Is He Chinese, Korean or Japanese? —— East Asian Ethnicity Classification

*All the images in this poster are from our dataset

Haoxuan Chen, Yiran Deng, Shuying Zhang

Introduction

Chinese, Japanese, and Korean have similar facial features partially due to their geographical similarities. Some people claim that they can differentiate these three subgroups of east asian based on how they look. But it is very hard to eyeball the difference.

Use **Machine Learning** methods to differentiate **Chinese, Korean and Japanese**

Key idea:

- Each Image is resized to **64×64** for CNN and **128×128** for other classifiers
- The dataset is divided into Chinese / Japanese / Korean(**3 classes**), then each subset is divided into male / female (**6 classes**)
- Dataset is randomly shuffled before training

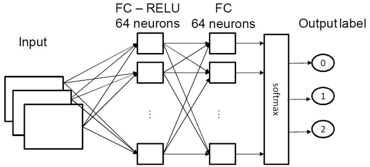
Classification methods

k-Nearest Neighbor:

- Simply store the training set, compare test images with all the images in the training set and gives it the majority label of k most similar training examples
- Use validation set to try different k, we have the best k = 8

Support Vector Machine:

- Linear SVM with hinge loss
- Use validation set to tune hyperparameters



Two-layer Neural Network:

- Inner structure: fully connected layer - ReLU layer - fully connected layer
- Simple implementation yields much better result than kNN and SVM

Convolutional Neural Network:

- Use the TensorFlow Library to build and train our convolutional neural net
- Two convolutional layers with a fully connected layer and a dropout layer

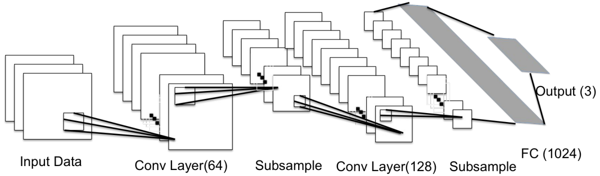


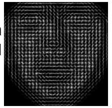
Image Processing Methods

Face cropping:

- Use Haar-Cascade Face Detection from OpenCV to crop faces from original images
- The face is cropped and positioned upright in the output

Feature extraction:

- Convert RGB images to grayscale, get the color histogram over the Hue of the grayscale image, compute the final Histogram of Oriented Gradient (HOG) feature



Mean subtraction:

- Get a "mean face" from training set, then subtract it from all images
- Equivalent to centering the data around the origin along every dimension

Data Collection

- A total of 1380 profile profile photos of university faculty members and famous celebrities from the three countries were used. We flipped the images and added a random brightness, therefore quadruple the dataset.
- The celebrity images were downloaded using the Google Custom Search API by using a person's name as the searching keyword.
- 80% of the images were used for training; 10% for validation; 10% for test.

Table 1. Number of Images Per Subgroup

	Chinese	Japanese	Korean
Male	835	1257	1119
Female	835	733	742

Results

2-layer NN:

(Params: Learning rate = 1e-4, regularization = 0.5, #iteration = 3500, batch size = 50, #neuron = 64)

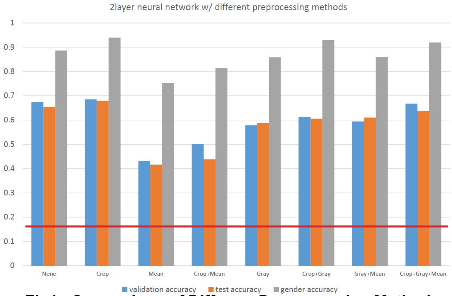


Fig1 . Comparison of Different Preprocessing Methods

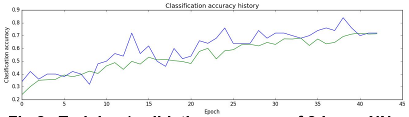


Fig 2 . Training / validation accuracy of 2-Layer NN

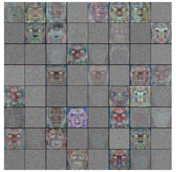


Fig 3 . Visualization of Weights in 64 neurons in 2-layer NN

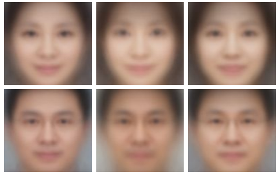


Fig 4 . (Not so) mean faces (C, J, K from left to right)

Table 2. Testing Accuracy of Each Method

	SVM	kNN	2-Layer NN	CNN
3 Classes	62.1%	57.5%	64.7%	89.2%
6 Classes	48.2%	50.9%	67.9%	---

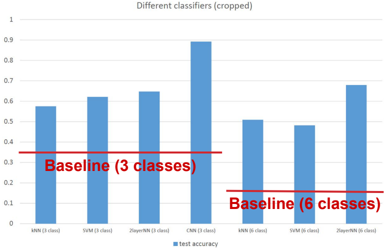


Fig 5. Accuracy Comparison of Different Methods

Future Works

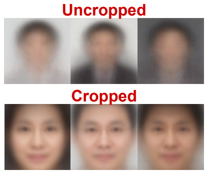
Improvement on failed trails / More tuning on hyperparameters:

- PCA and whitening are too slow (calculating the covariance matrix)
- Using HOG feature doesn't yield higher accuracy, since it mainly detects edges

Miscellaneous

Unsupervised Learning:

- Use k-means to autonomously divide the dataset into several clusters, we used k = 2 and k = 3



PCA:

- Reduce dimension, images are "blurred"

Whitening:

- Normalize the scale in every dimension