CV Report

1 Introduction

Motivation behind this project is for me to explore recognition, trying out multiclass classification, and lastly to have fun. I'm curious on how well systems perform person classification with limited data, compared to humans. This project helps me experiment with different data collection methods.

In this report, I will be describing data preparation and techniques that I used to analyze the video. In each section, I will briefly describe the challenges along with examples that I faced. Most importantly I will be showing and discuss the results of each tasks. Lastly, I will discuss things to consider for future projects.

I will be analyzing this video (click Genie) broken down into manageable scenes. This video is property of Music Core Korea, Genie and Girl's Generation is owned by SM Entertainment.

2 Data and Video Description

In this section I will be describing the training data and the video that I'm working with. The training data for the binary classifier are simply just pictures of the members singing and the members not singing. The pictures of girls not singing varies (standing up, sitting down, photoshoot, candid, selfies/self captures), while the singing pictures are mostly from their live shows, holding a microphone, and with their mouth in singing position (open). The pictures are also taken from multiple viewpoints, lighting conditions.

The training data for the multiclass classifier are random shoots of the four members. Because the video was taken back in 2009, I tried to stay relevant and gathered pictures of the members back from six years ago. I'm curious to find out on what kind of features are distinct from each member to a classifier. Most of the pictures are from their live performance on stage. **All of the training data are not face aligned** as shown below. Here is a visualization of the CNN features from the second convolution layer.

The images for the multiclass classification are collected from various websites, manually annotated for faces, and automatically cropped down to a 32×32 pixels images. I realized a little too late that I can automatically annotate the faces using the face detector that are provided to us. The binary classification set are almost fully automated, I had to go through the set quickly to get rid of very negative examples. In the end, I collected roughly 20 pictures per member for multiclass classification, and 55 pictures of singing and non-singing classification.

The video that I will be analyzing is a video of the group dancing on a clear sunny day. I picked this video because there are not a lot of complex shot boundaries, no complex scene transitions, no zoom ins, and no different background or crazy lighting effects. Lastly, it would be easier for me to work with as my first video analyzing task. The video however, does contain a lot of complex angles and are focused in the center; edges of the video are blurred by the music company.

Figure 1: **Multiclass: Taeyeon - 0, Jessica - 1, Tiffany - 2, Yoona - 3**

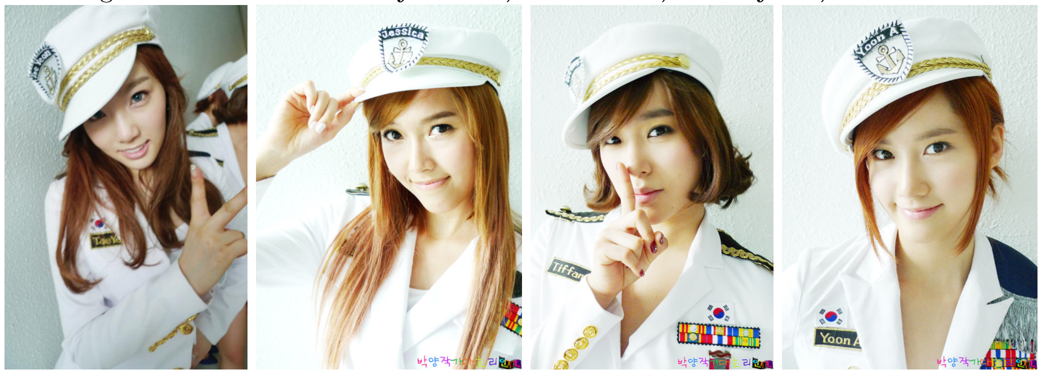
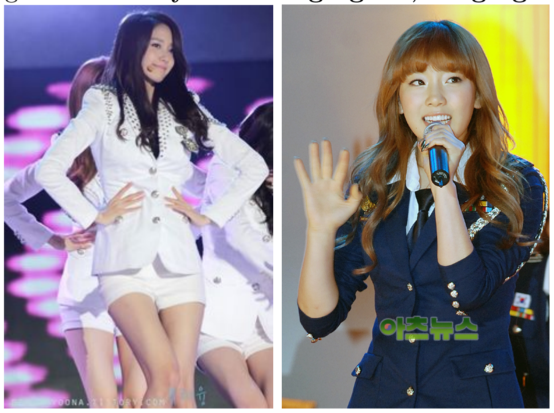
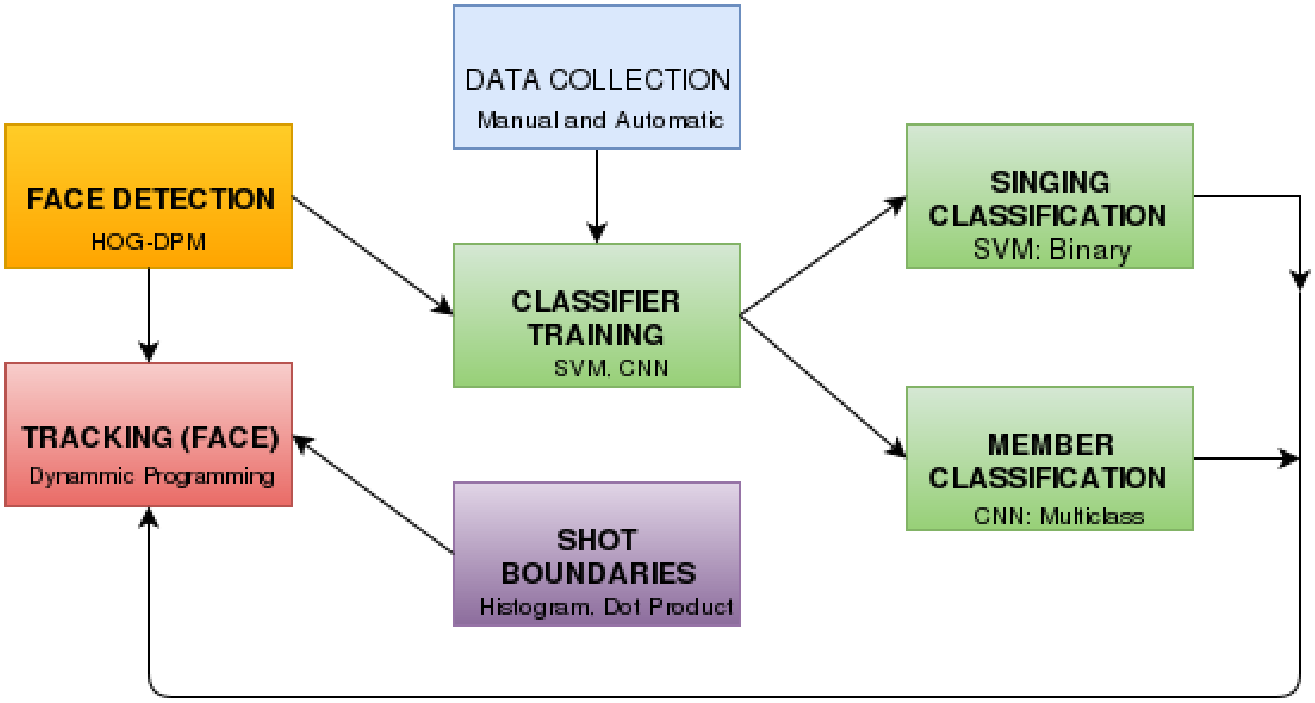


Figure 2: **Binary: Not singing - 0, Singing - 1**



3 Tasks



Here is an overview of what I did (or attempted with some success) for this project. Code pipeline are as follows (pipeable code):

3.1 Preparation*

I began the whole project by collecting data for the members. Because there are nine members, I was advised to collect images for four of them instead. Furthermore, there are no known, or at least to my knowledge, dataset of Girl's Generation faces. Check out the link at the bottom for my repository.

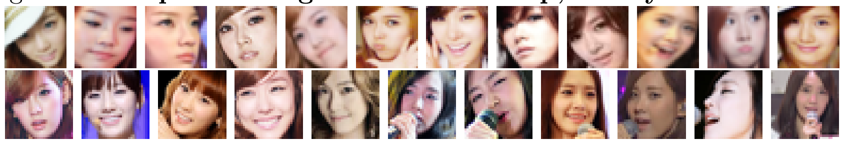
The multiclass training set are gathered semi-manually. I collected the image links, annotated the faces, and labeled the faces manually. As for the binary classifier, everything was done almost purely automated. The dataset and extraction code are shown at the end of this report. The manual data gathering was a slow agonizing processes over the span of two weeks, while the automated was done in less than half an hour! Aside from data gathering, I cut the video into four scenes. The scenes extracted shows a wide variety of scene changes, poses, and I believe it summarizes the video well.

Challenges: The challenges that I faced was gathering the data in general. As mentioned above, this video dated back to 2009 and thus, there are very limited number of clear images available online. Manual face annotation are proven to be a really hard and mind-numbing process to do. I tried to crowdsource manual face annotation but it did not work at all.

I also notice that running the frames on full size (720 $\times$ 1136) are very slow and memory inefficient on CDF. I had to downsample all my images down by half.

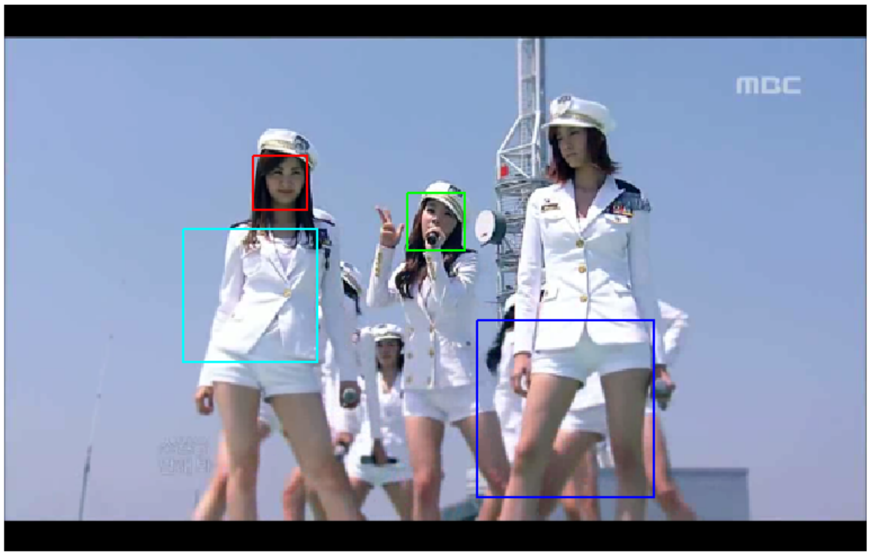
Future Work: Definitely will be using face detection and auto cropping to gather data. Perhaps to further automate I can also extract links from Google automatically. As for weeding out really negative examples, I could use PCA to determine if a given cropped patch is a face or not. This is done by reconstructing the patch with PCA and see if its close or not to its original patch. "The reconstruction of x is similar to x if x lies in the face subspace" [CSC320 Slides PCA].

Figure 3: **Sample training for multiclass top, binary at the bottom**



3.2 Object Detection

The second major task was to detect faces. Fortunately the code given for project one for face detection was working really well for me when I ran it on my clips. The detection (bounding boxes) are very essential to do the next few tasks. Simple tracking relies on detections as we have seen in assignment 5. I need the bounding boxes to crop the faces out from the frames and perform classification. **I performed all face detection on the full sized image** because when I perform it on the half sized image, a lot of the scores are very negative and unreliable. The code used here are the deformable-part-model object detection made by the folks at [Ref: 1]. Here is a sample result:



Challenges: There are a couple informal tweaks that I did to get the project to work smoothly. I noticed that the model threshold are very high (approximately 0.7) and was not picking up any detection due to the video being on the lower end of quality. In the end I settled for a lower threshold with a lot of false detections. I ignore possible false detections by checking their log score values for really negative numbers and imaginary results.

Getting rid of doppelganger detections was fairly easy; I lowered the NMS threshold.

3.3 Classification

There are two classification problems that I need to tackle in this project. Initially, I wanted to implement the singing detection as an object detection problem. But classifying if a person is singing or not is inheritly a binary classification problem; 1 for singing, 0 for not singing. In addition, classifying the members are done as a multiclass classification problem; Taeyeon - 0, Jessica - 1, Tiffany - 2, Yoona - 3.

Multiclass classification was done on caffe, CNN using the exact LeNet model. Its a standard model that utilizes two convolution steps and two inner products. I was initially using a bare caffe provided by CDF, but I found out about using a webapp GUI made by the folks at NVIDIA called DIGITS [Ref: 2]. Using the images that I have collected as my training images, and face cropped by the detection as my test set. I did not do any validation because of the limited amount of data that I am working with. Hyperparameters such as learning rate, momentum, and number of epochs are tweaked a couple times to yield what seems to be a better result.

Binary classification was done only with SVM through the python library scikit-learn. I constructed a linear SVM using hyperparameters as per suggested in this paper [gamma degree polynomial degree 1].

For all of classification, I decided to not operate in grayscale because hair color is a good indicator of which member the face belongs to (good feature).

Results: The results that I am getting was not too satisfying for multiclass classification, though extremely interesting. For the multiclass classification, I notice that most often the girls are classified as Yoona. For example below.

Figure 4: **Taeyeon classification example**

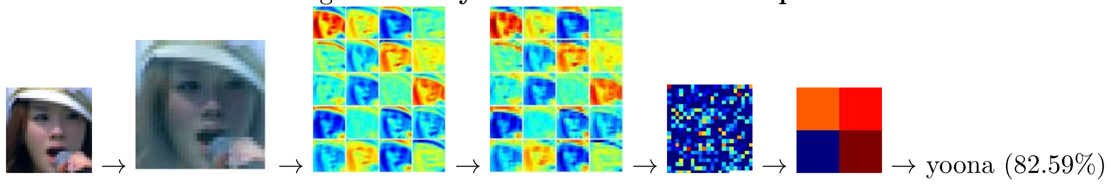
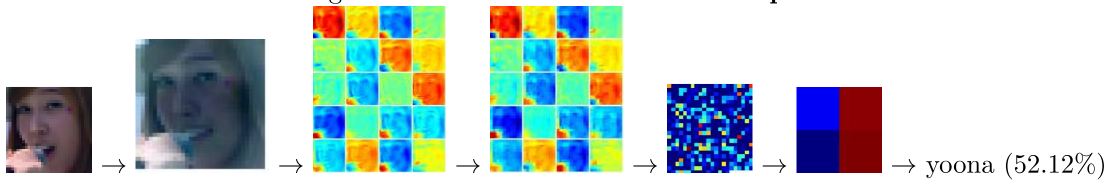
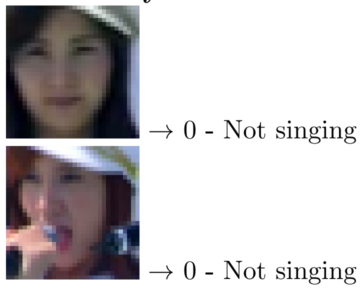


Figure 5: **Jessica classification example**



Binary classifier are pretty decent with SVM and small data. Here are a couple sample results: The reason the

Figure 6: **Binary Classifier example**



detections are still a flip of coin sometimes is because the small training data that I gathered.

Future Works: For the future, I might try a one-vs-rest classifier, i.e: is this Jessica or not as opposed to which member is this. Also, I wanted to do a backup linear SVM for classification to ”improve reliability”. Since I do not have pictures for all nine members, I cross-checked the result from SVM and CNN to see if they match up. If they do then output classification, otherwise do not. I also need to gather a lot more training data next time. I'm also interested in trying to image process the data before feeding into my classifier, for training or testing.

3.4 Shot boundaries

Shot boundaries are different from all of the other tasks that I have done. My method is similar to the one given in this paper [2], it compares histogram of the three color channels of one frame to the next one and compute scores. Histogram of color gives me back a vector, so scoring in this case is merely just the cosine of the angle between the current vector and the next. If the score is close to one, chances are the two frames belong in the same shot. By trial and error, I found 0.7 to be a good threshold. So everything below 0.7 would be a shot boundary.

$$\frac{\mathrm{H}_i.\mathrm{H}_{i+1}}{|\mathrm{H}_i||\mathrm{H}_{i+1}|} > 0.7 \Rightarrow \text{Boundary detected at } i - i + 1$$

To compute performance, I use the precision recall method. I first manually annotate the correct shot changes and compare that to the amount of correct shots I detected. For the scenes I have picked, my performance is a surprising 100%, perhaps overfitted when picking my threshold. Shot boundaries are important in tracking as it determines up to which frames are the detections relevant.

Future Work: My video contains no complex scene transition as I mentioned above, and it is why this simple method works very well. I assume that this method would work quite well for most of Girl's Generation videos because their shoots are very steady, transition are clear. The only problem might arise if the transitions are very similar from one scene to the next one. Then methods with optical flow might have to be employed.

```
Shot boundaries detected score: 4.110832e-01 scene: 1 frames: 69-70
Shot boundaries detected score: 5.138757e-01 scene: 1 frames: 117-118
Shot boundaries detected score: 5.227193e-01 scene: 1 frames: 147-148
Shot boundaries detected score: 5.524651e-01 scene: 1 frames: 203-204
Shot boundaries detected score: 6.623986e-01 scene: 2 frames: 96-97
Shot boundaries detected score: 6.842973e-01 scene: 2 frames: 145-146
Shot boundaries detected score: 3.644131e-01 scene: 2 frames: 178-179
Shot boundaries detected score: 3.041484e-01 scene: 2 frames: 207-208
Shot boundaries detected score: 5.884573e-01 scene: 3 frames: 29-30
Shot boundaries detected score: 4.490244e-01 scene: 3 frames: 81-82
Shot boundaries detected score: 5.684107e-01 scene: 4 frames: 49-50
```

3.5 Tracking

Using the information retrieved from the shot boundaries, we can now perform face tracking. Tracking is done similar to assignment five using dynammic programming. With the intention to be able to get all of the detections with more than half the length of the shots.

Challenges: Upon generalization, I ran into a major technical difficulties that was impossible to do with the time constraint that I had to deal with the past couple of days, which requires me to improvise a lot of the tracking. Instead of recursively checking, I hardcoded the amount of path to check over for each scenes.

Everyone is tracked as Yoona because of MATLAB difficulties. Explanation in the code section.

Future Work: I'm interested in trying the Lucas-Kanade method, using optical flow for better track matches.

4 Putting it All Together

I have provided an out.mov for the first scene of my projection. Due to catastrophic tracking malfunction, I was unable to provide the rest of the scenes. I was also having major difficulties with reading and working with different filetypes in matlab. I cannot seem to convert type in matlab, also fseek-ing was not working for me when I was reading in my predictions.

I however, attached all the cropped faces and its corresponding predictions for multiclass and singing.

5 Informal Citations

1. * [CSC320 Slides PCA] http://www.cs.toronto.edu/ guerzhoy/320/lec/pca.pdf#27
2. Digits https://github.com/NVIDIA/DIGITS
3. Scikit-learn SVM http://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html
4. Shot boundaries method
   http://info.ee.surrey.ac.uk/Research/VSSP/Publications/papers/Yusoff-ecmast98.pdf
5. Dataset format reference
   http://vintage.winklerbros.net/facescrub.html

6 References

1. P. Felzenszwalb, R. Girshick, D. McAllester, D. Ramanan
   Object Detection with Discriminatively Trained Part Based Models
   IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 32, No. 9, Sep. 2010

7 Code

I will briefly explain what each code is doing. And also the folders that I attached. **extractframe.m**
Extracts frame (later on and resizes) by half from the .avi video.

computehistscore.m
Computes histogram score between frames and returns the boundaries.

bestpath.m
Dynammic programming implementation of the face tracking, works for first and best path, does not work recursively.

graball.m
Preload all necessary informations such as detections, similarities, etc.

SVM.py
Loads the binary images, train and predicts with SVM classifier provided by scikit-learn's SVM library.

showme.m
Attempts to annotated frames with all the metadata such as members and is singing onto each frame. However due to fseek failures, and uint32 and uint8 conversion errors, I cannot fix the problem in time for the presentation and hard deadline.

trackmain.m
Main tracking code. Runs best path on all frames (ideally).

Caffe Models
Contains the CNN model that I use.

Classifier Predictions
Contains my classifier predictions, CNN for the multiclass and SVM for the binary classifications, for all of the faces detected.

Results
Contains a 25 seconds long mov file for my first scene.

Training Data
Contains all of the training data that I gathered and cropped already.

g3nie
Contains my cropping scripts, dataset samples, and fully cropped faces of the members.

Let $f(x) = xe^{x^2}$ and $g = f(f(x))$. What is $g'(1)$?
Solution: Don't sub in anything Lam wants you to do it this way. It gets too complicated when you sub in

$$\begin{aligned}
g' &= f'(f(x))f'(x) \\
f' &= e^{x^2} + 2x^2e^{x^2} && \text{By Chain Rule} \\
f(1) &= e \\
f'(1) &= e^1 + 2(1)e^1 = e + 2e = 3e
\end{aligned}$$

Therefore...

$$\begin{aligned}
g'(1) &= f'(f(1))f'(1) \\
&= f'(e)3e \\
&= (e^{e^2} + 2(e^2)e^{e^2})3e \\
&= e^{e^2}(1 + 2e^2)3e && \text{Factor out} \\
&= 3e^{1+e^2}(1 + 2e^2)
\end{aligned}$$

Figure 7: **Results**