Face Recognition Using Convolutional Neural Network

Machine Learning Engineer Nanodegree
Capstone Project

Xinkun Chu
March 13, 2017

I. Definition

Project Overview

Face recognition is a hot topic in the field of computer vision. It has wide applications around us. Computers and smart phones recognize users' faces to unlock the devices. Smart cameras use it to focus on the faces in portrait photography. Facebook uses the technique to speed up tagging people in photos. “Swiping face” is considered more safe and convenient than “swiping card” where people can pay by simply taking a picture of the face. However, face recognition is far from perfect. Different lighting, makeup, decorates, change in age, change in facial expressions and low resolution image all make face recognition difficult. More efforts are needed on the topic to get face recognition technique further applied in everyday life.

The study on face recognition has a history of more than 50 years. The first stage dates from 1960 to 1990. Face recognition was studied as a branch of general pattern recognition, and mainly used geometric feature based method[1]. In the 1990s, face recognition came to the second stage with fruitful results. Researchers came up with new algorithms like Eigenface[2] and Fisherface[3]. Appearance based and statistical pattern recognition based algorithm showed good performance on well-controlled, front- face images taken in the lab. In the current stage, researchers are working to improve the robust and accuracy of face recognition under complex circumstance.

In this project, I’ll look into the convolutional neural network (CNN) based face recognition algorithm. The CNN model will be trained using the Labeled Faces in the Wild (LFW) dataset.

Problem Statement

The problem of face recognition can be generalized as:

Given input face images and a database of face images of known individuals, determine the identity of the person in each input images.

To identify a person from a raw image like the LFW dataset, the task contains two parts: face detection and face recognition. Face detection is beyond the scope of this paper. I'll use the facial point detection tool, provided by the paper.[4] The tool can detect the face region and positions of the five facial points in the face images, two for eyes, one for nose and two for mouth.

Then, I'll train CNN models for face recognition, based on the detected regions. CNN can automatically learn features to capture complex visual variations in the images. The person identity can be one-hot encoded as the label. The model can be trained with the face images of known individuals with labels, and calculate the probability of the person in test images belonging to each identity. The identity with the highest probability will be taken as the predicted identity for the person in the image.

Metrics

Precision and recall will be used as the metrics to judge the performance of the model on test dataset. Precision shows the class agreement of the data labels with the positive labels given by the model. Recall shows the effectiveness of the model to identify positive labels. The weighted precision and recall can be expressed as:

$$\text{precision} = \frac{1}{\mathrm{S}} \sum_{i=1}^{l} S_i \frac{TP_i}{TP_i + FP_i}$$

$$\text{recall} = \frac{1}{\mathrm{S}} \sum_{i=1}^{l} S_i \frac{TP_i}{TP_i + FN_i}$$

where $TP_i\ is$ the true positive for class i, FP_i is the false positive for class i, FN_i is the false negative for class i, l is the number of class, S_i is the support for class i in the test dataset, S is the sample size of the test dataset.

II. Analysis

Data Exploration

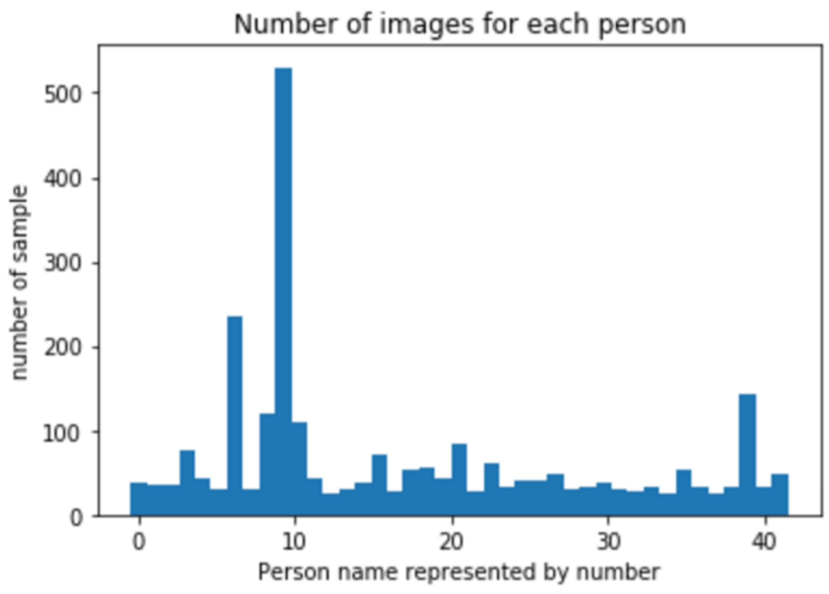
The whole LFW dataset contains more than 13,000 images of faces collected from the web, with each face labeled with the name of the person. The dataset represents people in a wide variety of settings, poses, expressions, and lighting. The data were checked and relabeled by hand, with duplicates removed. In the dataset, face images belonging to the same person are classified into the same directory, with the directory named by the person's name. The raw images are colored, and 250*250 pixels each. This project will use a subset of the data, with people pictured having a minimum of 25 face images.

We can check the statistical distribution for the number of images each person has.

count	mean	std	min	25%	50%	75%	max
5748	2.3	9.0	1	1.0	1.0	2.0	530

It's a highly right skewed distribution, with about 70% of the person has only one image. People with too few images cannot be used for face recognition task of this project. We will need different images of each person for training, validation and test. A subset of the data with a minimum of 25 face images for each person is selected, so there will be on average at least 15 images for each person to train the model. The person with the most face images has a number of 530 images. The corresponding name of the person is George W Bush. It's natural that the pre-president appeared often in news on websites during his tenure.

Exploratory Visualization



The plot above shows the distribution of the number of images for each person in the subset data, where each person has a minimum number of 25 face images. This subset data will be used in training and testing the CNN model. In the plot, the names of the 42 persons in the dataset are represented by numbers from 0-41. From the plot, the distribution is imbalanced. George W Bush, Colin Powell and Tony Blair, represented by number 9, 6, 39, has obvious more face images than others. The information is import in choosing the metric to judge the model’s performance. Accuracy alone can be biased, because of the “accuracy paradox” for imbalanced data.

Algorithms and Techniques

The convolutional neutral network model will be used as the classifier of face recognition in this project. CNN is widely used and performs well in the fields of image recognition, video analysis and natural language processing. Face recognition is a branch of image recognition, where CNN is applicable.

The basic architecture of the model is INPUT – [CONV - ReLU – POOL- DROP]n – FC – CL.

The input images will be in the shape of array (N, h, w, 1), where N represents the Nth image, h is the height, and w is the width of the image. ‘1’ means the grey scale image will be used. Though people may differ in skin or eye color, different lighting will make the situation complicated. It will be hard in training the color feature to differentiate people, based on this not large dataset. So grey scale is used for simplicity.

The convolutional layer is the core building block of the CNN. Each filter will compute a dot product between their weights and a small region connected to in the input volume. The main parameters to tune:

- ✧ Filters: number of filters in the convolution, defines the dimension of the output space
- ✧ Kernel size: the width and height of the 2D convolution window;
- ✧ Strides: the pixel to jump each time when we move the filters
- ✧ Boarder mode：valid or same. For ‘valid’, the convolution is only computed where the input and the filter fully overlap. Therefore, the output is smaller than the input. For ‘same’, the output is the same size as the input, which means the filter go outside the bounds of the input by half of the filter size.

ReLU is short for rectified linear units. It applies an elementwise activation function, such as the max(0,x), thresholding at zero.

The pooling layer will use the max pooling. It performs a down-sampling operation along the spatial dimensions and outputs the maximum of each sub region. The pooling layer can reduce number of parameters and help control over fitting. The main parameters to tune:

- ✧ Pool_size: factors by which to downscale (vertical, horizontal)

- ✧ Strides and boarder mode: same meaning as described above.

Dropout will drop out individual nodes with pre-set probability at each training stage to prevent overfitting. The method also improves the speed of the training. The combination of [CONV - ReLU – POOL- DROP] will be applied n times, based on the performance of the model.

Fully connection layer has full connections to all activations in the previous layer. The main parameter to tune is the output size of the layer. The classification layer is the last layer. The output size is the same as the number of classes in the data. It will use "softmax" for activation.

Benchmark

The example model of face recognition using eigenfaces and SVMs from scikit-learn[5] will be used as the benchmark model. The example model uses the labeled faces in the wild. It uses PCA for unsupervised feature selection and dimensionality deduction. Eigenfaces are the components of the PCA. Then it trains SVM using GridSearchCV for optimization.

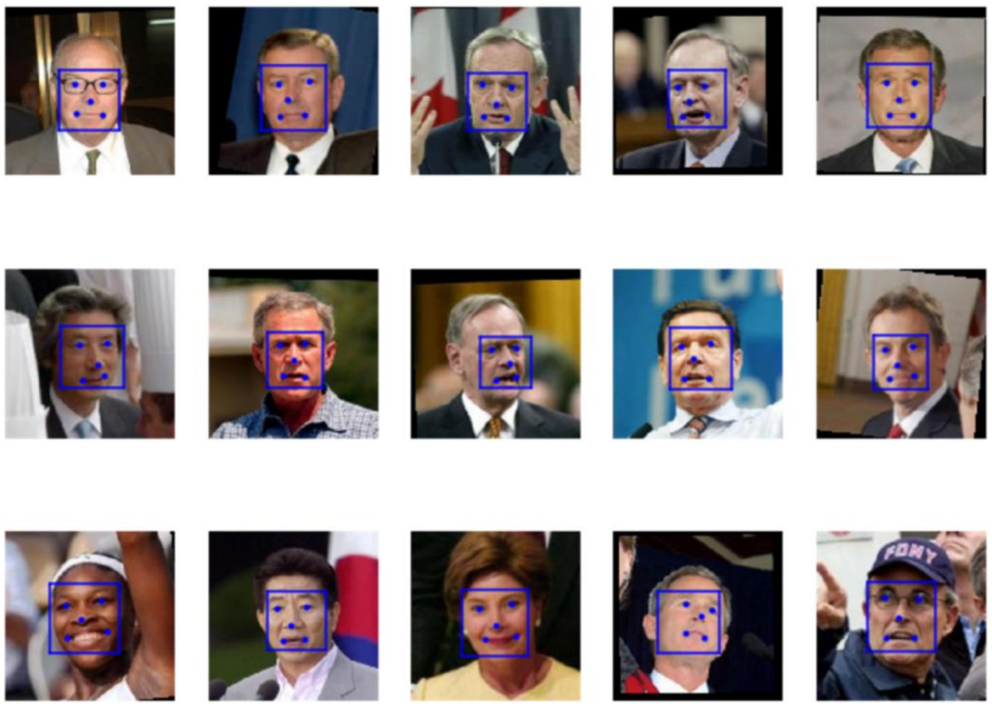
The example model only contain 7 people with images for each person more than 70. The averaged precision, recall of the model's predication for these people is around 80%. The example model will be extended to include more people with a minimum of 42 face images in the dataset. The precision and recall dropped to 64.7% and 62.4%, respectively.

Eigenfaces was the popular algorithm in the 1990s and achieved good performance with front-face images taken in the lab. It would be interesting to compare the performance of CNN model and Eigenfaces model on the more complex LFW dataset.

III. Methodology

Data Preprocessing

For preprocessing, the first step is to loop all the directories and return the file name with path if the person has more than 25 face images. As discussed in the problem statement section, the face detection tool in Ref[4] will be used to detect the bounding box of the face region and positions of the five facial points, two for eyes, one for nose and two for mouth. The image list is prepared as the input to the detection tool. If multiple faces are detected in the image, only the central one will be retained. The detection tool further uses the bounding box and the corresponding image to get the facial points and stores the positions in a binary file. And I use an array to read out the facial points positions for each person from the binary file. The plot below shows the calculated bound box and facial points on the raw image from the LFW dataset.

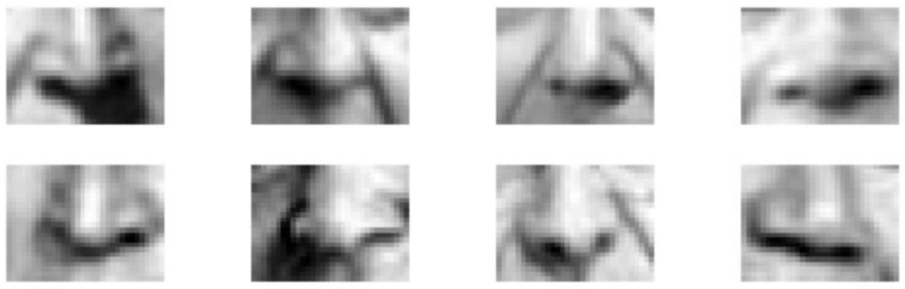


The following regions will be extracted:

a) The region within the bounding box. It's resized to (-1, 62, 47, 1).

b) The region around the eyes, as shown below. It's resized to (-1, 32, 64, 1).

c) The region around the nose, as shown below. It's resized to (-1, 24, 32, 1).



d) The region around the mouth, as shown below. It's resized to (-1, 24, 64, 1).

All the pixels in the crapped regions will be assigned the type as float32, and normalize to (0, 1) by dividing 255.

Each person's name will be grepped from the image file name. A number will be given for each unique name, and the person's name will be represented by numbers as the labels in training the model. Then, the labels will be one-hot encoded.

Then the data will be divided into training set, validation set and test set. In this project, we'll have 1,656 training samples, 414 validation samples and 518 testing samples. The LFW public dataset has already been well cleaned, so there is no need for further cleaning in this project. As discussed in the data exploration section, it's also reasonable that Geroge W Bush has obvious more face images samples than others.

Implementation

In the project, all the processes are documented in the Jupyter notebook. It includes the following steps:

1) Data preprocessing, as described in detail in the previous section. Several helper functions are defined:
    i. file_read_txt(…): write the person's image file names along with the path into a txt file, if the person has number of face images more than the minimum number defined by the user.
    ii. capture_images(…): if scale set True, given the txt file that contains file names and the corresponding bounding box positions, it will return the crapped entire face region and the names of the image in the format of array.
    iii. capture_eyes(…), capture_nose(…), capture_mouth(…): input txt file is the same as above. It will return the crapped regions around the eyes/nose/mouth of each image, in the format of array.

2) Build the network architecture and set the training parameters. I use the Keras to build the CNN model. Keras is a high-level neutral networks library running on top of TensorFlow. The basic architecture of the model is INPUT – [CONV - ReLU – POOL- DROP]n – FC – CL, as described in the Algorithms and Techniques section.
3) Compile the model with the loss function and optimizer. The loss function will

use categorical crossentropy, and the optimizer will use adam.

4) Train the model with the training dataset and check the performance on the validation dataset. Stop training when the loss stops decreasing.
5) Use the model to predicate on the test dataset. Calculate the precision and recall.
6) Step 2-5 will be performed separately for the entire face within the bounding box, the region around eyes, the region around nose, the region around mouth.
7) Compare the results and decides the best model.
8) Compare with the bench mark model.

Refinement

Initially, I tried to train a CNN model using the entire face image within the bounding box only. It achieved a precision of 81.2%, and recall of 80.3% on the test dataset. Then, I trained CNN model separately for the regions around the eyes, the nose and the mouth. The results are listed in the table below. The model trained with nose or mouth regions get precision and recall around 70%. The model trained with eyes region get a slight better result than the model with the entire face. Next, I use the Keras merge layer to combine the model 2, 3, 4, as model 5; combine the model 1, 2, 3, 4, as model 6. Model 5 got a precision of 88.9% and a recall of 87.3%. The performance of Model 6 is no better than Model 2. For each face image, the CNN model gives the probability of the image belonging to each person. I use the product of the probability predicated by model1-4:

$$P_i = P_{i，face} * P_{i，eye} * P_{i, nose} * P_{i, mouth}$$

eg. $P_{i，face}$ is the probability of the image belonging to person i predicted by the CNN model trained with entire face region only(model 1). The person with the highest probability will be taken as the predicted identity in the image. This model (model 7) gets 91.7% precision and 90.2% recall.

	entire face	eye	nose	mouth	Merge1	Merge2	Final
Model	1	2	3	4	5	6	7
Precision	81.2%	85.4%	73.0%	73.6%	88.9%	85.4%	91.7%
recall	80.3%	83.8%	72.6%	72.0%	87.3%	83.4%	90.2%

IV. Results

Model Evaluation and Validation

The final model, as discussed above, gets 91.7% precision and 90.2% recall on the test dataset. It uses the product of the probability predicated by model1-4, which trains the CNN model using the entire face image within the bounding box, the region around the eyes, the region around the nose, and the region around the mouth, separately. For each image, the person with the highest product probability will be taken as the predicted identity in the image. This model was chosen because it got the highest precision and recall among the 7 model under test.

The model1-4 are trained separately, and the CNN parameters for each model is optimized based on the performance of the model on the validation dataset. The model 1-4 follow the same module of [CONV - ReLU – POOL- DROP]n as described above. The convolutional layer uses 32 filters, the kernel size is 3*3, the border mode is 'same'. It will use ‘relu’ as the activation function. For the pooling layer, the max pooling is used. It use the pool size of 2*2. Then a dropout layer is set with drop out rate 0.5.

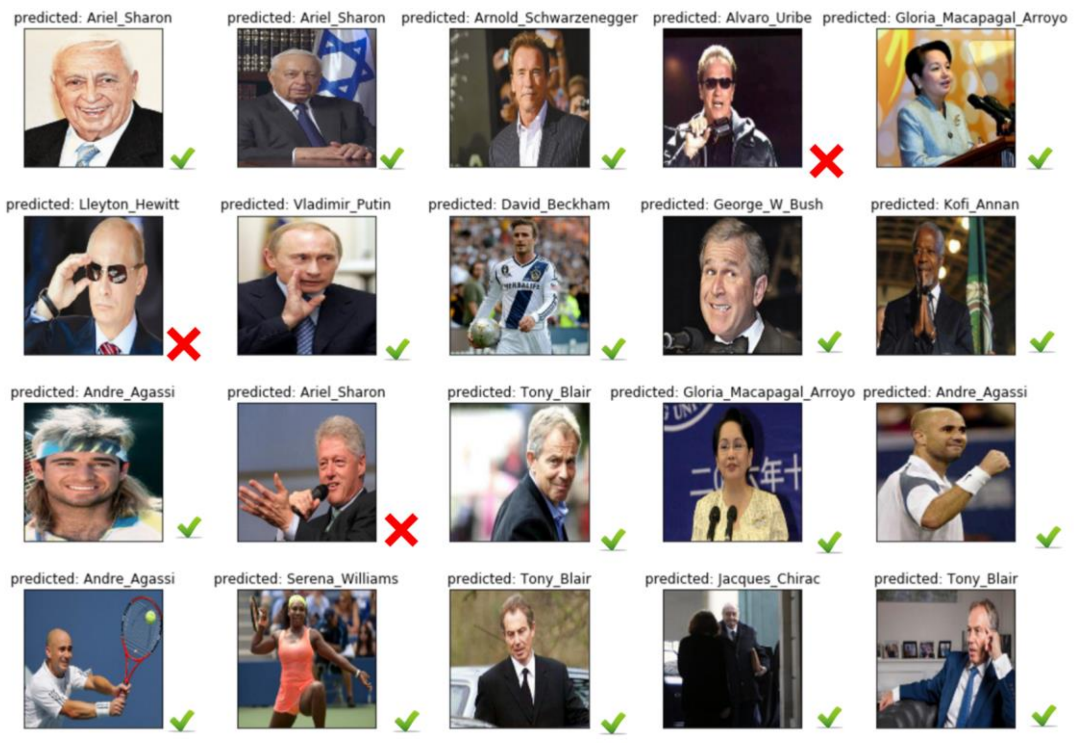
For mode1, the input shape is (62, 47, 1). It uses 3 modules, followed by a flatten layer, a fully connected layer with output size 1024, dropout layer with rate 0.5, classify layer with output size 42, activated by softmax.

For mode2, the input shape is (32, 64, 1). It uses 2 modules, followed by a flatten layer, a fully connected layer with output size 512, dropout layer with rate 0.7, classify layer with output size 42, activated by softmax.

For mode3, the input shape is (24, 32, 1). It uses 2 modules, followed by a flatten layer, a fully connected layer with output size 512, dropout layer with rate 0.7, classify layer with output size 42, activated by softmax.

For mode4, the input shape is (24, 64, 1). It uses 3 modules, followed by a flatten layer, a fully connected layer with output size 512, dropout layer with rate 0.7, classify layer with output size 42, activated by softmax.

To check the reliable of the model, I further downloaded 20 images from the internet. The persons on the images are contained in the dataset, but the images are different from the images in the dataset. The final model gets 17 out of 20 correct. The result is expected and agrees with the performance on the test dataset.

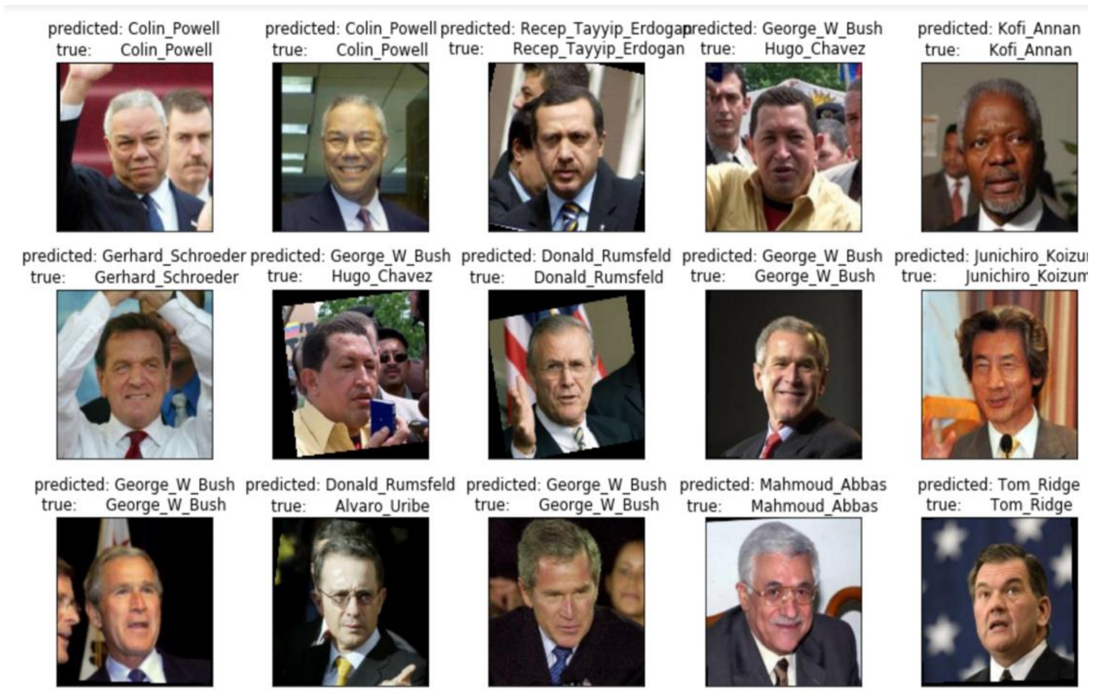


Justification

In this project, I use the LFW dataset with 25 minimum face images each person. The benchmark model using eigenfaces and SVM has precision and recall of 64.7% and 62.4%, respectively. Using the same dataset, my final CNN model has precision and recall of 91.7% and 90.2%, respectively. The result is much better than the benchmark model. Though the model is not perfect and there is still space for improvements, the model generally does well on face recognition with images taken from everyday life.

V. Conclusion

Free-Form Visualization



Examples of the final model's predication on the test dataset. Above each image, upper title is the identity predicated by the model, lower title is the true label of the person.

The picture above shows the model's predication and the true label on LFW images randomly selected from the test dataset. In the selected samples, two images from Hugo Chavez are misidentified as George W Bush. This may be caused by the fact that the two persons have similar face shape. And the model didn't capture enough details on the face in the training. For example, the model is trained with grey scale, so the information of hair/skin color is not utilized. Also, in the dataset, George W Bush has obvious more images than others. For any images with similar face shape to Bush, the model may have the tendency to identify the person as Bush. Another mistake in the examples above is taken Alvaro Uribe as Donald Rumsfeld. Checking the images of the two persons, one thing in common is that they both have glasses. Decorates especially glasses make the face recognition difficult. For images of Alvaro Uribe, he sometimes wears glasses, sometimes not; sometimes wears transparent glasses, and sometimes wears sun glasses. It may make the model confused what feature to capture from this person.

Reflection

The process of training the final model used in this project can be summarized as the following steps:

1. Download the LFW dataset

2. Data preprocess, facial detection tool is used to crap the entire face region, the regions around the eyes, nose and mouth.

3. Build the CNN model for each defined region, separately.

4. Train each CNN model separately with the same training dataset.

5. For each image, combine the probability of each person predicted by each model. The person with the highest will be taken as the predicted identity in the image.

For these steps, I found Step3-4 the most difficult. Because the face images for each person is not quite many, the CNN model can easily get over-fitted. At some epochs, the loss on the training dataset keeps decreasing while the loss on the validation dataset is fluctuating. A larger dropout rate is assigned and the model is re-trained. The learning rate also needs to change to a smaller value during the training. So the model can learn fast at the beginning, and optimize with small steps during the end of the training.

Improvement

In the project, several CNN models are compared, and the optimized model is used as the final model. However, there are still several improvements to make:

1. The CNN model can be trained with the color images. It's expected that skin color, hair color can help differentiate persons. But different lighting condition is a concern. A possible approach is to detect the average background lighting and get the relative lighting of the face images. This improvement can help improve the precision and recall of the current model by further differentiating person with different hair/skin colors.

2. The person in the image may tilt the head to one direction. The image may need to rotate a small degree to adjust the tilting. Otherwise, the CNN model may take the tilting images of the same person as different shapes.

3. The dataset of LFW is not large enough. CNN model trained with larger dataset can achieve better performance than the current model. In the training process, the CNN model can easily get over-fitted on the training dataset, but performs worse on the validation dataset. A large dropout rate is set as a balance between over fitting and speed of training.

[1] W Zhao, R Chellappa, PJ Phillips, A Rosenfeld, Face recognition: A literature survey, ACM computing surveys (CSUR) 35 (4), 399-458 (2003)

[2] M. Turk, A. Pentland , Face recognition using eigenfaces, Proc. IEEE Conference on Computer Vision and Pattern Recognition. pp. 586–591 (1991).

[3] Peter N. Belhumeur, Joao P. Hespanha, and David J. Kriegman, Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection, IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE, VOL. 19, NO. 7, JULY 1997

[4] Y. Sun, X. Wang, and X. Tang. Deep Convolutional Network Cascade for Facial Point Detection.

In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2013

[5] http://scikit-learn.org/stable/modules/generated/sklearn.datasets.fetch_lfw_people.html