Rheumatoid Arthritis Measurement Application

Chase Denecke, Jared Tence, Ethan Patterson

CS 461 Fall 2018

CONTENTS

Section	Original	New
Goal Overview	Stated that our goal was to deploy a working app capable of measuring RA inflammation.	States that our goal is to make progress on developing the tools that will allow us to measure RA inflammation, specifically the data cleaning step, the joint detector, and the data collection app
Sample image collection	Evaluating noise for possible removal.	Noise definitely needing to be removed.
Next Steps if a Data Collection Assistant is Needed	Stated that our data collection app would be able to measure distance and angle in the image. No details about storing images or labelling data.	React native app. Minor grammar changes, removal of requirement that distance and direction be measured attributes. Addition of basic features like the ability to take pictures, label swollen joints, and store images in a database
Next Steps if a Data Collection Assistant is not Needed	No changes	
Collecting Real Data	Stated that we would advertise the data collection app on Facebook and undertake re-engagement campaigns	Stated that everything having to do with collecting real data was a stretch goal
Neural Network Architecture and Training	No mention of Mask_RCNN, no explanation that entire neural network for measuring inflammation is a stretch goal. Plan was to use a 1080 Ti graphics card to train our neural network on (which was never provided).	Added information about MASK_RCNN and AWS
App Development Outline	Included a lot of description of Xamarin and Progressive web apps	Includes description of React Native apps
Deployment	Contained many statements about things that will never happen	Includes many statements about things that will never happen and a disclaimer stating that they will probably never happen.
Neural Network Accuracy	Contained explanation of neural network accuracy measurements and error rates in classification	Replaced entire section with "Application" success. Defined application success in terms of a set of features and abilities we want the app to have.
Ease of Use	Contained explanation of intuitive UI description and how that would impact the user experience.	Replaced with a section on "Data Cleaning", in which we describe the proper performance of our data cleaning tool.
Utility of App	Contained a description of the ways in which we wanted the app to help patients	Replaced with section about "Collection of Lablled Data" in which we talked about how many labelled images we want to collect through Amazon Mechanical Turk.
Business Value of App	Contained a bunch of uselessly speculative writing about future business applications of our app	

I. Goal Overview

The goal of this project is to make progress towards creating an application that can track Rheumatoid Arthritis activity (RA) in patients through the use of image recognition using a patient's smartphone. Our product will be both a mobile app that may be used to collect data from patients in the future, the data preprocessing software that will allow us to reduce the noise in these images, and the data set of labelled images that will allow us to create a joint detector if we have time (creation of the actual joint detector This will allow KarateHealth to evaluate and track disease activity in an individual patient. If this proves possible treatment plans tailored for the individual can be made at a much lower cost compared to current methods of analysis.

II. Road Map

A. *Data Collection*

1) Sample Image Collection: The first step to build this application is to collect a sample image set. This sample image set will need to be validated, labeled, and cleaned. Because of noise in the images, such as background objects we will have to develop an application to assist in the image collection cleaning of this background noise.

2) Next Steps if a Data Collection Assistant is Needed: If an application like the one described above is needed, we will need to begin by determining if this application should be developed as a React Native application or a cross platform app using Xamarin. Both options will allow any user with a smartphone to use the application regardless of their operating system. This also means we will need to create clear requirements for this application regarding the features it needs to assist in the image taking process. These requirements will depend on what issues we see with the sample image set we are given if any. The application will need to access the phone's camera. It will also need an simple, somewhat intuitive UI with which users can label the joints in their hand that are swollen. Another feature it will need is the ability to store images taken by users on a server for future reference by us and by the patient.

3) Next Steps if a Data Collection Assistant is not Needed: If a data collecting assistant application is not needed we will proceed to the development of a neural network using Python with the TensorFlow and Keras API.

4) Collecting Real Data: This step will be one of the trickiest, thus we are treating it as a stretch goal with the expectation that we will not get to it. Karate Health will need to advertise the app on Facebook and get people to download the data collection application. Those people will then need to sign some type of e-consent form, agreeing to send those pictures to Karate Health after they are taken. It's also likely that we or Karate Health will need to create a re-engagement campaign, to re-target people who already agreed to take pictures at some later point in time. The idea of a re-engagement campaign is that people who have already taken pictures of their hands are more likely to agree to do so again, and the new data will be just as useful as that coming from a new user. The pictures will be stored in a secure database (whose location and interface is yet to be determined), from which we will retrieve the data when we are ready to train the neural network.

B. *Neural Network Architecture And Training*

The Neural Network for this project will be a fork of MASK_RCNN with minor changes. Using this framework will allow us to focus on collecting our custom data set and developing our data collection app. By using this fork we will not have to work on re-inventing the wheel by making another convolutional neural network with inferior performance to existing tools.

Creating a working network will require us to create two "filters" to detect features in the image. The first filter is the joint detector. This filter will simply locate joints within the image. This first filter will be a stretch goal for our network. The second filter will process the background data out of images collected by the phone application. This will leave only the hands in the image and therefore reduce the amount of data needed

We have already tested code for image classification using alternative data sets such as MNIST. However, to construct a score for patient's risk factor of flair up and treatment plan, we will need to train the network to identify sub features in the images such as distance between joints, swelling, surface area of the hand, and any subtle change in the overall structure of the hand. Modern neural networks extract these features from the image automatically, but we will need to understand more about how this process works to incorporate it effectively into our network.

Due to color being unreliable in the image set, due to issues such as white balancing done by the smartphones, lighting in the area, season, and diet we may need to gray scale the images before feeding it to the network. A process known as kernel image processing will be done to the image after any gray scaling has been applied, this will reduce much of the noise in the image. We will then proceed to flatten the image into a one-dimensional vector of pixel values. This pixel value vector will be feed into another vector of optimization functions, taking each vector value as an input. The output of each of these functions will then be fed to another optimization function. This will continue for the number of layers in the network. Once a final output is produced, known as the classification layer, this will be a score of possible flair risk. This flair risk can then be used to design a treatment plan for a patient.

During the training process, any error in this final output will be calculated and back-propagated to all the optimization functions in the network and the process starts over again. This will be done until the network reaches a point that it has little to no error. These optimization functions, each individually known as Support Vector Machines, are not known at this time. For use to create these optimization functions we will have to first evaluate the image set that we obtain so we can design and tweak these functions for the most optimal output.

Hyper-parameter optimization, preset variables for optimization, have not been determined at this time and will be investigated as we evaluate the data set. However, there have been recent papers discussing the use evolutionary methods for automating the discovery of these hyper-parameters. This will most likely be the best approach for this data set. The total image count required to reach our desired results is also not known at this time, but could range between 1,000 to 3,000 in total. We will know if we need more images once we train on the first image set. The general golden rule is at least 1,000 and originates from the ImageNet classification challenge. We will know that the neural network is learning by giving it a test. This process is referred to as cross validation. Cross validation is a reliable way to test that a neural network is actually learning and not just memorizing a problem. This is done by splitting your data set into segments of different proportions. We may do a 50% split on our data. One half will be used for training, and the other 50% that the network has never seen before will be used for testing. If the network performs well we know it is learning.

In the process of refining our architecture, we will be training our model on an EC2 instance using Amazon Web Services (AWS). We chose AWS because it allows us to easily install packages and create custom environments and it gives us access to high throughput hardware capable of significantly speeding up the training process of neural networks compared to our laptops.

*C. **App Development Outline***

The application we develop must work on both iOS and Android mobile devices. For this reason, we will make a React Native application. Using react native we will have two major requirements for our application. First is the ability for the user to submit RA data. The application has to be able to take a photo of the users hand, prompt and receive swollen joint information from the user pertaining to that hand photo, and send that information to a server. Second the application needs to be able to pull submissions from the server and display it to the user. When a user interacts with a submission the app should pull more details about that submission such as the submissions photo to show to the user. These requirements must be developed in a way that is easy to interact with the user and aesthetically pleasing. The user shouldn't be confused about the data collection and viewing process, In the case that a user is confused their should be able to view information on the purpose of a component.

*D. **App Server Outline***

The application server will go hand and hand with the development of the application. We will need the server http requests from the application. The server will need to service a POST request from the application, it will need to store the data in the POST request to a database as well as any photo files attached. The server will also have to service a GET request from the application in which it will respond with all submissions submitted by that applications users.

*E. **Deployment***

Deployment of any app on a wide scale is a stretch goal and will likely not be accomplished. However, if we do get around to completing this goal, here is what we belive it will look like.

Deployment of our app to at least hundreds, possibly thousands, or tens of thousands of users is more or less completely unknown to us. The more people downloading an app, the more potential there is for users to encounter corner cases that you never planned for. Given our knowledge of several other mobile apps that rely on neural networks to classify or otherwise transform images, it should be possible to run neural network inference on a mobile device.

There are also serious questions we must deal with that concern WHEN we would put the app on the app store, whether it would be a free or paid app, whether we would wait until we have gathered pictures labeled by blood tests to deploy, or possibly deploy an earlier version trained on images labeled by patient self-reporting. Another consideration would be how our agreements with Karate Health and other possible venture funding groups would have evolved by that point. If our app is able to classify RA activity with greater accuracy than patient self-diagnosis, it would have considerable commercial potential.

However, the decision about how to commercialize the app may not be entirely up to us at that point. Collecting images labeled with blood tests will likely cost an extraordinary amount of money. If our funders were to pay the whole cost of the blood test, it would cost a minimum of $1.2 million for data collection alone.

Another potential issue might arise if Crescendo Bioscience found out that we were using their blood tests to create an app that would effectively eliminate the market for their product. Would they refuse to sell blood tests to people participating in our data collection project? Or, if we tried to leverage individuals who were already taking the blood test as part of their normal treatment process, would they add some condition to their terms of service preventing patients from releasing the results of their blood test to a third party (us in that case)?

Given these cost barriers and potential thorny legal issues that might arise from training our app with the results provided by a competitor's product, it seems likely to me that we will need another means of labeling data to move beyond the accuracy of patient self-diagnosis. The question is what can provide this improvement? An avenue we might pursue would be to find out how VectraDA validated the accuracy of their blood tests. There must be some "gold standard" against which they measured their blood test to confirm that it correlated with patient outcomes. Perhaps we could apply that method to our case.

Or, if the "gold standard" against which they validated their blood test was an extremely expensive involving test like the DAS28, then perhaps we could find an alternative cheaper measure that was highly correlated with the gold standard measurement.

III. Performance Metrics

A. Application

The goal of the application is to be capable of taking user data related to RA. It should be able to access the camera of the device it's on and take a photo. It should prompt the user to give a self diagnosis of their hand. The app then should send this information to a server to be stored. Last the user should view all past submissions to the server by pulling the server for past submissions.

B. Server Metrics

Be able to service all requests made by the application and interact and store information into a database. This will include but is not limited to POST requests to store user RA data as well as a GET request to send users past submissions.

C. Data cleaning

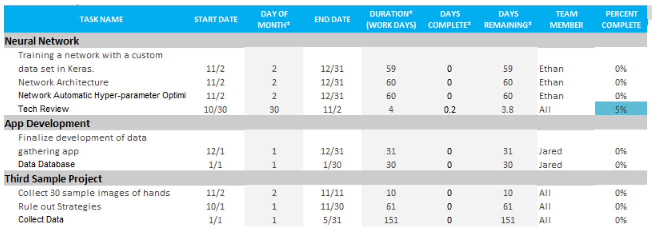
The goal of data cleaning is to reduce the amount of noise present in each image of hands. Lower level of noise will decrease the total number of labelled examples we will need. Success in data cleaning means being able to feed a picture of a hand to the the segmentation network and seeing an image being output with the background behind the hand cropped out and none of the hand removed.

D. Collection of Labelled Data

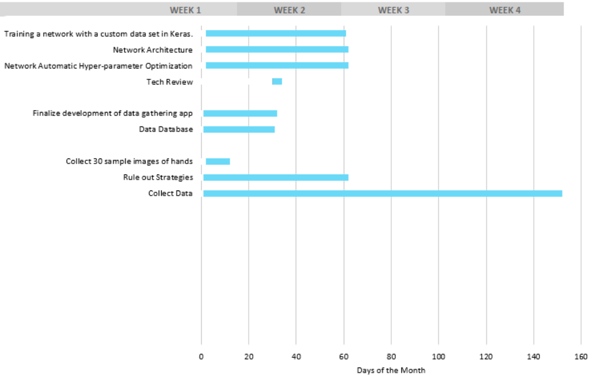
Success in the collection of labelled data would mean publishing a custom data labelling tool on Amazon Mechanical Turk and receiving the results. Success would mean collecting at least at least 500 labelled images.

E. Joint detector

Creation of the joint detector is a stretch goal. But should we get to it, our goal would be to be able to pass an image of a hand to the detector and have it return a labelled image of the hand with bounding boxes drawn around the joints. Accurate drawing of these boxes is the main criteria for success.



TASK NAME	START DATE	DAY OF MONTH*	END DATE	DURATION* (WORK DAYS)	DAYS COMPLETE*	DAYS REMAINING*	TEAM MEMBER	PERCENT COMPLETE
Neural Network								
Training a network with a custom data set in Keras.	11/2	2	12/31	59	0	59	Ethan	0%
Network Architecture	11/2	2	12/31	60	0	60	Ethan	0%
Network Automatic Hyper-parameter Optimi	11/2	2	12/31	60	0	60	Ethan	0%
Tech Review	10/30	30	11/2	4	0.2	3.8	All	5%
App Development								
Finalize development of data gathering app	12/1	1	12/31	31	0	31	Jared	0%
Data Database	1/1	1	1/30	30	0	30	Jared	0%
Third Sample Project								
Collect 30 sample images of hands	11/2	2	11/11	10	0	10	All	0%
Rule out Strategies	10/1	1	11/30	61	0	61	All	0%
Collect Data	1/1	1	5/31	151	0	151	All	0%


WEEK 1
WEEK 2
WEEK 3
WEEK 4
Training a network with a custom data set in Keras.
Network Architecture
Network Automatic Hyper-parameter Optimization
Tech Review
Finalize development of data gathering app
Data Database
Collect 30 sample images of hands
Rule out Strategies
Collect Data
Days of the Month