Mobile assistant app for visually impaired people, with face detection, gender classification and sound representation of image.

Xiaofei Fu, stevenfu@stanford.edu, Electrical Engineering department, Stanford University

***Abstract*—In this project we built a mobile assistant app for visually impaired people. The app runs completely offline on the mobile device, it includes a deep convolutional neuron network for gender detection, and have a way to represent picture content with sound.**

***Keywords*—Mobile, Face detection, gender classification, deep learning, convolutional neuron network.**

I. Introduction

WITH recent development of mobile chips, we now see more and more powerful smart phone in people's hand, processors inside these phones has more computing power than normal desktop PC in 2000. This means a lot can be done on mobile devices that could make people's life easier. Especially, nowadays mainstream mobile operating systems including iOS and Android had carefully thought after accessibility functions build in. With these functions, visually impaired people can use a smart phone almost as conveniently as fair sighted people. However, there has yet to be a good app to help them "see" the world, so the main motivation behind this project is to help visually impaired people to "see" through the lens of their smart phone.

The first aspect of the project is to have a mechanism that run on the phone and detecting faces continuously, to make the most use of it in any environment, ideally this function should run completely offline and do not require any cloud service. In that way, the user can use it when there is no network connection, and it also helps with minimising delays. In the short period of this project we wanted to implement face detecting and gender classification, and build that with future functions in mind, so that the app can be relatively easily expended to be able to recognise other things in the future, for example identify people.

Instead of using the traditional method of Egen faces and Fisher faces, we want to use some new method that can work when the people are not perfectly facing the camera, because that is the situation we have in most of the use cases. We tried to explore the possibility of using a deep learning method called Convolutional Neuron Network(CNN)[7] for our purpose. Further more, we tried not to use a CNN that is specifically trained for our task, instead, we want to use a CNN trained for general purpose and extract features from it's result to achieve our goal. With this approach, the end product will be very flexible to be adapted to perform other tasks instead of only doing gender classification. With that in mind, we chose to use the "ImageNet 2012 mobile" model trained by liu liu of libccv.org[5]. Another reason that we chose this specific model is a traditional Imagnet CNN[7] would have to big a memory footprint(several hundred MegaByte) to be able to run on a mobile device, the model we chose had gone through an extensive size reduction process to reduce the memory foot print to about 100 MegaByte[5]. Also with this model that can identify 1000 object categories, we have free bonus functionality of detecting objects, for example 200 breed of dogs, and lots of everyday objects.

II. Implementation

Overview of the face detection and gender classification process:

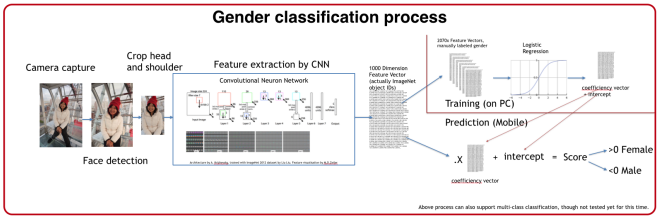


Figure 1. Workflow overview.

As illustrated in about figure, an image will go through these steps in order for the gender of person to be classified:
1. Camera will run in a loop and taking picture at rate of about 2Hz.

2. Picture will go through face detection, and any face bound will be drawn on screen as a preview.
3. User then decide whether they want to only detect the biggest face in the frame, or, the whole picture will be send to CNN.
4. If user chose to only detect biggest face, then the head and shoulder area of the biggest face bound will be cropped out, and the new picture will be send to CNN for classification.
5.CCN will output a 1000 dimension vector of confidences of detected features.
6.Feature vector will be feed into the logistic regression prediction function, then give probability/score for each of the classes.

A. Face detection

There are various techniques that can be used for face detection, for example SVM[13], template matching, etc. Among these algorithms, Viola Johns[1] algorithm was the most popular algorithm that is widely used in industry, because of it's good balance between performance and computational complexity(or simplicity). The Viola Johns algorithm in CIImage class that's provided by iOS SDK can find the face bound, eye/mouth location and face angle. It's working reasonably well for our purpose, so we decided to use it as the face detection mechanism in this project.

B. Gender classification

Facial gender detection was an very active research area, some new approach of image processing related to human face will target this task as a first step, for example we have Egen/Fisher faces approaches that were taught in the course, also those using PCA[12], Logistic Regression[14], etc. But most of these technics works well as long as face is detected, and not facing too sideways to the camera. What we wanted to do in this project, is to explore a possible method that can work relatively well no matter which angle the target is facing.(we can even have 70% without seeing the face at all) As stated above, we chose to use a CNN as the feature extraction part of the classification algorithm. The CNN was trained to detect objects and high level semantics instead of detecting gender directly, the idea is if we can identify people's high level appearance, for example wearing a suit or a dress, hair style etc, we can infer people's gender appearance from those features. By using a CNN that is trained to detect general objects, we can later extend the framework to do other classifications that can be done with high level semantic understanding. We then used Logistic Regression as the classification method on feature vectors from the CNN. As same any other machine learning method, the Logistic Regression need to be trained to give a meaningful result.
Main work of classification part can be divided into 3 parts: Training data acquisition, training and testing.

1) Training data: To train a working logistic regression model, we need a reasonable amount a training data.
We found one good source of labeled training image is ImageNet, but after downloaded all images labeled "man" and "woman" from ImageNet, we found that the diversity of those images are not very good, for example almost 100 of pictures labeled "men" seems to come from the same seminar-like event, and featured the same group of people. To curate the data, we then go ahead deleted some of those pictures and also corrected about 5% of the pictures that was labeled incorrectly. To make our training data set more diverse, we also downloaded several hundreds of images from a online stock photo website, and manually corrected some of the labelling errors in there too.
Considering the CNN model we chose to use do not have localisation build in. To get better accuracy, we chose to train the function with only with portion of the picture that contains most of the features that is related to people. We run face detection on each image, found face bounds and face angle for each faces, then rotate the picture so that the face is in up-right angle, then cropped the area of head and shoulder as our training data.

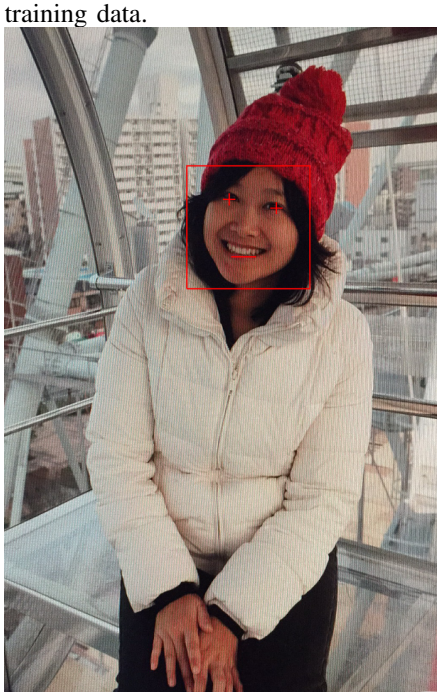


Figure 2. Face detection.

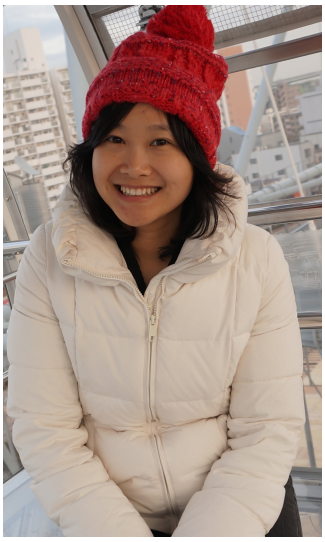


Figure 3. rotated and cropped.

Then we manually checked these images, and used these images as the final training image set, we have in total 2070 pictures (1111 male, 959 female):

Figure 4. Original picture samples(male).

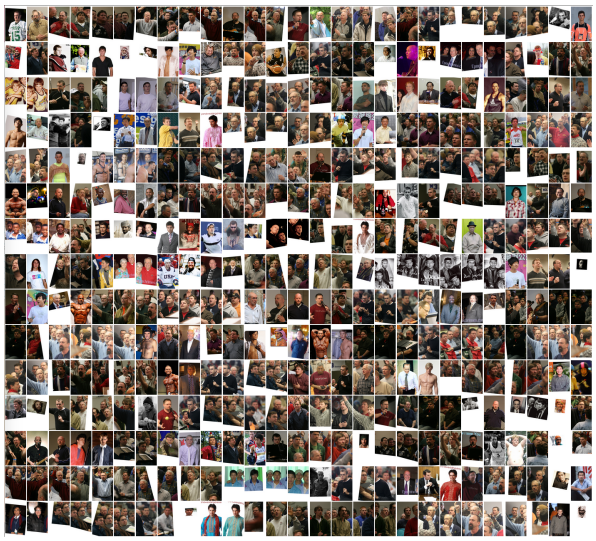


Figure 5. Croped training data set samples(male).

III. Training

To train the logistic regression, we need to get the feature vector from CNN, and since the version of Model we want to use is specifically tailored to run on mobile, we don't want to invest time to make it run on PC and verify it will gave the same results as it's running on mobile. Instead, we chose to feed all the training data on mobile and extract vectors from there. What we got from CNN for each of the training image is a 1000 dimensional vector. Noting the confidence value of each of the detected high level features. Following is a sample of the vectors that we got from training images.

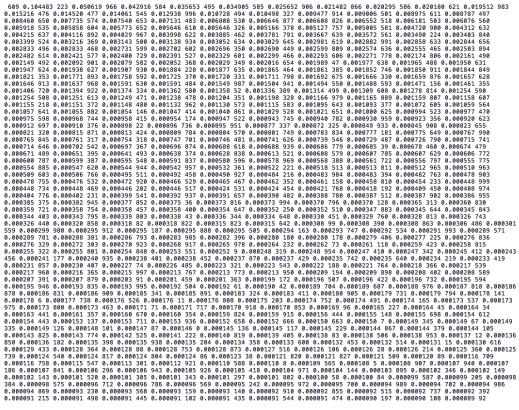


Figure 6. Sample training vector.

We studied these training vectors and found the weight of vectors are very long tail, so we decided to preserve all of them instead of trimming them with top weights and store them as sparse matrix.

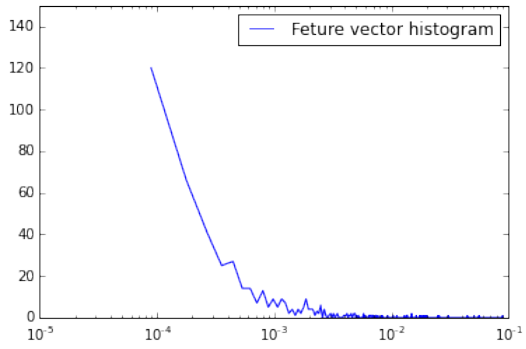


Figure 7. Histogram of training vector weights.

Then we divided all training set into 80% training and 20% validation set. We used the 80% training set to train the Logistic Regression model on a PC using science kit machine learning libraries[3], validation with the 20% set give us 80% of prediction accuracy.(78 errors out of 390 validation images.)
We then extracted the coeficiency vector and "intercept" scalar value from the trained model to use as the forward prediction path we will later use on mobile.
After that the prediction process is a dot product of image feature vector and coeficiency vector then adding "intercept".

$$score = (V_{feat}).*(V_{ceof}) + Intercept$$

If we want the probability of each class, we need to use logistic function on the $score$ to get it, but since our case here only have only 2 classes, we can simply judge which class it belongs by looking at the sign of the $score$.
We tested our forward prediction algorithm with the whole training set to validate the equation, and we got 4.27% and 5.15% error rate for male and female classes, which means it's working reasonability well.

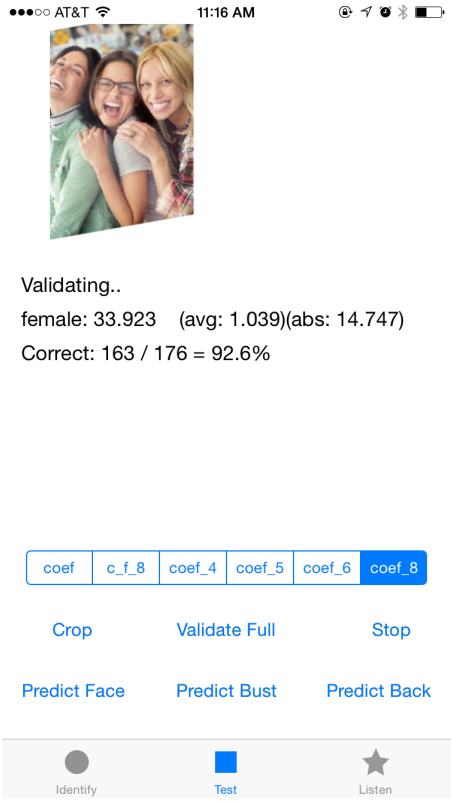
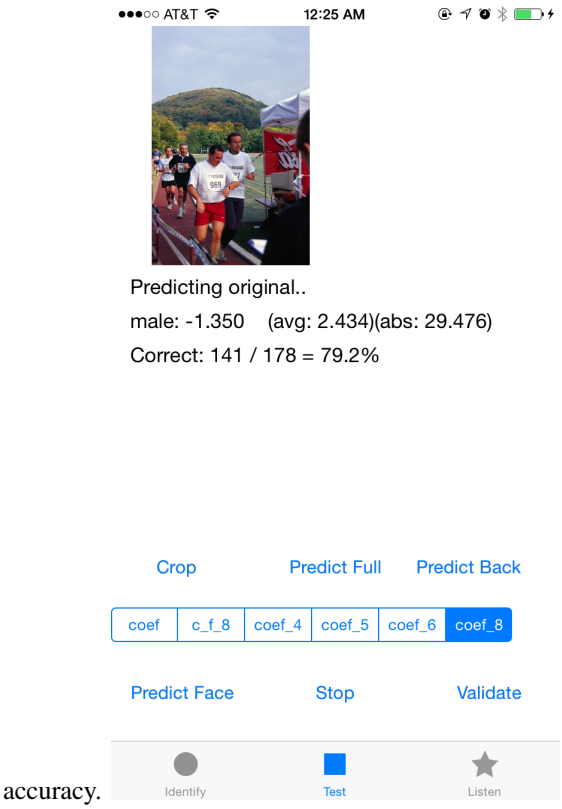


Figure 8. Validating with training data set.

IV. Testing

We then implemented the logistic regression prediction function on mobile iOS using Objective-C, then validated the whole pipeline again with training image set. We got 5-8% error rate, which means the whole prediction pipeline is working as expected:

To test the result of the prediction, we get a totally new dataset from a online search engine, then manually labeled them and feed them into the testing function we build in the app. Result shows average of above 70%



accuracy.

Figure 9. Test result with new test image set, run 1.

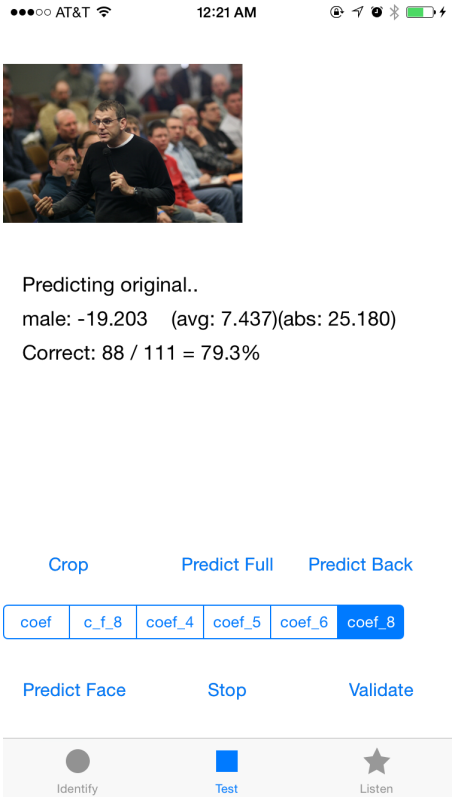


Figure 10. Test result with new test image set, run 2.

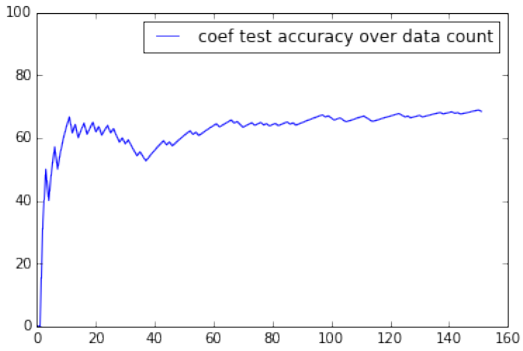


Figure 11. Test result with new test image set, run 3.

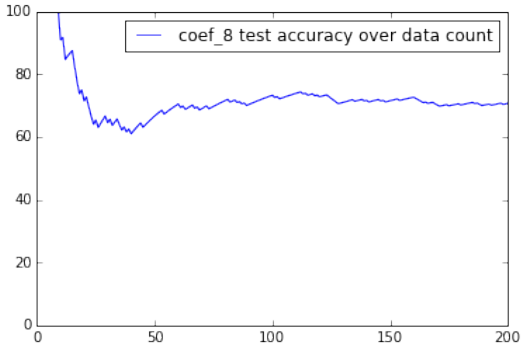


Figure 12. Test result with new test image set, run 4.

V. Image Representation with Sound

Another component of the project is to figure out a way to represent a image with sound in conjunction of detection function. Here is a overview of how it works:

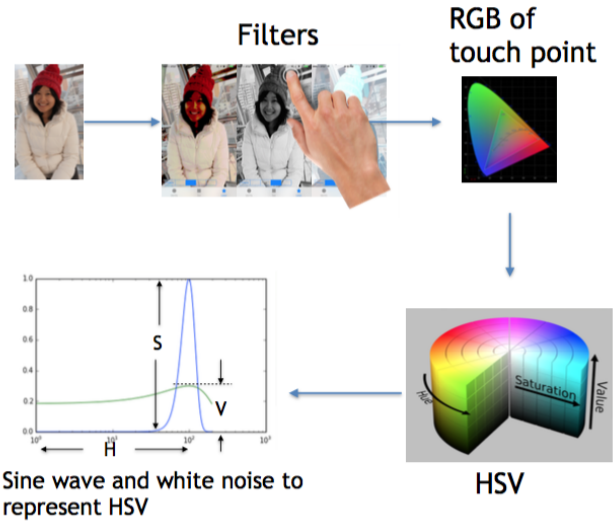


Figure 13. Sound representation work flow overview.

The sound representation works as in above flow:

1.When the view get a image from the result of the

CNN, it will first run several image filtering on it, then present the filtered image in the present view.
2.When user touch the screen, the app will sample the touch point and get the RGB value of the pixel at that location.
3. RGB value of the touch point were then converted to HSV color space.
4.Then we use a combination of a sin wave and white noise to represent the HSV values. The frequency of sin wave will represent the Hue, relative volume of the sin to the white noise will be the saturation, and finally overall volume level will represent value.

Result of this algorithm is, when user touch a black patch, there will be no sound, a white patch will induce full volume white noise. If there is any color, there will be sin component and it's frequency represent color ranging from 200Hz for red, to 6200Hz for blue. The volume of the sin wave representing the color saturation. By scrubbing finger across screen back and forth, user can get a rough idea of the shape of the object in frame.

To improve the user experience, some filters was subjectively chosen to enhance the image before presented to user.
"Posterize" will saturate color and erode color blocks, which makes the sound representation contains mostly only the sin wave representing the color, which makes it much easier to grasp the color of the scene, use case of this filter could be try judge whether a pair of socks are the same color.
Gray scale filter will make the picture lose color information but with subtle variations between volume of the white noise it better suited to identify shape of an object.

VI. Conclusion

Here are some screenshot of the outcome of this project we got running on the iPhone, we did not put too much work into UI design yet since it's still a technical Proof of Concept at this stage, but we are planning to move forward and build a actual product based on this study and release it into App Store.

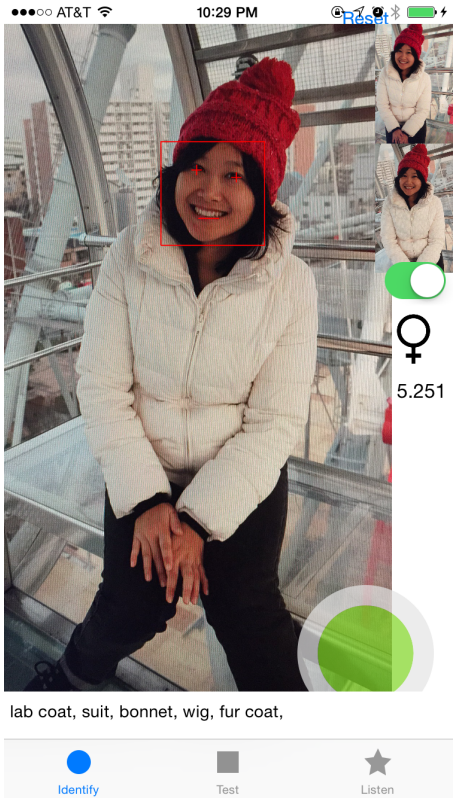


Figure 14. Screenshot of main tab.
Main tab includes the main functionality of this app, the small window on top right corner is the real time preview of camera feed, it's taking pictures in a loop, then we run face detection on the captured image and display it in the big view in the middle. At the same time based on the use chose of the green switch button, either the whole image or the detected head and shoulder area will be feed into CNN for classification. The classification will run 1-2 seconds on a iPhone 6, when it gets a result, the app will display the actual image that had just went though on the smaller view that's below the live camera feed view. It will also display the gender and score with the icon and score label on the right. The text view on the bottom will display object labels that come out of the CNN object detection. Will also be helpful when users are interested in detecting object instead of people. Testing shows the app has above 70% percent of accuracy, considering the CCN underneath has top-1 missing rate at 36.83% and top-5 missing rate at 18.20% after quantisation, by consolidating high dimensional knowledge, our gender classification actually managed to beat the CNN's top-1 accuracy and only slightly behind it's top-5 accuracy, which is a good result for this experimental stage.

Figure 15. Screenshot of "Listen" tab.
Listen tab will apply filters to the latest image that when through CNN, and be able to synthesis sounds to represent the RGB value of touch point.

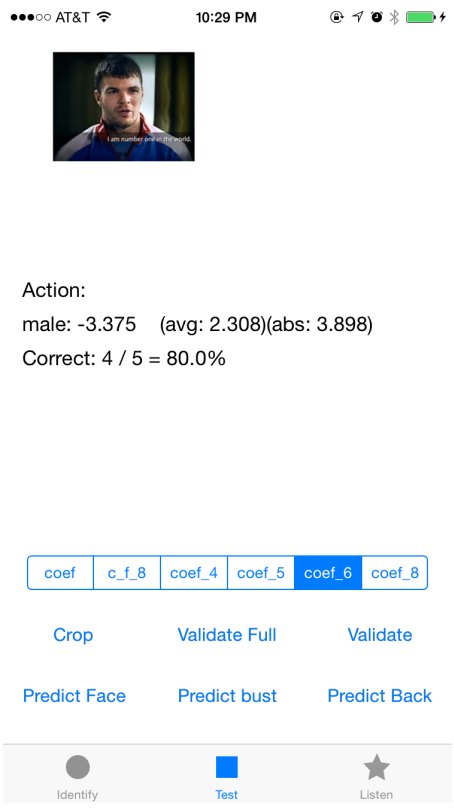


Figure 16. Screenshot of testing tab.
This tab is build only to test the classification function, it has various testing and validating functions to read and test images from pre-defined directories then log and present test results.
We used these function extensively during the experimenting of training phase, testing different coeficiency vectors that we got from different training settings.

VII. Future work

When we compare the testing result with the validation performance, we can see a clear sign of over fitting the model with the training dataset. Also, by examining the training dataset, we see there seems to be a lack of diversity of scenes, locations, population, etc. in the ImageNet dataset, which was majority of the training data. In the future we could add some more image source to the training set for example adding images from social media sites like facebook or tumblr to enrich the training data set. Also training data amplification technics like blur, scale and crop had been mentioned in various literatures[2][6] to be able to help lower chance of over fitting, and worth a try in this project too.
Also, when face can be detected with good orientation, we could implement a traditional face processing algorithm to work together with the method in this project, so the combined effort would have much better overall performance.

APPENDIX A
CHALLENGES, WORK AND SOURCE CODE
INTRODUCTION

In this project, we build a gender classifier using a CNN that work on mobile phone, making the CNN running on mobile and extract feature vectors from it took a lot of work, considering traditional ImageNet CNN will have near Gigabyte of memory footprint and impossible to run locally on mobile devices, it was only made possible on late 2014 by the open source project[9] that used.

Due to the computational intensive nature of CNN forward path, it was very tricky to make it run well together with the camera capturing loop, without locking up UI occasionally. We got it CNN running very well in separate thread in the testing suit without locking up UI at any moment, since the tests will run for hours sometime, and without getting that right, we would not get any meaningful test result.

Another challenge is to gather training image, processing them with face detection and crop out areas that we need to train the model. As with any machine learning project, large amount of high quality training data is always the key of high quality results. We wrote a download script to download images from imageNet link file, also a scrubber to download images from a stock photo website. Then we wrote a cropping function on mobile to detect face and rotate and crop "head and shoulder" area that we needed. We also wrote the function to crop "face only" and gathered a huge amount of training data for face only images, and trained a model to work with face only, it has ¿95% accuracy with training data, but lack of meaningful performance on test data(a sign of over fitting model and shortage of training data), so I decided to leave that outside of the end report.

A big part of the work goes to build the testing framework on mobile, as mentioned in conclusion part, we build the testing module to randomly chose testing image from pre-defined directory in app's "documents" folder, then log the testing result, and display testing progress on the fly. This testing frame work is essential for choosing the right training parameters to come up with a useful model.

When building the sound representation function, a lot of work goes into experimenting the best way or representing RGB value to arrive to the final design. Then we spend a lot of time trying out different image filters to enhance the image, before sending them to the touch/scrub interface.

In the uploaded source code package you will find following key components of the project:

ee368_prj_imagenet.py
This is the script that I used to train the logistic regression model that I extracted from the mobile device, there are also many data analysing functions inside of this file.

data_presenter.py
This is the collection of functions that I used to analysis various data during the process, also to generate some of the illustrative graphs.

imagenet_downloader.py
This is a multi-thread downloader to download ImageNet images from the individual links in a text file.

testLibccv.zip
This is the XCode project for the mobile client, inside of it you can find most of the work represented in these files:

"FirstViewController.m"
This is the main view of the app, includes camera capture loop, face detection and image rotation/crop, drawing face bound and eye location on preview window, also the function calls to CNN and gender classifier.

"SecondViewController.m"
This is the testing framework. It includes the cropping function that I used, image resizing, rotation. Also a automated testing framework for classifier, which includes live result update, and logging functionality.

"ThirdViewController.m"
This is the view of "sound representation", it will take the output from main view(when there is a picture went through classifier), and represent it will sound representing RGB values of touch point.

"KLHierarchicalClassifier.m and KLClassificationHierarchy.m"
These originated from the libccv project[9], I have modified them intensively to extract 1000 feature vectors from CNN, load gender detection coeficiency vector from file, and perform logistic regression prediction then give a classification score.

"AudioController.m and BleepMachine.m"
These are the code that converts RGB value into HSV value then generate sound to represent it. Some of the sin wave and white noise generating functions were adapted from various open source code base that I lost track of the origin.

Note:
The project will only build to run on hardware devices with valid provisioning profile, it will not run in XCode iPhone simulators due to it's dependancy on NEON vector acceleration.

ACKNOWLEDGMENT

The authors would like to thank Professor Girod and Wetzstein for a wonderful course. Also thank TA Chen and Boin for their helps throughout the quarter.

APPENDIX B
REFERENCES

1 Robust real-time object detection P Viola, M Jones - International Journal of Computer Vision, 2001.

2 Imagenet classification with deep convolutional neural networks A Krizhevsky, I Sutskever, GE Hinton - Advances in neural, 2012 - papers.nips.cc

3 http://scikit-learn.org/stable/

5 Yunchao Gong, Liu Liu, Ming Yang, Lubomir Bourdev. Compressiong Deep Convolutional Networks Using Vector Quantization.

6 Zeiler, Matthew D and Fergus, Rob. Visualizing and understanding convolutional neural networks. arXiv preprint arXiv:1311.2901, 2013.

7 Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems, 2012

8 LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., and Jackel, L.D. Handwritten digit recognition with a back-propagation network. NIPS, 1990.

9 libccv.org

10 developer.apple.com/

11 https://github.com/scikit-learn/scikit-learn

12 Face detection and gender detection using principal component analysis (PCA) Suri, P.K.; Walia, E.; Verma, E.A. Communication Software and Networks (ICCSN), 2011 IEEE 3rd International Conference on Year: 2011 Pages: 679 - 684, DOI: 10.1109/ICCSN.2011.6014983

13 Training support vector machines: an application to face detection Osuna, E.; Freund, R.; Girosi, F. Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on Year: 1997 Pages: 130 - 136, DOI: 10.1109/CVPR.1997.609310

14 An automatic face detection and gender identification from color images using logistic regression Rahman, M.H.; Bashar, M.A.; Rafi, F.H.M.; Rahman, T.; Mitul, A.F. Informatics, Electronics & Vision (ICIEV), 2013 International Conference on Year: 2013 Pages: 1 - 6, DOI: 10.1109/ICIEV.2013.6572576