Detecting Orchards in West Africa Using Convolutional Neural Networks

Solomon Kim, Josh Och, Eric Wang
Department of Computer Science
Stanford University
Stanford, CA 94305
solomon3@stanford.edu, joch@stanford.edu, ericw553@stanford.edu

Abstract

Food shortage and economic strife are two endemic problems in West Africa. TechnoServe, a charitable organization, is attempting to combat both of these by providing services to local orchard farmers to help them increase their yield. In order to find orchard farmers, they need a software program that will look at an image from OpenStreetMaps of either an orchard or a forest and identify whether or not it contains an orchard.

This report presents a method that uses a pre-trained ResNet CNN with a binary cross entropy loss function and the Adam optimization algorithm to classify whether or not an image is an orchard. The model was found to be moderately successful, reaching a test accuracy of 70%. Based on analysis from saliency maps, we decided to rerun the model on images from OpenStreetMap that were more zoomed-in, and we achieved a higher test accuracy of 79%. As a direction for future research, we tried to see how well our model would generalize by running it on a dataset of orchards and developed land, instead of orchards and forests. Our results were even better, with a test accuracy of 98%. These results inspire confidence that our model will generalize well, but we think more work should be done in order to verify the model's performance in other settings.

1. Introduction

In West Africa, there is a charitable organization called TechnoServe that works to help orchard farmers increase their crop yield through training and technology. This initiative helps increase the food supply in the area and increases the economic prospects for these farmers.

According to [1], "Acute food insecurity in Africa has increased by over 60 percent in the past year and threatens to widen further as the effects of COVID-19 exacerbate other drivers such as conflict and political mismanagement." This acute food shortage, especially in West Africa, where the number of people facing food crises has risen from 3.2 million to 13.7 million in just the last year [1], has made providing services for the people of West Africa all the more critical.

In order to provide these services, TechnoServe needs to figure out where these orchard farmers live. The organization currently employs a manual process of looking through OpenStreetMap images to identify orchards. In order to make this process more efficient, the organization has asked us, through Professor David Lobell of the Earth System Science department, to write software that can predict whether a given OpenStreetMap image contains an orchard.

Specifically, the input to the model is an OpenStreetMap image from somewhere in West Africa of either an orchard or a forest, the model is a trained convolutional neural network, and the output is a prediction: either "Orchard" or "Forest".

2. Related Work

There is literature on how to detect object from images using traditional approaches. [3] developed feature-based approaches to characterize and detect orchards using high resolution satellite imagery. Several newer papers focus on the deep learning method.

[4] explains the efficiency and benefits that deep learning has on the detection of orchards, while [12] described a new unsupervised method for simultaneous detection and segmentation of orchards in complex scenes.

[13] designed two types of deep learning models: one is based on Long Short-Term Memory (LSTM), and the other is based on one-dimensional convolutional (Conv1D) layers to classify vegetation images from satellite images. Three widely-used classifiers were also tested for comparison: XGBoost, Random Forest, and Support Vector Machine.

[7] breaks down the principles of CNN and explains why this technique is particularly suited for vegetation, while [5] used additional filters to improve on detection performance. Furthermore, [6] dealt with the issue of overlap common in

many orchards, whereas [11] sought to deal with detection at different stages of growth.

[14] successfully detected citrus plants using images from Brazil. [8] extended their study to not only citrus orchards but also cornfields.

3. Method

3.1. Other potential approaches

Convolutional neural networks have emerged as a dominant strategy for image classification problems. But, before beginning this approach, it is worth considering whether a different, possible simpler model might be more effective.

Our group attended a PhD defense presentation given by Sherrie Wang, a student in Stanford's Earth Sciences department. In it, she discussed a method for differentiating different types of vegetation by looking at images taken at different times throughout the year. By looking at the rate of change in the shade of green throughout the year, she can fit a sinusoidal curve to each class of vegetation. For the prediction step, she is given a set of images of the same location, and she can report the sinusoidal curve that best fits these images.

This approach would require that we get multiple images from the same location, and that those images are labelled with dates. This might have been possible. However, what Sherrie reported in her presentation is that although this strategy works well in relatively homogenous regions like the American Midwest, in Africa, where there are many diverse types of orchards and forests, the number of different classes of vegetation is too high, and this model does not work well.

Thus, we decided to pursue a deep learning classifier, and given the research cited in the Related Works section above, we decided to use a CNN.

3.2. Method summary

The general strategy is to adopt a pre-trained 18-layer ResNet, add our own linear layer, and re-train the last layer of the ResNet and our linear layer. This gives us a model that can classify whether or not an image contains an orchard.

3.3. Neural network architecture and hyperparameters

In order to benchmark our performance, we first created our own small baseline model.

Baseline model: For a baseline, we wrote a simple three-layer neural network. This baseline model consists of two repeated sets of three layers each: a convolutional layer, a ReLU layer, and a MaxPool layer. These two sets are followed by a final linear layer. We were able to run this model entirely using CPU. We tuned the learning rate and regularization to maximize performance. Our tuned hyperparameters can be seen in Figure 2.

ResNet model (Figure 1): ResNet-18 [9] is an 18-layer architecture originally proposed to deal with the issue of vanishing gradients in a multi-layer deep neural network. An architecture with two blocks, the Identity Block (input/output dimensions are the same) and the Convolution Block (input/output dimensions are different), it is composed of many of what is known as a skip connection, a connection from the early parts of the network to its later parts, with information being transferred over. This allows us to bring over lost information that was previously located in the early layers, and it allows us to get a better idea of the network as a whole. Although it may seem at first that this might simply reduce overall performance, it actually does not because stacking identity mappings would produce almost the exact same rate, proving that stacking layers together normally has little effect on overall performance.

Layer Name	Output Size	ResNet-18
conv1	$112 \times 112 \times 64$	7×7, 64, stride 2
conv2_x	$56 \times 56 \times 64$	3×3 max pool, stride 2
		$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 2$
conv3_x	$28 \times 28 \times 128$	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 2$
conv4_x	$14 \times 14 \times 256$	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$
conv5_x	$7 \times 7 \times 512$	$\begin{bmatrix} 3 \times 3, 512 \\ 3 \times 3, 512 \end{bmatrix} \times 2$
average pool	$1 \times 1 \times 512$	7×7 average pool
fully connected	1000	512×1000 fully connections
softmax	1000	

Figure 1. Diagram of ResNet-18 Architecture

Our final model starts with the original pre-trained ResNet-18 and adds our own linear layer on top. The first 17 layers of the ResNet are left untrainable, while the last layer of the ResNet and our added layer are trainable. We arrived at this decision after trying a few different thresholds and determining that allowing more layers to be trained results in more overfitting.

We also tuned the learning rate and weight regularization hyperparameters for the ResNet. The values we arrived at can also be found in Figure 2.

	Baseline	ResNet18
Learning Rate	1e-6	1e-5
Regularization Rate	1e1	1e-1

Figure 2. Tuned hyperparameters for the baseline and ResNet models

3.4. Computing platforms

All of the code for our models is written using PyTorch 1.8. The data is hosted on a virtual machine in Google Cloud Platform. The code for the model is run on that same virtual machine, and the weights, along with the data, is moved over to a GPU for efficient computation, before returning the results back to the virtual machine. Our interface for writing code and running the model is a Jupyter notebook.

3.5. Loss function

Our loss function was a binary cross-entropy loss with L2 weight regularization.

Binary cross-entropy can be represented as follows. 1:

$$-\frac{1}{N}\sum_{i=1}^{N} y_i \log \hat{y}_i + (1-y_i)\log(1-\hat{y}_i) \qquad (1)$$

where $\hat{y}_i$ and y_i are the output scalar value and the target value of the i'th term, respectively, and N is the number of scalars within the output data for the model. Our purpose for using binary cross-entropy is to allow quick processing and classification of our training examples, as it is equivalent to maximum likelihood estimation fitting, guaranteeing consistency and statistical efficiency.)

3.6. Accuracy measurement

We used a simple accuracy measure:

$$accuracy = \frac{\#correct}{\#total}$$

We thought about using F1 score [10] to address any possible class imbalance, but instead, we arranged for our datasets to have approximately 50% orchards and 50% non-orchards, so this imbalance was not an issue.

For our validation and test accuracies, 50% would mean that the model's performance was no better than a random classifier. We were hoping to get our validation and test accuracies to be above 70%. In order to make sure that our model was expressive enough to overfit the training data, we also wanted to get our training accuracy to be above 95%, but this goal was mainly in service of the validation and test accuracies mentioned above.

3.7. Optimization procedure

Our method adopted Adaptive Moment Estimation, or ADAM, for stochastic gradient descent optimization. ADAM optimization relies on the first and second moment of gradient to update its learning rates. It has an increased cost, due to requiring the calculation of the second derivative, but with the added benefit of converging in circumstances that standard gradient descent may not, as it is invariant to gradient rescaling.

Mathematically, by defining the estimates of m_t and v_t as the mean and the uncentered variance of the gradients of current mini-batch g_t, respectively, as well as the decay rates as β_1 and β_2, ADAM can be represented as following equations 2:

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1-\beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1-\beta_2) g_t^2 \end{aligned} \qquad (2)$$

This gives a decaying average for both m_t and v_t that allows the gradient descent to proceed and eliminate more jagged routes to the intended destination.

4. Dataset and features

4.1. OpenStreetMap data and label generation

Bruno Lopez, a research assistant in Professor Lobell's lab, downloaded the images from OpenStreetMaps. [2].

Bruno downloaded 500 images of orchards and 500 images of forests that did not contain orchards. They were all set to have a medium-level zoom (zoom level 16 in the OpenStreetMaps API). See Figure 2 for examples of each class.

Figure 3. Example images of orchards and forests

We moved the images from a bucket in Google Cloud Storage onto our virtual machine, and we wrote Python code to combine it into a NumPy array, where each image was labeled with a 0 or 1 depending on whether or not it was an orchard. We then split the data into train, validation and test datasets and created corresponding data loaders using Pytorch in order to feed the data in small batches into our network. These data loaders were used as input for both our baseline model and our ResNet model.

5. Experiment and results

5.1. Baseline model results

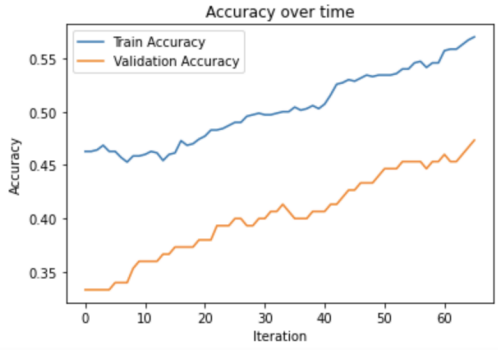

Figure 4. Baseline model results

This model has far too few parameters to adequately learn the data. From the way the accuracies rise continuously, it looks like the model is underfitting the data. By running the model for a few more epochs, we could probably continue to increase the validation accuracy, but given the limited resources of the CPU, we decided to stop training here. The training accuracy was around 55% while the validation accuracy was around 45%.

5.2. ResNet model results

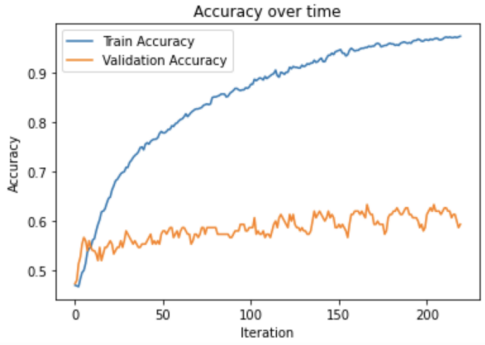

Figure 5. ResNet on Zoom Level 16

In this model, the spread between the train and validation accuracies tells us that the model is clearly overfitting the data. We tuned the weight regularization parameter, as mentioned in the Method section, and we also tuned the number of layers in the ResNet that were allowed to be changed during training, and we found the values used here to result in the highest validation accuracy. The training accuracy was around 97% while the validation accuracy was around 60%. We felt that on a test dataset size of about 200, 60% validation accuracy was good enough to indicate that our model's performance was better than random, but we were hoping to do a bit better.

5.3. Saliency map

In order to determine why our model was not performing as well as we expected, we created saliency maps, which highlight the pixels on each image that were most important in the model's decision. Specifically, it highlights the pixels where at least one of the color values has a large positive or negative gradient with respect to the output. Two examples of the saliency maps produced are given below.

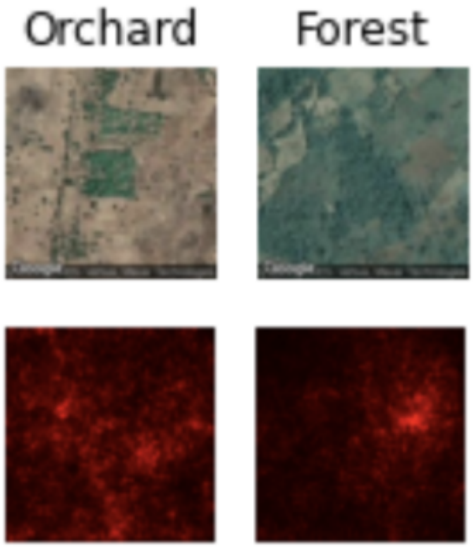

Figure 6. Saliency Maps for Zoom Level 16

We saw that in these images, the plants were too small for the model to determine whether they were crops in an orchard or trees in a forest. Instead, judging by the saliency maps that we looked through, it seemed like the model was looking at the background of the images and determining whether they were more brown or more green. If they were more brown, it classified them as orchards, and if they were more green, it classified them as forests. This heuristic doesn't generalize well to this dataset, which is evidenced by the 60% validation accuracy, and we were afraid it would not generalize well outside of our dataset. We were hoping that the model would start to pick up on the shapes of plants in an orchard. Thus, we decided to rerun the model, but this time on a more zoomed-in version of the images.

5.4. Increased Zoom level

With the help of Bruno Lopez, we downloaded new images from the same locations, but this time using OpenStreetMaps's 19-level zoom, instead of 16-level zoom. When we ran the model, we were happy to see that the validation accuracy was much better.

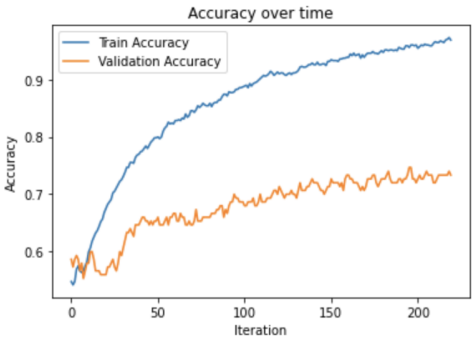

Figure 7. ResNet on Zoom Level 19

For images with zoom level 19, the training accuracy was around 96% while the validation accuracy was around 73%. This was a marked increase from the original model.

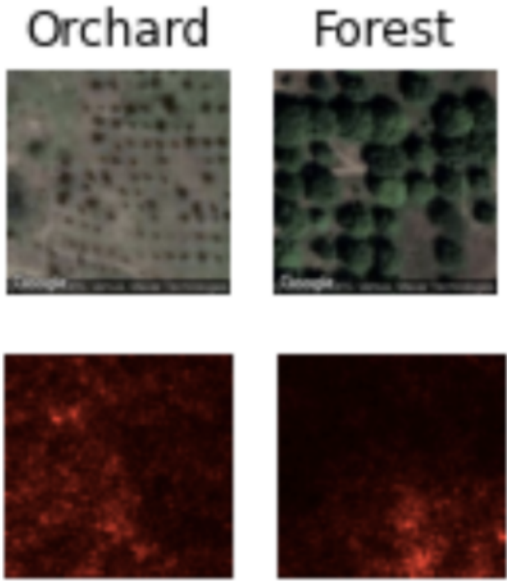

Figure 8. Saliency Maps for Zoom Level 19

We expected that at this higher zoom level, the model would have made its decisions based on the "palm" shape of orchard plants and the circular shape of forest plants. However, this was still not the case. In the saliency maps above, you can see that the background is still more important than the foreground. For example, in the forest image, there are two large highlights towards the bottom of the image, where there is a gap in the trees. Either the color of the background, or just the pattern of the missing trees, is contributing heavily to the model's decision.

5.5. Testing accuracies

With our hyperparameters tuned and our zoom level set, we decided to run the model on our test dataset in order to report our final results. Fortunately, our test accuracies were even better than our validation accuracies.

For our baseline model, the final test accuracy of our baseline model was 55.33%. The final test accuracy of our ResNet18 model was 70.00% on images with zoom level 16, and the final test accuracy of our ResNet18 model was 78.67% on images with zoom level 19.

6. Discussion and future work *1-3 paragraphs*

While the initial purpose of the model was to classify orchards vs. forests, we decided to also run the ResNet 18 model on a more general dataset containing both developed land and orchards. We attempted to classify orchards versus developed land (commercial, residential, retail, etc.) The results are shown below.

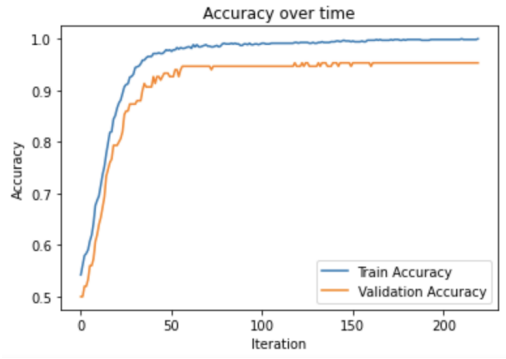

Figure 9. Orchard vs. Developed Images

We were blown away by the results on this dataset. Not only was the training accuracy around 99% and the validation accuracy around 95%, but the test accuracy was 98%. Since we chose to classify orchards against developed lands, we noticed that it was much easier for our model to differentiate between the two. There are significant differences that were unique to the agricultural characteristics of orchards vs. the artificial characteristics of the developed land. To highlight this difference, we give an example of an orchard and a plot of developed land below.

Figure 10. Orchard vs. Developed

As demonstrated above, the images look significantly

different, especially due to to the absence of agricultural characteristics within the developed images. We also generated saliency maps for this new dataset in order to try to understand why the model was able to perform so well.

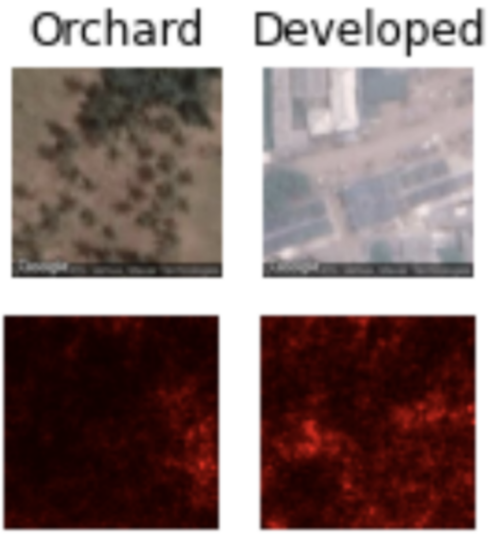

Figure 11. Orchard vs. Developed Saliency Map

It seems that the model is highlighting the untouched brown space on the right side of the orchard. In developed areas, there tends to be less untouched dirt. The model also seems to be highlighting the large rectangular building in the developed image.

One benefit of having our model perform well on different types of data is that TechnoServe, the nonprofit, can now use our model on a more general set of images, instead of having to first filter to just forests and orchards. By verifying that our model can be generalized to another situation, we increase the likelihood that our model has real world impacts for farmers in West Africa. In the future, we hope to run this model on even more different types of data, including datasets that combine orchards, forests and developed land all together.

In the future, we also hope to try a multi classification approach rather than just a binary classification one. We would like to classify not just orchards against forests, or orchards against developed lands, but also classify different types of orchards, segmented by crop type.

We hope that our work can be a good starting point for further work in the utilization of more complex models to achieve social good.

7. Contributions and Acknowledgments

Eric researched the problem and guided our strategy by looking at related works. Josh created the baseline model and the original CNN. Solomon ran the model on a variety of data and created all of the saliency maps.

Our group would like to thank Professor David Lobell for introducing us to this problem, providing us with our data, and mentoring us throughout this process. We would like to thank Bruno Lopez for helping us download the data and for helping us analyze the results of our saliency maps. We would also like to thank our TA, Guanzhi Wang, for helping us to prioritize among various potential tasks in the final weeks of our project.

References

[1] Food insecurity crisis mounting in africa. https://africacenter.org/spotlight/food-insecurity-crisis-mounting-africa/, Feb 2021. 1

[2] Openstreetmap. https://wiki.openstreetmap.org/wiki/API, May 2021. 3

[3] Selim Aksoy, Ismet Zeki Yalniz, and Kadim Tasdemir. Automatic detection and segmentation of orchards using very high resolution imagery. *IEEE Transactions on Geoscience and Remote Sensing*, 50(8):3117–3131, 2012. 1

[4] Pengyu Chu, Zhaojian Li, Kyle Lammers, Renfu Lu, and Xiaoming Liu. Deepapple: Deep learning-based apple detection using a suppression mask r-cnn. *arXiv preprint arXiv:2010.09870*, 2020. 1

[5] Longsheng Fu, Yaqoob Majeed, Xin Zhang, Manoj Karkee, and Qin Zhang. Faster r–cnn–based apple detection in dense-foliage fruiting-wall trees using rgb and depth features for robotic harvesting. *Biosystems Engineering*, 197:245–256, 2020. 1

[6] Weikuan Jia, Yuyu Tian, Rong Luo, Zhonghua Zhang, Jian Lian, and Yuanjie Zheng. Detection and segmentation of overlapped fruits based on optimized mask r-cnn application in apple harvesting robot. *Computers and Electronics in Agriculture*, 172:105380, 2020. 1

[7] Teja Kattenborn, Jens Leitloff, Felix Schiefer, and Stefan Hinz. Review on convolutional neural networks (cnn) in vegetation remote sensing. *ISPRS Journal of Photogrammetry and Remote Sensing*, 173:24–49, 2021. 1

[8] Lucas Prado Osco, Mauro dos Santos de Arruda, Diogo Nunes Gonçalves, Alexandre Dias, Juliana Batistoti, Mauricio de Souza, Felipe David Georges Gomes, Ana Paula Marques Ramos, Lúcio André de Castro Jorge, Veraldo Liesenberg, et al. A cnn approach to simultaneously count plants and detect plantation-rows from uav imagery. *ISPRS Journal of Photogrammetry and Remote Sensing*, 174:1–17, 2021. 2

[9] Pablo Ruiz. Understanding and visualizing resnets. https://towardsdatascience.com/understanding-and-visualizing-resnets-442284831be8, Apr 2019. 2

[10] Koo Ping Shung. Accuracy, precision, recall or f1? https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9, Apr 2020. 3

[11] Yunong Tian, Guodong Yang, Zhe Wang, En Li, and Zize Liang. Detection of apple lesions in orchards based on deep learning methods of cyclegan and yolov3-dense. *Journal of Sensors*, 2019, 2019. 2

[12] Yijie Wang, Jidong Lv, Liming Xu, Yuwan Gu, Ling Zou, and Zhenghua Ma. A segmentation method for waxberry image under orchard environment. *Scientia Horticulturae*, 266:109309, 2020. 1

[13] Liheng Zhong, Lina Hu, and Hang Zhou. Deep learning based multi-temporal crop classification. *Remote sensing of environment*, 221:430–443, 2019. 1

[14] Maciel Zortea, Maysa MG Macedo, A Britto Mattos, Bernardo C Ruga, and Bruno H Gemignani. Automatic citrus tree detection from uav images based on convolutional neural networks. In *2018 31th SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*, volume 11, 2018. 2