Attribute Classification: A Varied Approach

CS280 Final Project, Spring 2015

Michelle Nguyen, Allen Tang

1 Overview

Human attraction based on physical characteristics translates to a picky phenomenon. Individuals value a wide range of varying attributes, preferring certain facial shapes, hair colors, eye colors, hair lengths, etc. Due to advances in social technology, we are meeting more potential mates more quickly than ever. However, filtering out matches for physical compatibility can be a relatively time-consuming task.

We use classifiers built upon support vector machines, k-nearest neighbors, and convolutional neural networks[1] to classify various attributes that are important factors in human attraction. Some of these traits are commonly explored in prior works, such as race and hair color[2]. However, our method also allows us to consider more novel features, such as the presence of makeup. These classifiers are then run on "real-world" image sets to create a database of photos that are indexed by attributes. Upon this database, we implement a simple search engine that allows users to easily filter for people of their preferred features.

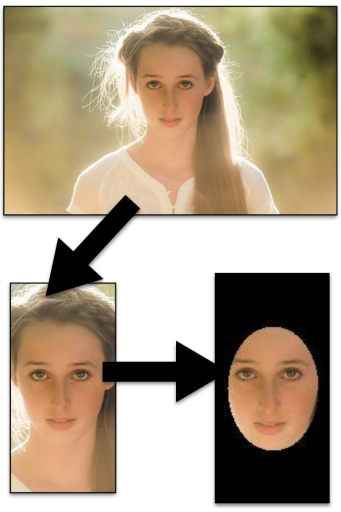

Figure 1: Steps in processing data

2 Tools

Most of our preprocessing work, and some classification, was done primarily in Python. Data was collected using the Python Flickr API kit. Face detection was then performed using OpenCV in Python. K-means and k-NN was ran using the Python package, scikit-learn. For SVM classification, we used LibSVM. Finally, CNNs were implemented in Caffe, running on a GTX 980 GPU.

3 Dataset

We created our own datasets using the Flickr API, collecting images based on a set of image tags, i.e. "asian portrait". By using this versatile method of querying Flickr, we can easily create data for features that have no available dataset online. After pulling these images from Flickr, we had to manually clean our data by removing falsely tagged photos, excessive repeats, grayscale images, bad pose, and more. This constraint required us to generate a reasonably large dataset. This was cumbersome to obtain, but ideally allowed us to generalize our classifiers on a more extensive set of images that capture a wide range of conditions, such as illumination and expression. This ability to generalize is very important given that our ultimate goal is to run our classifier on user-generated images. However, it is important to note that Flickr image data is not completely diverse, as it is heavily biased toward photos of females and certain styles of photos.

After collecting the images, we used the Viola-Jones algorithm in OpenCV to detect faces in the photos. Since one of the attributes we wanted to consider was hair color, we increased the vertical size of the cropped bounding box to include the person's hair as well. Finally, each crop was resized to 100x200.

For our project, we considered three attributes:

race, hair color, and the presence of makeup. Our dataset for race consists of four possible categories. For the races Caucasian, Asian, Hispanic, and African we have 1097, 1627, 1124, and 607 images respectively. We categorize hair color into three categories, blonde, dark (which includes black and brown), and red. For these categories, we have 572, 915, and 415 photos respectively. Finally, the problem of detecting the presence of makeup is a binary classification. We have 709 images of people with no makeup and 946 images of people with makeup. We used 20% of our images as our test set.

4 Feature Classification

We classified three different attributes: race, hair color, and presence of makeup using several classifiers.

4.1 SVM

Our first approach was to classify race using a SVM with a linear kernel. We tried a variety of techniques to standardize the images:

1. No change
2. Normalization to mean 0 and unit variance
3. Normalization by dividing image mean

We denote the above as 'config' in our results section.

We also experimented with using the raw pixel values versus converting the image to grayscale. Then, we applied a Gaussian blur with $\sigma = .7$, and applied a specific mask which gave us the face and hair. We experimented with parameters and found $c = 0.05$ gave us best results.

4.2 k-means/k-NN

For the attribute of hair color, we tried an approach involving both k-means and k-NN. We first applied a mask to each image to extract the hair at the top of the person's head. After applying the mask, we used k-means and vector quantization to find the most dominant color in that image region. The RGB values for that dominant color were used as the feature vector for that image in k-NN. By doing so, we could determine the label for an image by taking the most common label for images with similar colors. From our predictions, we found that using a large number of neighbors kept us from overfitting. We determined that our best value of k was 300.

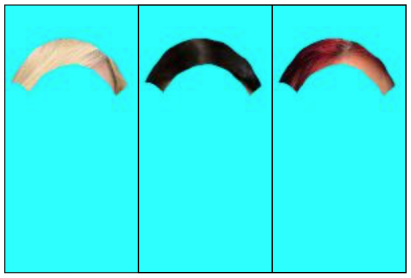

Figure 2: Examples of masked hair. A cyan background was used to distinguish from black hair in the k-means step.

4.3 CNN

CNN was used as a classifier for all three attributes. Before sending the images through our neural network, we first resized our photos to a size of 227 x 227. Then, we attempted a variety of CNN architectures, each trained for 10,000 iterations. Our most successful neural networks were:

1. AlexNet with Softmax
2. Modified AlexNet, with convolutional layers 3, 4 removed with Softmax
3. Modified AlexNet, with convolutional layers 3, 4, as well as fc6, relu6, and drop6 removed with Softmax

We denote the above as 'config' in our results section.

5 Results

5.1 Race

Figure 3: Examples of training data for each race

We found that training a CNN on our race dataset gave slightly better results than those that we got from training a SVM. This was no surprise, and is likely due to all the variance in our dataset, which CNN is more robust against.
Surprisingly, when running a CNN on our dataset without masking for faces, we obtained an accuracy of 74.9%. This is not much worse than training the CNN on the masked version of the data, which gave us an accuracy of 76.2%.
Finally, we found that despite the limited amount of training data we had, the full implementation AlexNet gave us the best results over the simpler networks that we attempted.

Table 1: Linear SVM, config=3, c=0.05, facemask

Race	Precision	Recall	Samples
Caucasian	0.69	0.69	219
Asian	0.62	0.62	325
Hispanic	0.81	0.81	225
African	0.93	0.93	121
	MAP	Accuracy	
Total	0.76	0.73	890

Table 2: CNN, config=1, facemask

Race	Precision	Recall	Samples
Caucasian	0.73	0.73	219
Asian	0.73	0.68	325
Hispanic	0.82	0.83	225
African	0.75	0.89	121
	MAP	Accuracy	
Total	0.76	0.76	890

5.2 Hair Color

For hair color, we tried using k-NN and CNN as our classifiers. For k-NN with a large number of neighbors, we were able to obtain reasonable results which gave us 76.7% accuracy. Training a CNN on images with a mask for hair gave us even better results at 80.9% accuracy. For both classifiers, we see it is easier to distinguish red hair from the other two categories. However, training CNN on images without the mask was not as successful and gave numbers that hovered around a 60% accuracy.
Overall, we found our outcome surprisingly successful despite the high variance in the photos for lighting, pose, and hairstyle.

Table 3: k-means/k-NN[k=400], hairmask

Color	Precision	Recall	Samples
Blonde	0.68	0.74	114
Dark	0.77	0.72	183
Red	0.90	0.92	83
	MAP	Accuracy	
Total	0.78	0.77	380

Table 4: CNN, config=1, hairmask

Color	Precision	Recall	Samples
Blonde	0.74	0.82	114
Dark	0.81	0.76	183
Red	0.90	0.92	83
	MAP	Accuracy	
Total	0.82	0.81	380

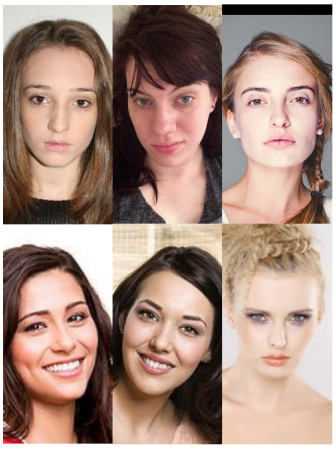

Figure 4: Faces with no makeup (top) versus with makeup (bottom)

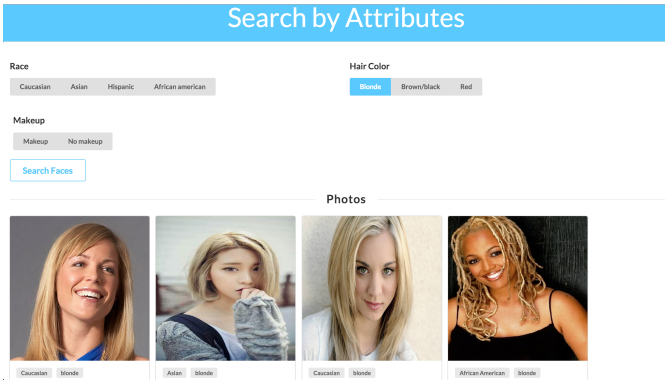

Figure 5: The Attribute Search frontend

5.3 Makeup

Finally, we trained a CNN for our binary classification problem of detecting the presence of makeup on a person's face. This gave us an accuracy of 82.5%. Although makeup can often be very subtle, such as simple eyeliner and mascara, these numbers are promising.

Table 5: CNN, config=1, unmasked

Race	Precision	Recall	Samples
Yes	0.76	0.89	142
No	0.90	0.78	189
	MAP	Accuracy	
Total	0.83	0.83	331

6 Search Engine

http://attributesearch.herokuapp.com

We then collected images by using the Flickr API with search queries that did not correspond to any of our attribute categories. For instance, we pulled images from Flickr corresponding to the generic tag, "portrait". Then, with the best classifiers for each of our attributes, we classified each image and stored them with their corresponding categories in a PostgreSQL database.

Upon the database, we built a simple search engine using Ruby on Rails. This search engine idea is similar to the search engine created by Kumar et al. The search engine allows users to filter for attributes for a set of images that have been categorized by our classifiers. The images are displayed along with tags showing what attributes the photo's subject has.

7 Conclusion

Although our results, given our limited dataset, were not as good as those obtained via the region-specific SVMs by Kumar et. al, we have shown that neural networks can provide a decent framework for general classification problems involving attributes. We were able to get close to 0.75 accuracy or above on a wide range of attribute problems with an almost off-the-shelf implementation of AlexNet. In contrast, our semi off-the-shelf SVM approach gave decent results, but certainly required more tuning.

Obtaining good data from Flickr was very challenging and tedious. Resorting to using Amazon Mechanical-Turk would have certainly helped in this regard. Due to these constraints, we were forced to run our classifier on a smaller dataset (orders of magnitude smaller in size than those used in previous approaches) than we would have liked. Due to our

own human error, these datasets may also be slightly noisy as well. However, even so, we were able to get promising results. With an even larger and cleaner dataset, predictions may be even more accurate.

8 Future Work

There is certainly a possibility that well-tuned neural networks may eventually achieve results comparable to state-of-the-art approaches using region-based SVMs. To do this however, will likely require a very large, accurately-labeled, and diverse dataset. This area would certainly be of interest for future work.

Another topic to consider is the exploration of even more novel attributes. Some traits we had wanted to tackle were eye color, which is feasible with Flickr's high resolution photos, and detecting whether the photo's subject had dimples.

References

[1] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification with deep convolutional neural networks. In NIPS, 2012.

[2] N. Kumar, P. N. Belhumeur, and S. K. Nayar. FaceTracer: A Search Engine for Large Collections of Images with Faces. In European Conference on Computer Vision (ECCV), pp.340-353, Oct, 2008.

[3] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun. Pedestrian detection with unsupervised multi-stage feature learning. In CVPR, 2013.