Classifying Crop Types in Rwanda Using Drone Imagery

Kristin Chang (krchang@design.upenn.edu) & Jenna Epstein (jennaeps@design.upenn.edu)

Introduction

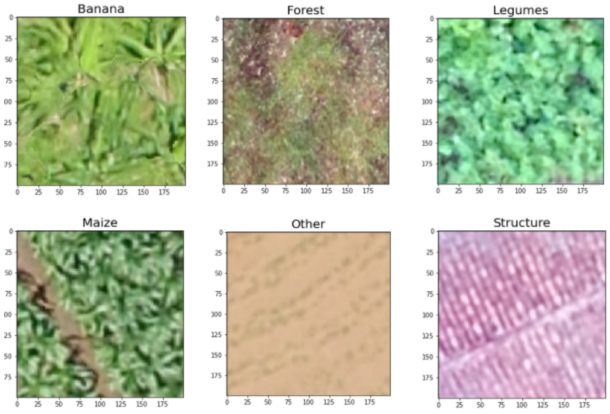
This project utilizes drone imagery of Rwanda from December 2018 to February 2019 produced by nonprofit research institute, RTI International (RTI), to better inform the country's crop management decisions. We are applying machine learning techniques to classify each image into one of six classes - three of which are staple crops for food security in the country[1].

The goal of this project is to develop a robust machine learning model to be able to analyze large plots of land at a time and more efficiently gain an understanding of crop distribution in Rwanda. Specifically, we seek to classify images so that we can see how many of the three essential crop types - banana, maize, and legumes - are present across the area and how prevalent these staple crop types are compared to the other two image classes. We are also interested in how well different models will be able to classify the three main crop types as they appear fairly similar to the human eye.

We hope to show that machine learning techniques can inform decision making to prioritize agricultural investment - especially if there is relative scarcity of these essential crops compared to non-crop land. According to the World Population Review, Rwanda had a population of 12.3 million people in 2018 and is expected to grow for the rest of the century[2]. Additionally, the World Food Programme states that almost one fifth of the population is food insecure and that small-scale agriculture accounts for 89% of the country's economy[3]. Classifying Banana, Maize, and Legume crop types using machine learning will more efficiently inform food security levels as these three crops are crucial to people's livelihoods in Rwanda. Identifying these crop types among various land cover images will also provide a useful overview to help farmers and government leaders assess whether their food supply can support the country's growing population.

Data

We used the "Drone Imagery Classification Training Dataset for Crop Types in Rwanda'' produced by RTI and published by Radiant MLHub for open access in April 2021. The data is provided in two SpatioTemporal Asset Catalog collections: a source imagery collection and a labels collection for 2,611 images[4] in five agro-ecological zones in Rwanda. Source imagery was collected from aerial drones over a three-month period. The labels collection contains six classes: Banana, Maize, Legume, Forest, Structure, and a catch-all "Other" category; labeling was performed by team members manually using a digital tagging web interface. Samples of each image class are shown at right.



[1] https://mlhub.earth/data/rti_rwanda_crop_type
[2] https://worldpopulationreview.com/countries/rwanda-population
[3] https://www.wfp.org/countries/rwanda
[4] 2,606 images were available upon download

Methods

Our approach was to start with two supervised machine learning methods - K-Nearest Neighbors and Random Forest - and compare their performance to more complex deep learning approaches. These deep learning approaches involved three different forms of neural networks. This section describes the various steps we took to create each model and how we decided on the parameters displayed in the final version of our notebook. All results with the best parameters are discussed in the results section below.

KNN

For our supervised learning approaches, we loaded in the data using the *PIL* and *os* libraries, extracted the color channels from each image, and flattened to a feature matrix. We also created a matrix of numeric labels for each image. Then, we split the data using 80% as the training set and 20% as the testing set and using the stratify parameter to ensure an even distribution of labels across both sets.

Per our background research on agriculture classification, KNN is a suitable method for multiclass classification[5]. KNN assumes similar things are within close proximity of each other. We used grid search cross validation to identify the best set of parameters based on classification accuracy.

Random Forest

Random Forest is a type of ensemble model preferred for supervised classification because of its ability to address overfitting while maintaining high accuracy. We used the same matrices and train-test data split as above prior to applying this model. Again, we experimented with different depth and estimator parameters by using grid search cross validation and applied the set with the highest accuracy.

Neural Networks

For our deep learning approaches, we loaded the RGB images using glob, os, and skimage libraries and converted them into a matrix using *numpy.stack()*. The data was initially split using 50% as the training set and 50% as the testing set - we found this split provided the most balance among the labels in our data for training our models.

We built three deep learning models using neural networks. We began with a simple convolutional neural network (CNN) structure and then experimented with more complex structures, transfer learning, and applied some data augmentation techniques to improve our accuracy score. The following subsections describe the methods for each neural network using the keras and tensorflow libraries.

Simple CNN

For our first deep learning model, a simple CNN consisting of:

- a rescaling layer to preprocess the data,
- a 2-dimensional convolution layer with 64 output filters with kernel size (3,3) and strides of (1,1), a batch normalization layer,
- an activation layer using the *relu* function,

[5] https://towardsdatascience.com/multiclass-classification-using-k-nearest-neighbours-ca5281a9ef76

- a 2-dimensional maxpooling layer with a pool size of (2,2),
- a dropout layer with rate of 0.5,
- a flattened layer ,
- and a dense layer with units equal to the number of labels (6) and the *softmax* activation function.

The above sequence of layers provided the highest accuracy after compiling using the adam optimizer and evaluating on our test set. Previous versions of this model consisted of using MinMaxScaler() on the data prior to applying the model, up to 4 total convolution layers with 32 or 64 output filters, dropout layers with rates of 0.25 or 0.5, and multiple dense layers with units of 512 or greater (*Two different pre-processing approaches are displayed in the notebook attached in the appendix as CNN Attempt #1 and CNN Attempt #2*). Adding more complexity to this CNN; however, decreased accuracy. One explanation is that increasing density and sampling with convolution layers may cause overfitting of our training data despite using BatchNormalization() and dropout layers for regularization. Nonetheless, the above sequence of layers provided the best accuracy when applied to our test set, but the accuracy score was not sufficiently acceptable so we explored two different neural network architectures: VGG16 and MobileNet.

VGG16

VGG16 is a large neural network that consists of 16 layers with weights and is considered to be one of the best models for object detection and classification to date. The *VGG16()* keras function allows us to load pretrained convolution layers using ImageNet weights[6]. We flattened the output layer and created a new "top" of the model with the following fully-connected layers:

- a dense layer with 256 units and *relu* activation function,
- a dropout layer with rate of 0.5,
- and another dense layer with units equal to the number of labels(6) and the *softmax* activation function.

The depth and number of fully-connected nodes of VGG networks are what make it a popular neural network architecture. Drawbacks to this framework include computation time and large weights, but we did not run into issues here because our dataset is not significantly large.

MobileNetV2

For our last CNN, we elected to use the MobileNetV2 model available upon import from *keras.applications.mobilenet_v2*. This model is a small, low-latency convolutional neural network, and it is based on the popular, open ImageNet model used for image classification[7]. We elected to try it after reading through a tutorial from Radiant Earth MLHub that a variation of it to some success with this same Rwandan crop dataset[8]. Its size of 14MB makes it the smallest of all deep CNN architectures, while also operating faster than VGG16.

We experimented with different data augmentation techniques and data splitting. Our research indicated that the size of the dataset (2,606 images) is relatively small, and using the image data generator during preprocessing may help. Data augmentation increases our dataset size by creating variations of the images, thus

[6] https://www.learndatasci.com/tutorials/hands-on-transfer-learning-keras/
[7] https://www.tensorflow.org/api_docs/python/tf/keras/applications/mobilenet_v2/MobileNetV2
[8] https://github.com/RadiantMLHub/ml4eo-bootcamp-2021/tree/main/Lecture%206

crafting synthetic data. We experimented with different techniques for augmentation including rotating images to various degree points, shifting images, and flipping images. We ultimately used the following:

- Rescaling
- Rotation range of 30 degrees
- Width shift range of 10%
- Height shift range of 10%
- Horizontal flipping
- Zoom range of 20%

The train-validation overall split was 80-20. We also set aside 20% of the data randomly as a testing set for use in transfer learning, when the model is applied to "new" data. Note that the MobileNetV2 requires images to be of specific sizes, so we reshaped the images from 200x200x3 to 224x224x3 (largest form that the model can take).

According to the literature on the MobileNetV2 architecture, the model is based on an "inverted residual structure where the input and output of the residual block are thin bottleneck layers opposite to traditional residual models which use expanded representations in the input…[and] MobileNetV2 uses lightweight depthwise convolutions to filter features in the intermediate expansion layer".[9] The basic model architecture from this source is shown to the right:

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	-	

Our implementation of MobileNetV2 has a model summary spanning over 300 lines of code with many layers, so we are not including those details here. The *mobilenet_model.summary()* line in the linked Python notebook prints the full output in a text editor.

Results

Below are some of the accuracy metrics for each of our models. Additional confusion matrices, all classification reports, and metrics plots can be found in the Python notebook linked to the appendix.

KNN

We used the *GridSearchCV()* function from scikit learn to identify the optimal number of neighbors (*k*) to specify in the KNN model: 27 neighbors. After running the model with *k*=27, the resulting accuracy score is 42%. See the appendix for the classification report and confusion matrix.

[9] MobileNetV2: Inverted Residuals and Linear Bottlenecks, Sandler M, Howard A, Zhu M, Zhmoginov A, Chen LC. arXiv preprint. https://arxiv.org/abs/1801.04381, 2018.

Random Forest

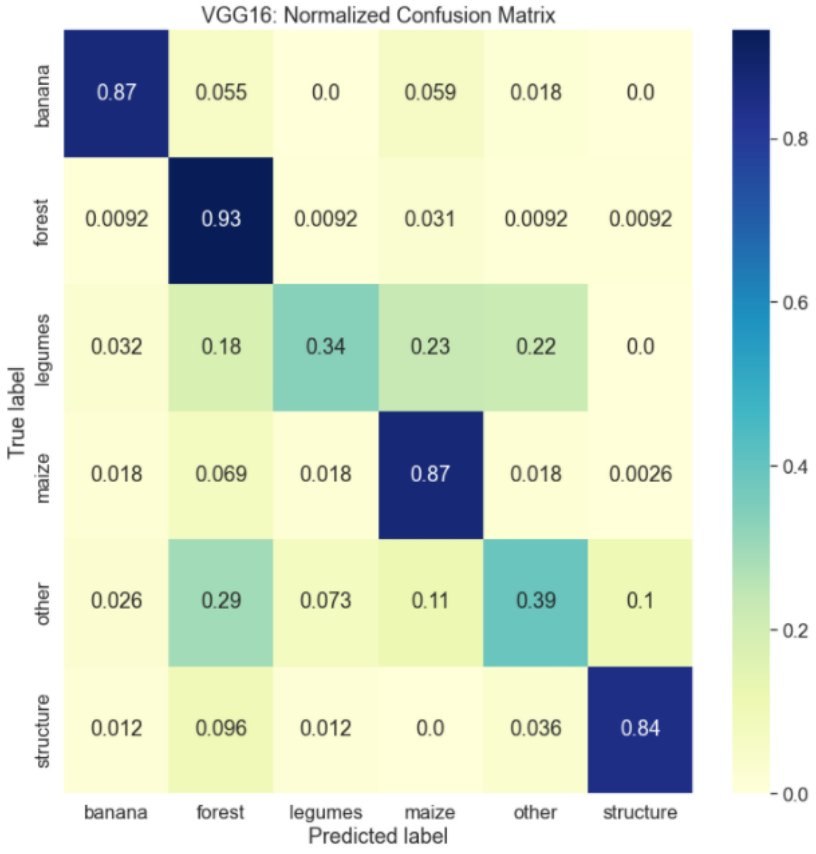
The results from the KNN model were not satisfying so we decided also to use a Random Forest classifier. We ran GridSearchCV, which max depth = 15 and number of estimators = 700 as optimal parameters. This model yielded an accuracy score of 61%, which is higher than the KNN model.

Simple CNN

Although the accuracy of our random forest model is okay, we expect a CNN model to be significantly better. Our simple CNN model, however, resulted in an accuracy of 45%. After manually adding more convolution and fully-connected layers to this CNN, we decided to implement more complex architectures.

VGG16

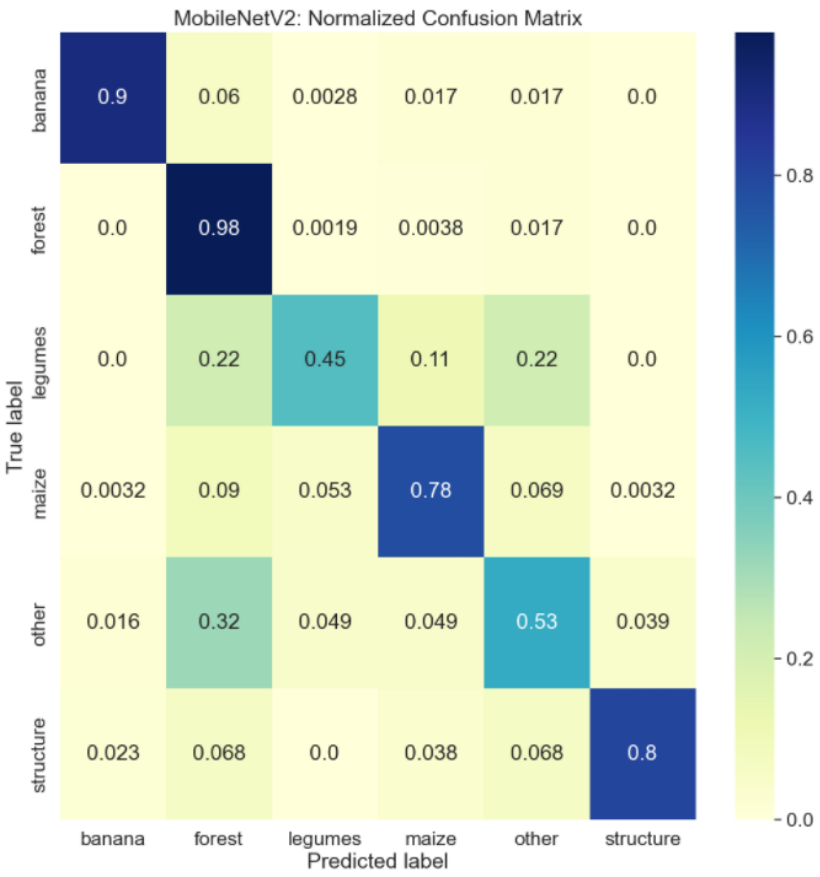
The architecture of VGG16 proved to be a fairly effective method of training for our data. This model resulted in a test accuracy of 78%. We applied data augmentation techniques and re-ran this model but it did not improve the accuracy score, so we did not include these results in our report. As such, we reverted the model to run again on the original RGB images to confirm the 78% testing accuracy - the resulting confusion matrix is shown below.



This matrix shows that our VGG16 model was able to classify forest images the best, banana, maize, and structure images fairly well, but did not classify legumes or "other" images well. These results prevent us from recommending this as an effective model to achieve our project goal and are discussed further in the following section.

MobileNetV2

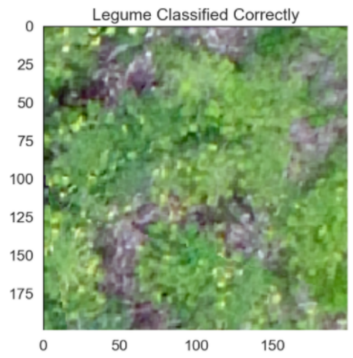
Peak validation accuracy hits 77.1% on the validation data with an overall of 76.9% (rounding to 77%). This is slightly better performing compared to the VGG16, so we move forward to predicting on the testing set of 1827 images that was held out earlier (note that there are more images run through the model across all sets in this approach due to data augmentation). This model resulted in an accuracy of 79% on the testing set - slightly higher than the accuracy of VGG16. The confusion matrix for this model is shown below.



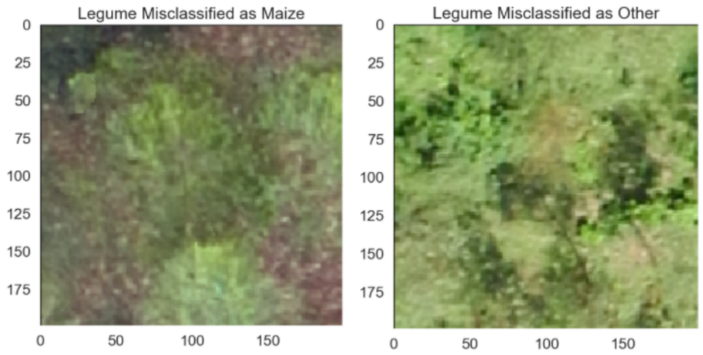
Similar to the VGG16 model, this matrix reveals that our MobileNetV2 model classified forest images the best, banana, structure, and maize images fairly well, and did not classify legumes or "other" images that well either. However, there is significant improvement with the legume and other labels when compared to the VGG16 model. Thus, this model is more effective at achieving our goal.

Discussion

The objective of our project was to develop a robust machine learning algorithm for classifying crop types in Rwanda, with a particular focus on crops indicative of food security: bananas, maize, and legumes. The model with the highest classification accuracy score is our MobileNetV2 model. The confusion matrix for this model (shown above) reveals that classification accuracy is high for banana, forest, maize, and structure labels. While we were able to develop a model with an acceptable level of accuracy and generalizability, 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 significantly 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. The images below show one successful classification and two misclassifications of legumes.



Successful with MobileNetV2



Unsuccessful with MobileNetV2

If we were to continue working on this model, we may consider oversampling the legumes dataset prior to implementing the MobileNetV2 model. The histogram displayed in the exploratory data analysis portion of our attached notebook shows that there are fewer samples for legumes than maize and banana, but the sample size was comparable to the structure label. We wanted to see how each model would perform on the data without over or undersampling, but if given more time we would go back to further balance the data. Additionally, we would be interested in running the MobileNetV2 model on just the three crop labels to see if the forest, other, and structure images are creating noise that prevents precise learning when training the model.

We would also like to improve on the MobileNetV2 model's overall accuracy by experimenting with some additional data augmentation techniques, try out other input weights (other than the pre-trained ImageNet weights), and run the model with different batch sizes.

Despite poor classification accuracy for legumes, we would consider exploring ways to hypertune our VGG16 model as the overall accuracy score was fairly acceptable (and only slightly below that of the MobileNetV2). Adjusting the "top" of the VGG16 model by increasing the number of units in the dense layer with the *relu* activation function or adding another fully-connected dense layer may further hypertune the model. We would need to add another dropout layer to avoid overfitting and could experiment with this rate as well.

Lastly, if we are able to obtain multispectral imagery at this scale, we would be interested to see how our MobileNetV2 model would perform using color channels specific to these labels. For instance, infrared and short wave infrared bands may provide more information to better classify legumes, bananas, and even the "other" category. Overall, further refinement of our modeling for greater classification accuracy across these secure crop classes can help serve as a framework for similar agricultural/crop classification challenges in similar agro-ecological regions where drone imagery is captured.

Appendix

Additional References

Dataset Download: https://mlhub.earth/data/rti_rwanda_crop_type
Dataset Documentation: https://radiantearth.blob.core.windows.net/mlhub/rti-rwanda-crop-type/documentation.pdf
MLHub Data Download Tutorial by Radiant Earth:
https://nbviewer.org/github/radiantearth/mlhub-tutorials/blob/main/notebooks/radiant-mlhub-api-know-how.ipynb

Project-Specific Links

GitHub Repository for our Project: https://github.com/jennaepstein/MUSA650_Final_ChangEpstein

- Python notebook for accessing the data:
  https://github.com/jennaepstein/MUSA650_Final_ChangEpstein/blob/main/Accessing_MLHub_data.ipynb
- Python notebook with exploratory data analysis and modeling:
  https://github.com/jennaepstein/MUSA650_Final_ChangEpstein/blob/main/MUSA650_Final_ChangEpstein.ipynb
- PDF of Python notebook with exploratory data analysis and modeling:
  https://github.com/jennaepstein/MUSA650_Final_ChangEpstein/blob/main/MUSA650_Final_ChangEpstein.pdf

Google Slides Presentation for our Project:
https://docs.google.com/presentation/d/1sY9L-K0ipNoKYSdK2JGLrXYgBashzYUcuNGf4wXdjls/edit?usp=sharing