RCOS Report - Fall 2018

Rahul Puppala - ImageJ Thresholding with TensorFlow

Overview of Report

My idea to join RCOS stemmed from two factors: my deep interest for learning things about Computer Science and my dream to apply my biology skills to a meaningful coding project. Thus I took on the project that uses the power of machine learning to make ImageJ Quantifications faster, as well as more accurate. This report will go over the following information related to my work this semester:

1. Review and Progress Report on Goals
2. Research Conducted
    a. Background on Thresholding/Quantifications
    b. Technology Research
        i. Machine Learning and Deep Learning
        ii. TensorFlow/Keras
        iii. Image Analysis Tech
        iv. Similar Projects
3. My Attempt at the Project
4. Future Work

1. Review and Progress Report on Goals

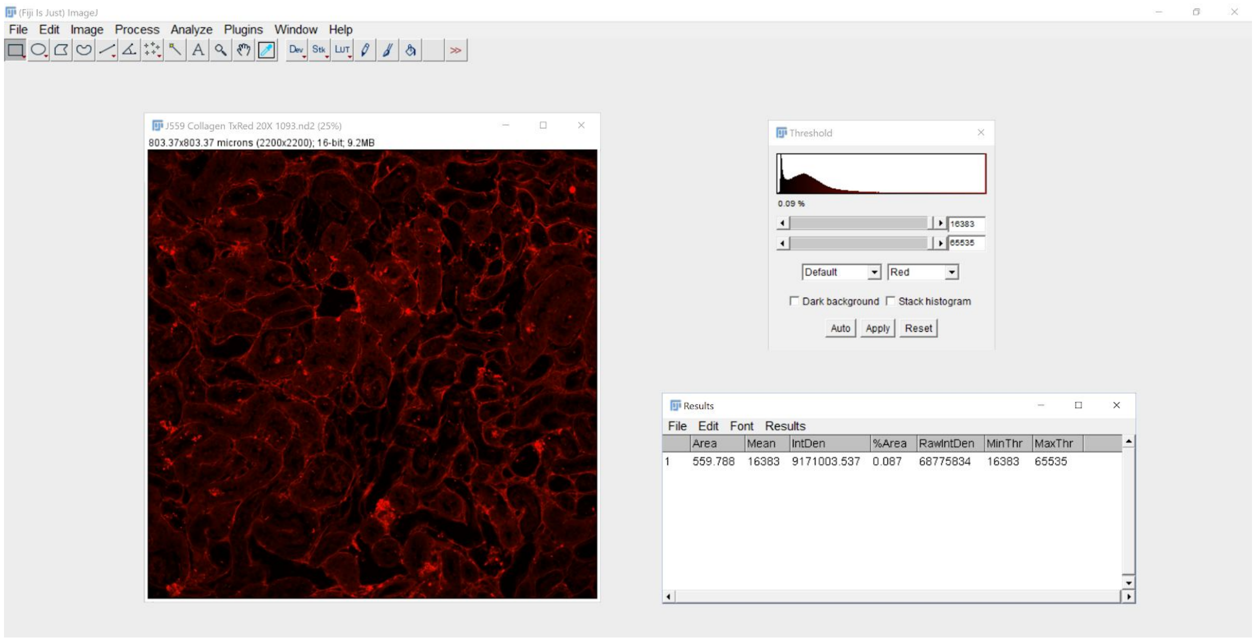
The primary goal for this project was to allow me to get accustomed with Machine Learning, and I have to say I have learned a lot about the subject. In later sections of this report, you will see a summarized account of all the research I did on the subject.

My secondary goal was to build a Machine Learning model that would increase the speed and accuracy of quantifying biological images. While I attempted this goal, the way I structured my model set me up to not succeed. I learned this the hard way quite late into the semester. I believe that this was a huge project for a single individual and with a good, detailed plan, this project could be very successful with the help of future RCOS teammates.

2. Research Conducted

Background on Thresholding and Quantifications

In the summer of 2017, I worked in a Nephrology Research Lab at Stony Brook University. One of the tasks I performed there was to take hundreds of pictures of kidney cells and analyze them. More specifically, I would open each image up on my computer through a program called ImageJ, run a macro script on the image, and calculate what percent area of the image was a certain color. This color would be directly proportional to the amount of collagen fibers are in the kidney cell. The amount of collagen fibers in the kidney cell would then indicate the severity of injury to that particular kidney. Since one image is not exactly representative of the whole kidney and there is a degree of bias when quantifying images, I would take hundreds of images and quantify them all. This is a lengthy process and more importantly, it is very prone to bias. A researcher who wants their experiment to work will more likely see different levels of collagen in kidney cells than a unbiased 3rd party who has no vested interest in the result of the quantification. Therefore I proposed and idea to have a machine learning model that is trained to identify a good image and adjust the levels of thresholding to an unbiased level.

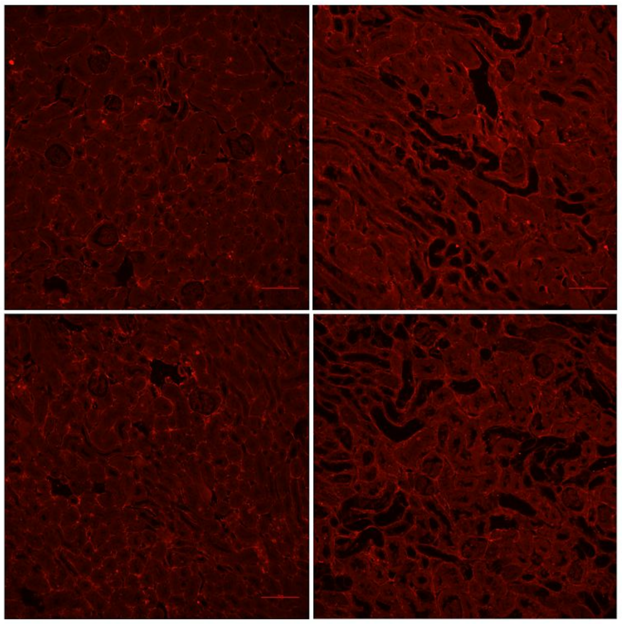


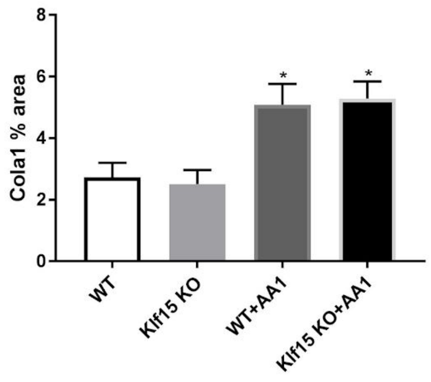
J559 Collagen TxRed 20X 1093.nd2 (25%)
803.37x803.37 microns (2200x2200); 16-bit; 9.2MB

0.09 %
16383
65535
Default
Red
Dark background
Stack histogram

	Area	Mean	IntDen	%Area	RawIntDen	MinThr	MaxThr
1	559.788	16383	9171003.537	0.087	68775834	16383	65535

The image above is a picture of a kidney being analyzed in the ImageJ program. It is currently measuring the % area of the picture that is at or above a certain pixel color threshold. This is how researchers quantitatively analyze the pictures they take. The crux of the issue here is the threshold box on the right. Depending on the conditions of the microscope slide, as well as the image capturer, the threshold needs to be adjusted to properly analyze the picture.





This picture above is the final summary of all the hundreds of quantified images. All the images in a single category are averaged and then each of the 4 categories are put in a statistical ANOVA test to determine significant difference. Obviously from our eyes we are able to tell that there is a visible increase in red borders, however this does not fly on a scientific paper. The goal of this project was to build a model that did the entire quantification process quicker and more accurately.

Technology Research

Machine Learning

Machine learning algorithms are described as learning a target function (f) that best maps input variables (X) to an output variable (Y): Y = f(X)
The goal of Predictive modeling is to make predictions in the future (Y) given new values of input variables (X). We don’t know what the function (f) looks like or its form so we need to learn it from input data(also referred to as training data when used in this way) using machine learning algorithms. The function (f) once learned would become our prediction model. Following are some of the techniques used in creating such a model.

1) Linear Regression

The representation of linear regression is an equation that describes a line that best fits the relationship between the input variables (x) and the output variables (y), by finding specific weightings for the input variables called coefficients (B0 & B1) in the function y = B0 + B1 * x. This can be done using algebraic technique like least mean squares or a gradient descent optimization.

2) Logistic regression

The logistic regression uses an nonlinear function that denotes a S shaped curve to categorize input values into a binary 0 or 1. It is useful for solving binary classification problems.

3) Naive Bayes

The model is comprised of two types of probabilities that can be calculated directly from your training data: 1) The probability of each class; and 2) The conditional probability for each class given each x value. Once calculated, the probability model can be used to make predictions for new data using Naive Bayes Theorem. This approach assumes the

4) K-Nearest Neighbours

Predictions are made for a new data point by searching through the entire training set for the euclidean distances between neighbours and finding the K closest neighbors. The output can be summarized as the output mean or class for those K instances.

5) Random Forests

A decision tree is a Machine Learning algorithm capable of fitting complex datasets and performing both classification and regression tasks. The idea behind a tree is to search for a pair of variable-value within the training set and split it in such a way that will generate the "best" two child subsets. The goal is to create branches and leafs based on an optimal splitting criteria, a process called tree growing. Specifically, at every branch or node, a conditional statement classifies the data point based on a fixed threshold in a specific variable, therefore splitting the data. To make predictions, every new instance starts in the root node (top of the tree) and moves along the branches until it reaches a leaf node where no further branching is possible.

Trees have a high risk of overfitting the training data as well as becoming computationally complex if they are not constrained and regularized properly during the growing stage. In order to average out the outcome of individual predictions by diversifying the set of predictors, thus lowering the variance, random forests use a ensemble of decision trees to arrive at a powerful prediction model that reduces overfitting our training set.

6) Support Vector Machines

SVM uses the technique of selecting a hyperplane(a line in 2 Dimensional space) with the coefficients that results in the best separation of the input variable space by their class, either class 0 or class 1. The hyperplane is selected based on the largest distance between the hyperplane and the closest data points

Supervised Vs Unsupervised:

In supervised learning, we use input values of input-output pairs that are manually classified or we know are correct that we can present to the algorithm during the training phase. For deep learning a lot of preprocessed images classified by humans will increase the accuracy of learning.

If the “right answers” are unobservable, or infeasible to obtain, or maybe for a given problem, there isn’t even a “right answer” then unsupervised learning has the best response.

Deep Learning

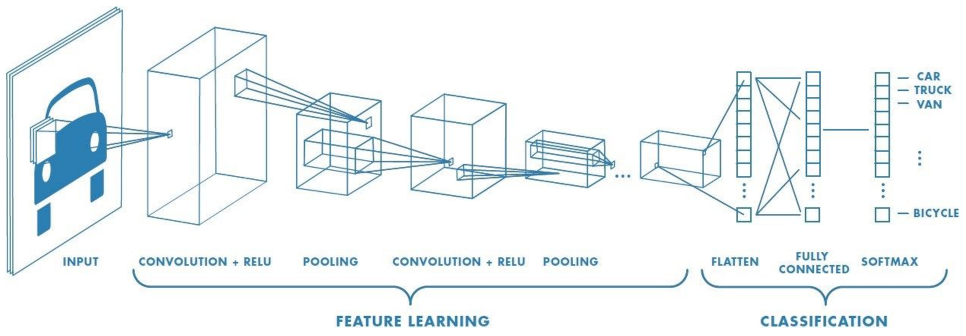
Deep learning is a class of machine learning algorithms that:

- use a cascade of multiple layers of nonlinear processing units for feature extraction and transformation. Each successive layer uses the output from the previous layer as input.
- learn in supervised (e.g., classification) and/or unsupervised (e.g., pattern analysis) manners.
- learn multiple levels of representations that correspond to different levels of abstraction; the levels form a hierarchy of concepts.

Deep learning models are based on an artificial neural network organized layer-wise in deep generative models such as the nodes in deep belief networks and deep Boltzmann machines. In deep learning, each level learns to transform its input data into a slightly more abstract and composite representation. In an image recognition application, if the raw input is a matrix of pixels; the first representational layer may abstract the pixels and encode edges; the second layer may compose and encode arrangements of edges; the third layer may encode a nose and eyes; and the fourth layer may recognize that the image contains a face. Importantly, a deep learning process can learn which features to optimally place in which level on its own. (Of course, this does not completely obviate the need for hand-tuning; for example, varying numbers of layers and layer sizes can provide different degrees of abstraction. The "deep" in "deep learning" refers to the number of layers through which the data is transformed.

Convolutional Neural Network

CNN image classifications takes an input image, process it and classify it under certain categories. Computers sees an input image as array of pixels and it depends on the image resolution. Based on the image resolution, it will see h x w x d(h = Height, w = Width, d = Dimension). Eg., An image of 6 x 6 x 3 array of matrix of RGB (3 refers to RGB values) and an image of 4 x 4 x 1 array of matrix of grayscale image.
Technically, deep learning CNN models to train and test, each input image will pass it through a series of convolution layers with filters (Kernals), Pooling, fully connected layers (FC) and apply Softmax function to classify areas of an image with probabilistic values between 0 and 1. The below figure is a complete flow of CNN to process an input image and classifies these areas based on values.

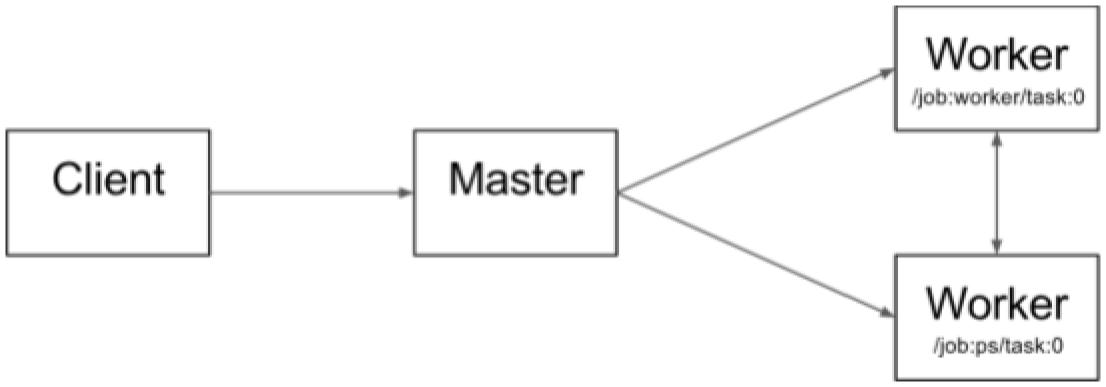


TensorFlow Research

TensorFlow (https://www.tensorflow.org/) is a software library, developed by Google for the purposes of conducting machine learning and deep neural network research. TensorFlow combines the computational algebra of compilation optimization techniques, making easy the calculation of many mathematical expressions where the problem is the time required to perform the computation.
The main features of tensorflow include:

- Defining, optimizing, and efficiently calculating mathematical expressions involving multi-dimensional arrays (tensors).
- Programming support of deep neural networks and machine learning techniques.
- Transparent use of GPU computing, automating management and optimization of the same memory and the data used. You can write the same code and run it either on CPUs or GPUs. More specifically, TensorFlow will figure out which parts of the computation should be moved to the GPU.
- High scalability of computation across machines and huge data sets.

Tensorflow consist of a client API in python and C++ , distributed master node and multiple worker services.



Client
- Defines the computation as a dataflow graph.
- Initiates graph execution using a session.

Distributed Master
- Prunes a specific subgraph from the graph, as defined by the arguments to Session.run().
- Partitions the subgraph into multiple pieces that run in different processes and devices.
- Distributes the graph pieces to worker services.
- Initiates graph piece execution by worker services.

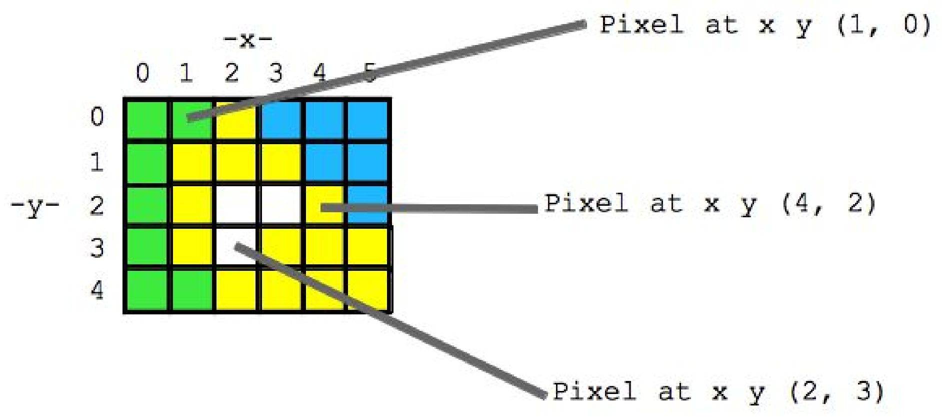
Worker Services (one for each task)

    - Schedule the execution of graph operations using kernel implementations appropriate to the available hardware (CPUs, GPUs, etc).
    - Send and receive operation results to and from other worker services.

This architecture enables very efficient, scalable distribution and execution of tensors(matrices) specified in a high level client API by the workers.

Image Analysis Tools

An image is a two dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x,y) where x and y are the two co-ordinates horizontally and vertically. The value of f(x,y) at any point is gives the pixel value at that point of an image, the pixel value describes how bright that pixel is, and/or what color it should be.
For grayscale images the pixel value is a single number that represents the brightness of that pixel, the most common pixel format is the byte image, which is stored as an 8-bit integer giving a range of possible values from 0 to 255. As a convention is taken to be black, and 255 is taken to be white the values in between make up the different shades of gray.
To represent color images, separate red, green and blue components must be specified for each pixel (assuming a RGB color model), and so the pixel `value' becomes a vector of three numbers. Often the three different components are stored as three separate `grayscale' images known as color planes (one for each of red, green and blue), which have to be recombined when displaying or processing.



Representation of an image as a matrix

Image processing and analysis involves tasks that manipulate the images represented as matrices and could be categorized as follows :

- Image acquisition, storage, transmission: digitization/quantization, compression, encoding/decoding.
- Image Enhancement and Restoration: for improvement of pictorial information.

- Information Extraction: for further computer analysis .

The libraries describes below provide the basic building block routines that support the above tasks.

OpenCV

OpenCV (Open source computer vision) is a library of programming functions mainly aimed at real-time computer vision and has a C++, Python and Java interfaces. OpenCV was designed for computational efficiency and with a strong focus on real-time applications. Written in optimized C/C++, the library can take advantage of multi-core processing. Enabled with OpenCL, it can take advantage of the hardware acceleration of the underlying heterogeneous compute platform. OpenCV provides a library of routines that can help in some of the Image analysis operations like :

- Applying linear algebra on images
- Image Normalization
- Video capture
- Histogram Equalization
- morphological transformations like Erosion and Dilation
- Extract R,G,B layers and applying binary thresholding of pixels
- Object/Feature detection and taking contours to detect the areas with common features
- Image Denoising
- Blob detection
- Finding contours of the filtered image.
- Gaussian blur and find local maxima on the image.

OpenCV includes a statistical machine learning library that contains:

- Boosting
- Decision tree learning
- Gradient boosting trees
- Expectation-maximization algorithm
- k-nearest neighbor algorithm
- Naive Bayes classifier
- Artificial neural networks
- Random forest
- Support vector machine (SVM)
- Deep neural networks (DNN)

ImageJ

ImageJ is a Java-based image processing program. Its first version, ImageJ 1.x, its newer version ImageJ2 and the related projects SciJava, ImgLib2 and SCIFIO are designed with an open architecture that provides extensibility via Java plugins and recordable macros that support Custom Image acquisition, analysis and processing plugins. User-written plugins make it a popular choice to solve many image processing and analysis problems like. One big advantage it has for my project is that it has integrations with TensorFlow.

- three-dimensional live-cell imaging
- radiological image processing
- multiple imaging system data comparisons
- automated hematology systems.

Similar Projects

After spending a semester looking through documentation online of similar projects, I have identified two projects that are very similar to the take I set out to do. These are namely:

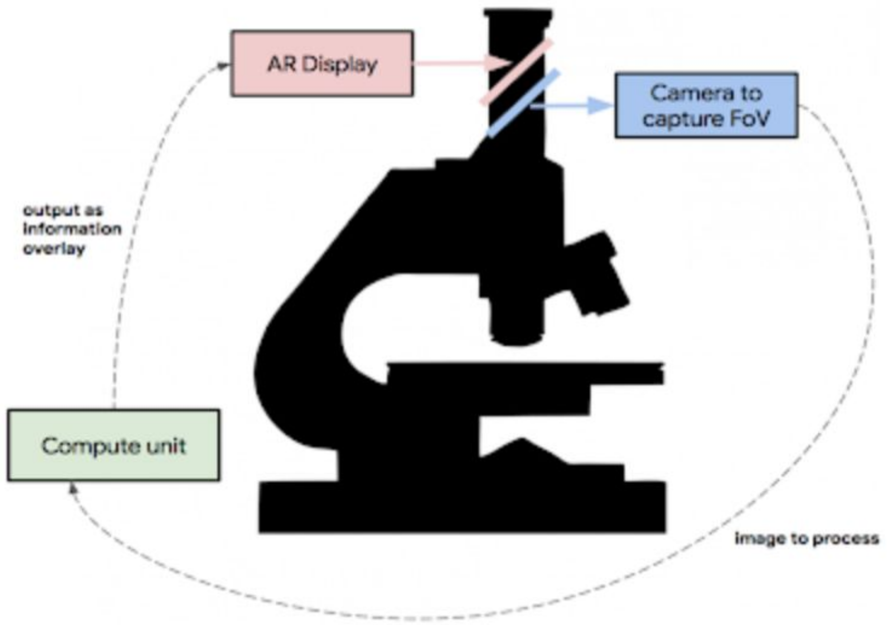
1. Google's Augmented Reality Microscope for Cancer Detection
2. Microsoft/Arccos' Virtual Caddie App powered by Machine Learning

Google's AI Microscope

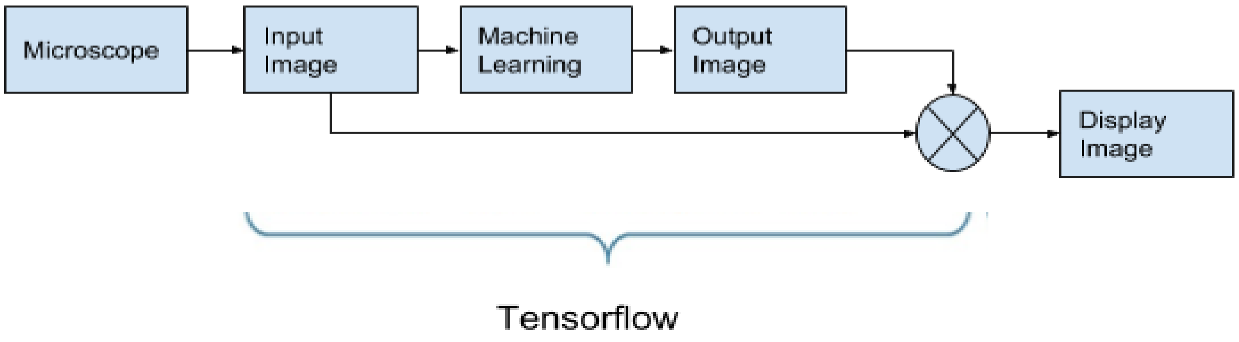
Advances in microscope technology like line-scanned dual-axis confocal microscope are helping researchers demonstrate that these miniature microscopes have sufficient resolution to see subcellular details comparable with those produced at a clinical pathology lab for identifying cancerous cells in tissues.

Advances in Artificial Intelligence technologies like deep learning have successfully applied to cell biology in particular detection of various aspects of cancer detection with accuracies nearing or exceeding that of human experts.

Combining the above two technologies with the high resolution displays available ubiquitously, one can process microscope originated images using algorithms and visualize them in real time to prove machine assistance to researchers. Google has demonstrated this by building a augmented reality microscope.



With the above shown setup images collected from a microscope can be analyzed live by artificial intelligence software using the tensorflow framework and overlaid with information that can assist human operator.



This project uses Tensorflow and ImageJ in real-time to augment the image that a researcher sees in a microscope. This is very similar to my project because I too am using ImageJ to analyze the images and TensorFlow to run my Machine Learning algorithms. Although this project uses live images and not statics ones, it will still be an excellent place to start understanding how to implement my project because of the similar technologies used, as well as the projects image processing techniques. This project is able to recognize structures in cells and that is exactly what I would want to do, so that I can get a correct threshold for my images.

Microsoft and Arccos' Machine Learning Virtual Caddie

This project will be very useful for me but in a different way. Even though it has nothing to do with cells or biology, this project has a very similar method of analyzing pictures in the form of thresholding. What this project basically does is that they built a machine learning model that can take new input images of golf courses and glean key golfing data from them through image analysis. Then they combine all the info they have gathered into an awesome app that is basically a virtual caddy. The part of this project that is interesting to me is the fact that they were able to successfully threshold a desired section of an image based on a machine learning model of training images. Although they used different technologies (OpenCV and Python), analyzing their approach to integrating machine learning and image analysis will allow me to develop my own method.

3. My Attempt at the Project

I came into this project with no knowledge of Machine Learning so a majority of this semester was spent researching every aspect of this project. However, there is a saying that you don't truly learn something until you actually do it yourself. This experiential learning taught me some other things as well. After about 2 months of research, I started playing with TensorFlow. I also listened to all introductory guides' advice and got Keras. Keras is really good for rapid prototyping of models and allowed me to try different things quickly. I'll have to admit there were certain weeks during this semester where I was so overloaded with other work that I did not work on this RCOS project as much; however, I do think all the research I did above does show that I did make significant and consistent contributions to my work.

The biggest mistake I made when trying to build a model was choosing a unsupervised learning method. When I got into this project, I was of the opinion that TensorFlow was a magical program that would automate and learn the problem very easily on its own and all I really had to do was give it the right pictures. I was very mistaken. From this preconceived notion I went down the path of unsupervised learning. This type of machine learning is not the type I needed. Unsupervised learning is used when the output is not predetermined. In my case, I have a task that requires the program to learn a target function that maps an input variable (kidney images) to an continuous output variable (threshold level).

Another thing that I should have done with my model was to try and identify all the different factors that a human uses to identify what the correct threshold level is. I tried to do this myself near the end of the semester, but I am not a quantification expert and I would need to work with someone at the lab to analyze what parameters are highly correlated with proper thresholding. At the moment, the factors I believe to be the most important parameters for my model are: total number of edges detected, and rate of % area increase.

The total number of edges detected are a good factor to analyze because the highest amount of edges usually occurs at the optimal thresholding level. Also another thing that I noticed was that 10 to 15 units after you pass the correct threshold, the image % area counter goes up very quickly because of all the background color. This means that possibly writing a program incorporates the number of edges as well as the rate of change of the % area with regards to a shift in threshold could make for an optimal solution.

4. Future Work

This was a very ambitious project for a single person; and for next semester, I plan to try and continue it with a few more team members on board. With a highly detailed pipeline of the work and some dedicated team members, I believe that this can be a successful RCOS project. This project has much bigger scope that I initially started with. I am only tackling collagen at the moment because that is the type of cell image that I worked with during the summer. There are dozens of other types of images that can be quantified and the project would need to be moderately modified to accommodate those different cell structures.

The next steps for this project would be:

1. Creating a very detailed outline of the technology being used.
2. Modularizing the work so different people can jump on the project.
3. Talking to an expert quantifier to identify parameters that the algorithms should focus on.
4. Have each team member start their own approach to the problem so that

I see potential for this idea and look forward to continue working on it. Thank you to all the people in RCOS who helped me get as far I did.

Sources:

Machine Learning Algorithms
https://towardsdatascience.com/a-tour-of-the-top-10-algorithms-for-machine-learning-newbies-dde4edffae11

https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

CNN
https://medium.com/@RaghavPrabhu/understanding-of-convolutional-neural-network-cnn-deep-learning-99760835f148

Learn linear regression using tensorflow
https://medium.com/all-of-us-are-belong-to-machines/the-gentlest-introduction-to-tensorflow-248dc871a224
code is at
https://github.com/nethsix/gentle_tensorflow/blob/master/code/linear_regression_one_feature.py

CNN using tensorflow
https://github.com/udacity/dog-project

Random forest and Logistic regression
https://github.com/aymericdamien/TensorFlow-Examples

Image Analysis - OpenCV
https://medium.com/@ariesiitr/image-processing-with-opencv-45c3a5cefd10

https://github.com/imagej/imagej

ImageJ macro Cell Colony Edge
https://www.future-science.com/doi/full/10.2144/000114535

Machine learning and thresholding for golf course images
https://www.microsoft.com/developerblog/2018/05/17/using-otsus-method-generate-data-training-deep-learning-image-segmentation-models/

OpenCV guide
https://docs.opencv.org/3.0-beta/doc/py_tutorials/py_imgproc/py_table_of_contents_imgproc/py_table_of_contents_imgproc.html

Image Analysis
https://www.kdnuggets.com/2018/07/basic-image-data-analysis-numpy-opencv-p1.html
https://stackoverflow.com/questions/33729432/c-biological-cell-counting-with-opencv

Google's Augmented Reality Microscope for Cancer Detection
https://ai.googleblog.com/2018/04/an-augmented-reality-microscope.html