Team 33 Project 2.2 Final Report: Cotton Fiber Tip Detection

Krishna Murali (kmurali2), Mit Pandya (mpandya), James Rotbert (jgrotbert)

I. INTRODUCTION

Cotton plays a critical role in today's textile industry. At NC State, the Department of Crop and Soil Sciences is attempting to better understand how cotton fibers form, so that they may control and manipulate these structures in ways that will be useful for the textile industry.

The main goal of this project is to train a neural network so that it identifies cotton tips from microscopic images of cotton strands. The microscopic images originate from two different types of cotton fibers. The first, Gossypium Barbadense, has only 1 type of tip and is considered to be a better quality fiber. The second type, Gossypium Hirsutum, has two types of tips, and is a more productive crop.

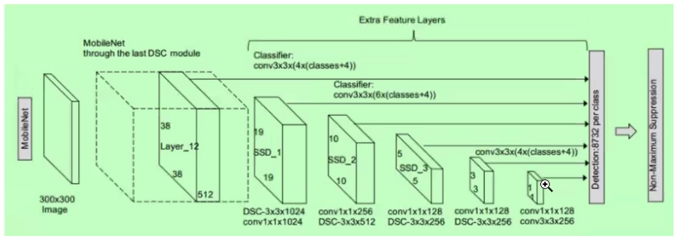
This project is an amalgamation of two tasks. The first task involves manually marking cotton tips in labelbox in order to have a reliable training and testing dataset. The second task is to use these labelled images in order to develop a neural network model. The input dataset for this project will consist of one thousand or more different fiber tip locations within an x and y coordinate plane of an image, along with labels of where in the image the tips are located from Labelbox. The dataset the team used to train the neural network consists of 450 labeled images.

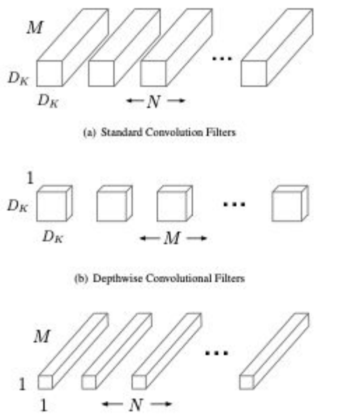
II. METHODOLOGY

The goal of this project was to train a neural network to predict locations of a cotton fiber tip within an image on unknown data. To accomplish this, the Faster R-CNN, R-FCN, and SSD models were initially selected as strong candidates. The team implemented a pre-compiled Tensorflow SSD (Single Short Detector) mobilenet architecture model and RetinaNet50 model and then chose the model which gave the best results. The RetinaNet50 model is pre-trained with the *Common Objects in Context (COCO 2017)* dataset. Using this model, the neural network was successfully able to output the coordinates of a box that bounded each identified cotton tip.

The metrics of success for this model will be primarily to minimize loss and maximize accuracy, while staying within the bounds of available computational capacity. For example, Tensorflow offers a wide variety of SSD models, yet a good portion of the models required more computational power than our local computers could withstand (Training was done using Nvidia RTX 2060 -- CUDA 10.1, tensorflow 2.3.1). The mobilenet model and RetinaNet50 were selected because the stats provided by Tensorflow suggested these routes would yield the highest accuracy while allowing training to occur on the team members' local machine.

The MobileNet model is based on depth-wise separable convolutions which is a form of factorized convolutions which factorize a standard convolution into a depthwise convolution and a 1×1 convolution called a pointwise convolution. For MobileNets the depthwise convolution applies a single filter to each input channel. The pointwise convolution then applies a 1×1 convolution to combine the outputs to the depthwise convolution. A standard convolution both filters and combines inputs into a new set of outputs in one step. The depthwise separable convolution splits this into two separate layers, one layer for filtering and one for combining. This factorization has the effect of drastically reducing computation and model size.



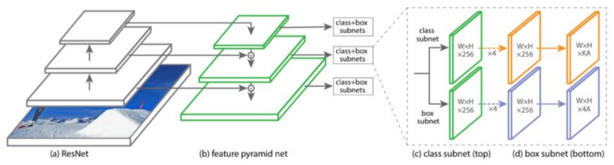


(a) Standard Convolution Filters

(b) Depthwise Convolutional Filters

(c) 1 × 1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

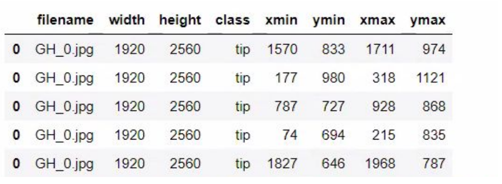
RetinaNet is a network comprising a backbone network and two task-specific subnetworks. The backbone, a standard convolutional network, generates a feature-map over the entire image that is inputted. Of the two subnetworks, the first performs object classification on the output from the backbone network, and the second performs convolutional regression on bounding boxes. The backbone network generates a multi-scale feature pyramid. This means that each layer is used for objection detection on an image that is at different scales from the original image inputted. This is referred to as a Feature Pyramid Network (FPN).



In order to use this model, several sensitive and critical steps were taken to ensure the inputs to this model matched the criteria needed for this model to function successfully.

The first step was to separate the training data into training files and validation files. 5 files (GH_430 to GH_434) were selected at random for validation. The rest would be used for training.

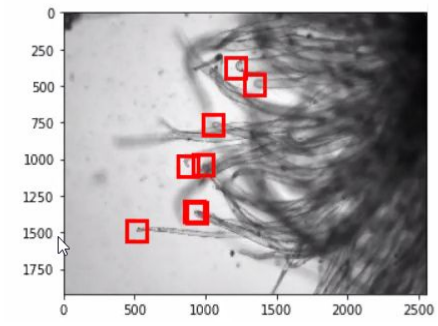
Then, the team needed to generate a CSV file that provided details about the location of each cotton tip in each image. Tensorflow requires a specific format for this CSV file, shown below:



	filename	width	height	class	xmin	ymin	xmax	ymax
0	GH_0.jpg	1920	2560	tip	1570	833	1711	974
0	GH_0.jpg	1920	2560	tip	177	980	318	1121
0	GH_0.jpg	1920	2560	tip	787	727	928	868
0	GH_0.jpg	1920	2560	tip	74	694	215	835
0	GH_0.jpg	1920	2560	tip	1827	646	1968	787

The width and height parameters represent the overall dimensions of the image, while the xmin, ymin, xmax, and ymax variables represent the bounds of the box where a cotton tip exists in the training data.

Hence, the CSV file and training images were converted to a Tensorflow dataset with the file extension, ".record". The process was repeated for the validation files. These files served as an input to the Tensorflow model. A ".pbtxt" file is created with the id and name of the class. Below is what a standard image from the training dataset looked like. Note the location of the tips are outlined with a red box.

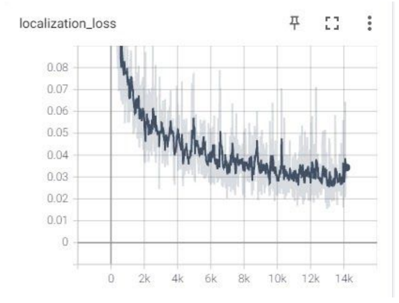
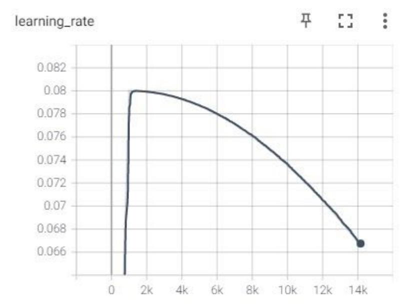


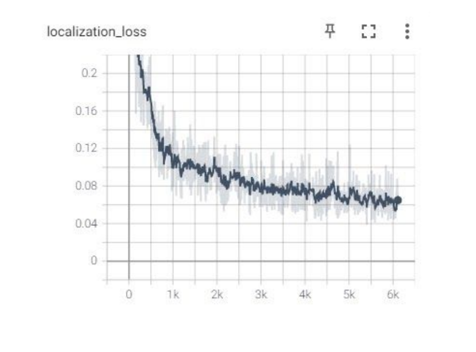
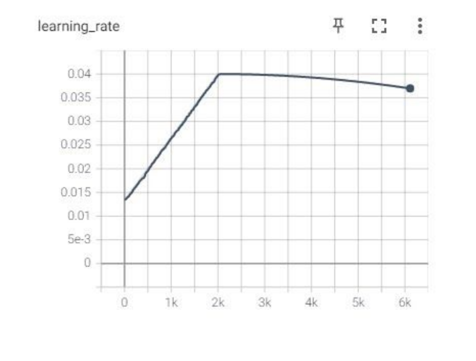
At this point, training could begin. During training, the model generates a checkpoint file at every 1000 time steps. These checkpoints act as recovery points, where the team could retrieve the evolving versions of the model as it continued to train. Because the team used an off-the-shelf, pre-trained model, the code needed to be converted to a custom object detector which trains on the cotton tip images.

After training, the results were saved to the local environment and the images could be retrieved.

III. Evaluation

The graphs below show the localization loss and the learning rate on the training data for the SSD mobilenet model and the retinanet50 model.

SSD MobileNet Results: Localization Loss	
SSD MobileNet Results: Learning Rate	

RetinaNet50 Results: Localization Loss	
RetinaNet50 Results: Learning Rate	

We evaluated our model performance using Mean Average Precision and Average Recall. The Mean Average Precision averages the average precision over all the N classes.Average recall describes the area doubled under the Recall x IoU curve. The Recall x IoU curve plots recall results for each IoU threshold where IoU ∈ [0.5,1.0], with IoU thresholds on the x-axis and recall on the y-axis.

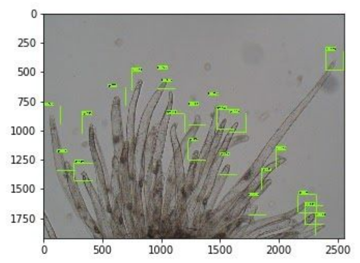
$$mAP = \frac{1}{N} \sum_{i=1}^{N} AP_{\text{i}}$$

$$AR = 2 \int_{0.5}^{1} recall(IoU)\, \text{dIoU}$$

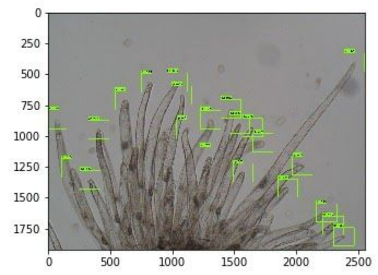
Results:

Type of Model	mAP	AR
SSD MobileNet	0.452	0.52
RetinaNet50	0.417	0.507

RetinaNet Prediction on test image GH_36:



SSD MobileNet Prediction on test image GH_36:



While the team was able to build a model that could identify tips, the model certainly was not able to identify all of the tips in each of the datasets. Higher mAP scores can be achieved with more complex NN structures. This of course requires more advanced computational power. Results of SSD MobileNet and RetinaNet50 are comparable but MobileNet performs slightly better compared to RetinaNet50.

IV. References

A. Farooq, M. Sarwar, M. Ashraf, “Predicting Cotton Fibre Maturity by Using Artificial Neural Network” Autex Research Journal, 2018

Y. Jiang C. Li, “Convolutional Neural Networks for Image-Based High-Throughput Plant Phenotyping: A Review”, Science Partner Journal, 2020

T. Lin, P. Goyal, R. Girshik, K. He, P. Dollar, “Focal Loss for Dense Object Detection”, Facebook AI Research (FAIR), 2018

Andrew G. Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, Hartwig Adam, Google Inc, 2017, “MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications”