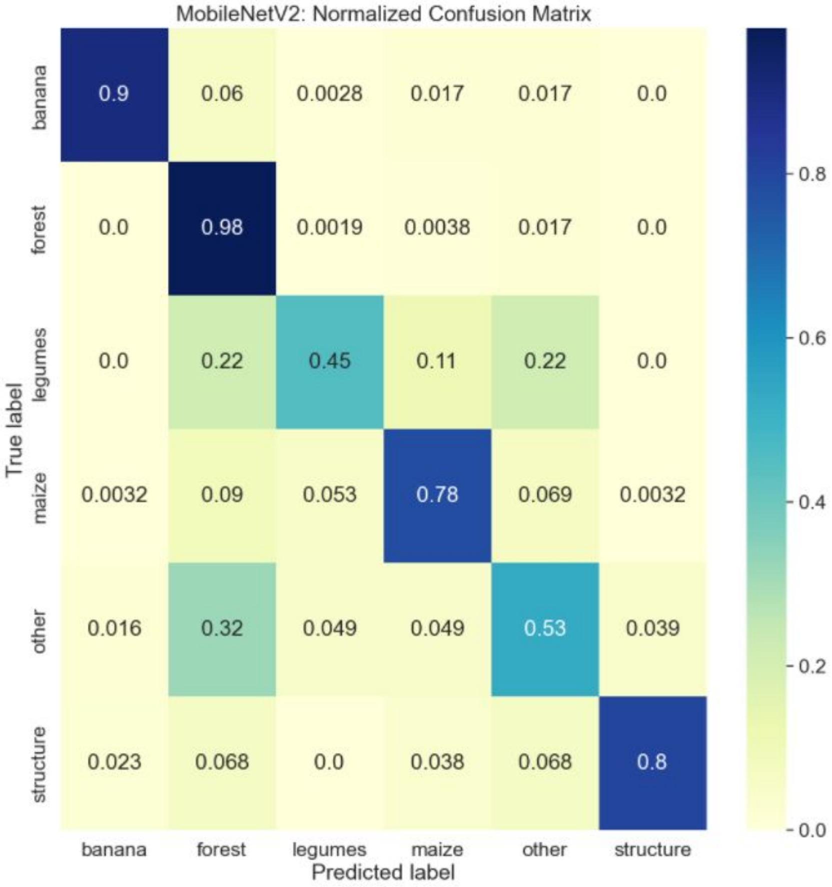
2) MobileNetV2: Training the Model



Test Accuracy: 0.79

2) MobileNetV2: Results

LABEL	PRECISION	RECALL	F1-SCORE	SUPPORT
0 (banana)	0.97	0.90	0.94	351
1 (forest)	0.70	0.98	0.82	520
2 (legumes)	0.58	0.45	0.51	152
3 (maize)	0.92	0.78	0.84	623
4 (other)	0.62	0.53	0.57	308
5 (structure)	0.88	0.80	0.84	133
ACCURACY			0.79	2087
MACRO AVG	0.78	0.74	0.75	2087
WEIGHTED AVG	0.80	0.79	0.79	2087



Discussion of Results

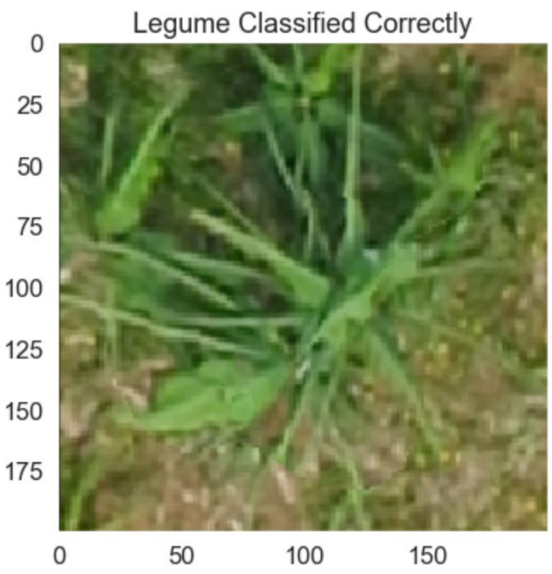
MobileNetV2 is our model with the highest classification accuracy score of 79% (VGG16 is the second best, with 78% accuracy)..
Classification accuracy is high for the banana, forest, maize, and structure labels.

While we were able to develop a model with an acceptable level of accuracy, we were not able to achieve acceptable accuracies **for all three** priority crops (bananas, maize, and legumes). In fact, our **classification accuracy for legumes is much lower** than the other labels.

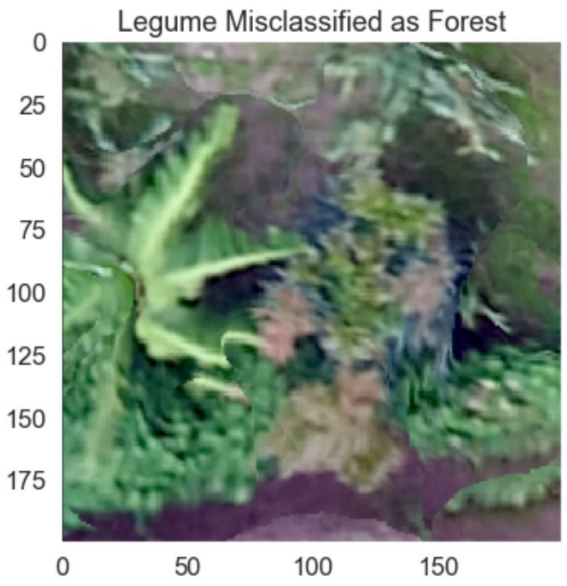
In practice, we would want to improve the classification accuracy of **legumes** before recommending our model for use in the real world.

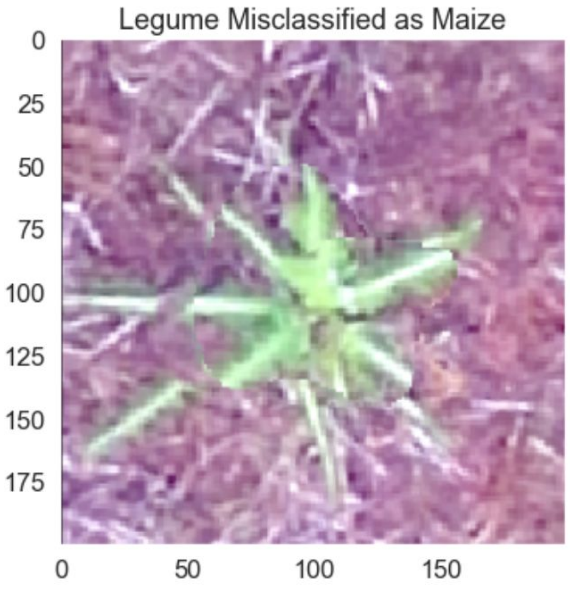
Discussion - VGG16

Despite poor classification accuracy for legumes, we would consider exploring ways to hypertune our VGG16 model as the overall accuracy score was fairly acceptable (78%).



Successful with VGG16

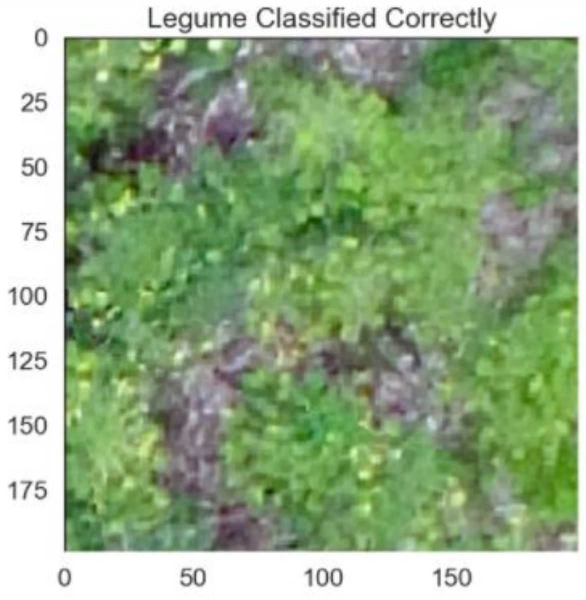




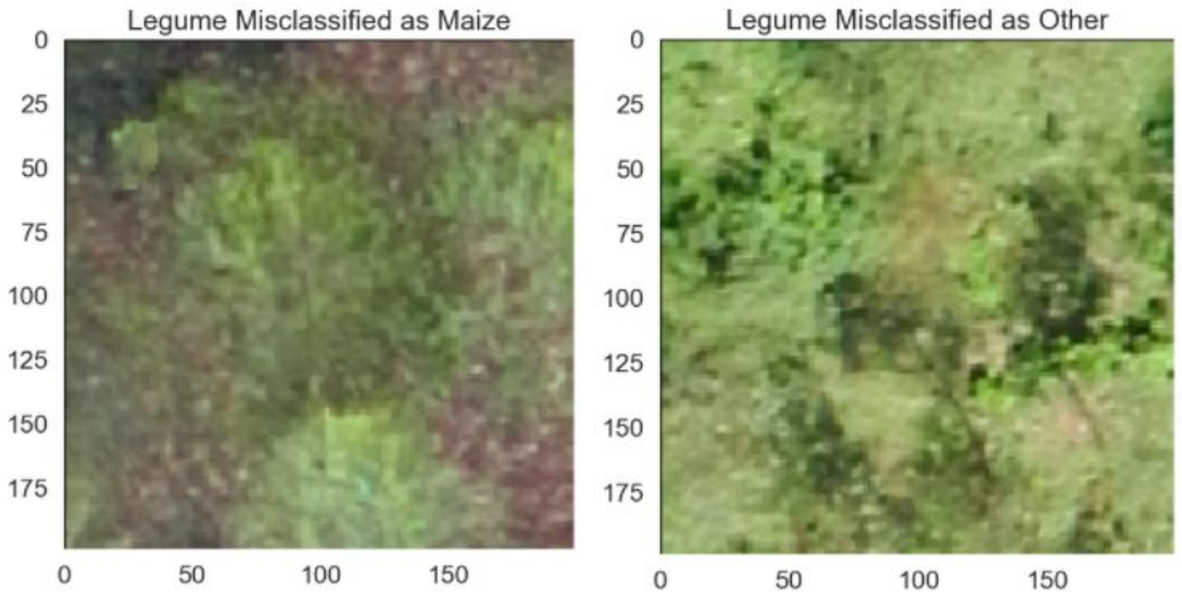
Unsuccessful with VGG16

Discussion - MobileNetV2

*Accuracy still the worst for **legumes**. Most misclassification for legumes are with images labeled **'other'.** Reminder that overall accuracy was 79%.*



Successful with MobileNetV2



Unsuccessful with MobileNetV2

******* *NOTE: images displayed may be augmented*

Areas For Improvement

1. Improve classification accuracy of legumes label
   a. Oversample legumes data
2. Improve overall model accuracy
   a. VGG16: Hypertune “top” of the model’s dense layers
   b. MobileNetV2:
      i. Experiment with other data augmentation techniques
      ii. Try other weights (pre-trained and not)
      iii. Run model on just the three secure crop labels to see if the forest, other, and structure images are creating noise that prevents precise learning when training the model
3. Explore multispectral imagery

Thank you!