A Facial Recognition System

Capstone Project

Machine Learning Engineer Nanodegree

Jianhua Li, Mar 12st, 2017

I. Definition

Project Overview

Face recognition, a form of computer vision, which uses the spatial geometry of distinguishing features of the face to identify or to authenticate a person. Software using webcam acquired images automatically log on an authorized user has been developed for many years by several companies. Computer access control system based on face recognition is also provided by three well-known computer manufacturers. Although it is convenient (hand-free), its security caused by the relatively accuracy is one of the major concerns compared with other biometric recognition systems. Nevertheless, it is still considered as a good supplement for current available solution.

The conventional face recognition pipeline consists of four stages which are face detection, face alignment, feature extraction and classification. One of the most import steps is the feature extraction. Conventional features include linear functions of the raw pixel values including Eigenface, Fisherface and Laplacianface. More recently, such linear features were replaced by hand-crafted features. Although hand-crafted features and metric learning achieve promising performance in constrained environment, the performance of using these features degrades dramatically in unconstrained environments where face images cover complex and large intra-personal variations such as pose, illumination, expression and occlusion.

In recent years, deep learning, in particular Convolutional Neural Network (CNN) has achieved very impressive results in face recognition. Unlike the conventional hand-crafted features, the CNN learning-based features are more robust to complex intra-personal variations. Indeed, the top three cases of face recognition in unconstrained environment (FRUE) reported on benchmark database LFW have been achieved by CNN. One of the advantages of CNN is that all processing layers have configurable parameters which can be learned from data, thus alleviate the burden of manual feature design. However, the number of parameters in CNN can be millions if not billions. To learn such a large number of parameters, a very large training datasets are required. In many cases, collecting large datasets can be very costly or even an impossible task.

To overcomes the limited size of datasets, two methods have been successfully applied to diverse recognition problems. For example, A 3D-aided face synthesis technique has been used for facial landmark detection and face recognition. Another commonly used technique is Data augmentation which applies mirroring, cropping, rotation, and scaling without changing semantic-level image label (reference 2 and https://arxiv.org/pdf/1603.06470.pdf).

Problem Statement

As the built-in cameras becomes more and more common in cell phones and personal computers, images and videos are piling up. The desire to understand what is in these images and thus to apply this information to facilitate our daily life becomes more and more strong. Computer vision is an interdisciplinary field and provides a powerful solution in high-level image as well as videos understanding.

As a data analyst, I interact daily with sensitive Protected Health Information (PHI). For security reasons, I am required to lock our computer whenever we leave our desk. When I come back to my desk, I have to unlock it to continue my work. There are multiple ways to do it. For example, one can type in the password through keyboard or use an external device to read identification information through smart ID card, but none of them are hand-free process. So, it is kind of annoying to repeat these locking/unlocking steps manually all the time. For this project, I propose to develop an auto-facial recognition system inspired by Hironsan's blog (reference 3). Specifically, this system uses the built-in camera in the computer to capture the image of a person in front of it. The captured image then is processed and routed to the auto-facial recognition system where a prediction will be made based on the input image. If the image is from an authorized person, a "password" message will be send to the computer so that it can unlock itself automatically, otherwise it will not do anything.

In this project, a CNN, the most recent deep learning face recognition method, is applied. Our goal is to achieve more accurate face recognition capability by carefully adjust both the CNN architectures as well as its parameters.

Metrics

To identify if an image from an authorized person, a 1:N authentication method is applied. Images of authorized person to use the computer are prepared by myself. Images of unauthorized users are collected from online LFW (reference 1 and http://vis-www.cs.umass.edu/lfw/). Images are classified into "authorized" and "unauthorized" two categories and labeled with authorized (1) and unauthorized (0). The captured image will then be processed with OpenCV. Decision will be made based on the prediction result from the supervised learning model.

Image data is divided into training, validation and testing three groups. To train the model, the keras python library is used with TensorFlow as the backend. A supervised learning model is generated with the state of art machine learning algorithm convolutional neural networks (CNN). For the final classification, the softmax classifier is used. This classifier is a kind of When training the model with CNN, the commonly used cross-entropy loss function is used.

$$L_i = -\log\left(\frac{e^{f_{yi}}}{\sum_j e^{f_j}}\right)$$

To measure performance of the model, accuracy is used to evaluate the performance with the validation dataset and then test with testing data. The final model is further evaluated with 20 pieces of new images.

$$accuracy = \frac{true\ positives + true\ negatives}{dataset\ size}$$

II. Analysis

Data Exploration

To reduce the labor, I have taken the advantage of the online resource. Images used for "unauthorized" category are downloaded from the website of computer vision lab at the University of Massachusetts (reference 1 and http://vis-www.cs.umass.edu/lfw/). This image data set contains more than 13,000 images of faces collected from the web. Each face has been labeled with the name of the person pictured. The images are packed as a 173 MB gzipped tar file. The zipped file is unzipped with the 7-zip software downloaded from its official website (http://www.7-zip.org/) and saved on a ../unauthorized folder. It has been known that there are some mismatches between the picture and the labeled name. This should not be a problem for the current project.

Images used for "authorized" category are prepared by myself either using a cell phone or a built-in camera on the notebook. All images are taken at natural color with standard contrast. Images from these two sources have different size settings. Images manually taken are pre-trimmed with photoshop as 250 pixel x 250 pixel. Due to the time and labor limitation, total 333 images were prepared. To further enrich the "authorized" image pool, images were automatically derived from a set of self-recorded videos (480 pixel x 600 pixel) by using OpenCV. All videos are taken with the faces at approximate center position. Backgrounds are mostly from an indoor setting which mimics the office environment. Images were picked every 2 to 10 frames depends on the variation of the background, illumination as well as the expression. Images were then cropped at [0:480, 60:540] and further resized as 250 x 250 x 3 format with INTER_AREA interpolation. The prepared images were saved on a ../authorized folder.

For initial image data exploration, Ipython.display library and OpenCV (opencv.org) are used. Images are kept in good condition and can be shown with IPython.display library (see Figure 1 and ImageExplore.ipynb file). All images shape is 250 x 250 x 3 with the size 187500. There are 10,843 images belong to "authorized" category and 13,233 images belong to "unauthorized" category.

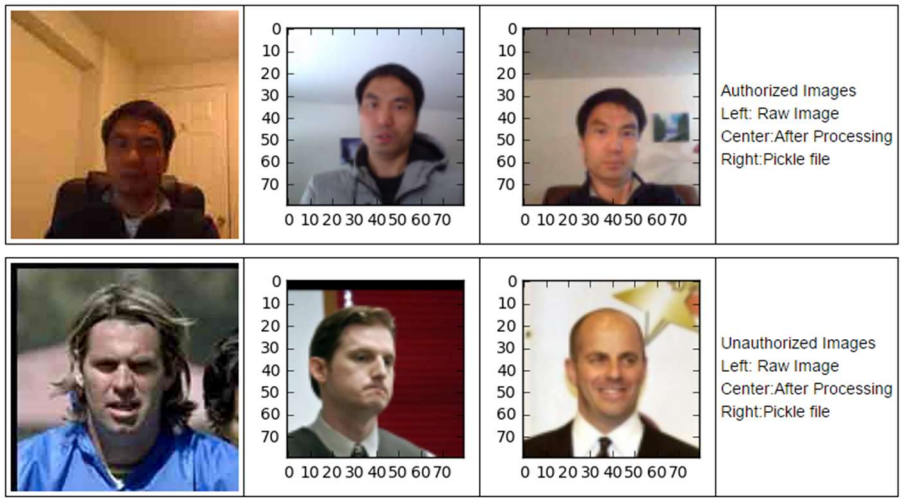

Figure 1. Images Exploratory

Top row: authorized images in different processing stages. Bottom row: unauthorized images

Exploratory Visualization

Raw images from either "authorized" or "unauthorized" categories are randomly picked for visualization. As we can see that images from both categories are properly displayed with normal color scheme (Figure 1 and ImageExplore.ipynb file).

Images after processing and resizing were further visualized using pyplot image. All images were shown properly with expected size and color scheme (Figure 1 and ImageExplore.ipynb file). Images data saved as pickle files were further converted back as image and can be also displayed normally with appropriate color scheme (Figure 1 and ImageRawdataProcessing.ipynb file). It is worth to note that some of the images downloaded from LFW have been shifted as you can clearly see thick black edges (Figure 1).

Algorithms and Techniques

In this project, I have applied Convolutional Neural Networks (CNN) classifier which is a state-of-the-art algorithm for image classification, object detection and face recognition. Although CNN is particularly useful for training model from image data, the drawback associated with CNN is that it requires a large number of training data for learning a deep image representation.

To overcome the inherited problem, data augmentation and data synthesis are commonly used for face recognition. For example, using the synthesized large training datasets, Hu et. al., (reference 2) improve the face verification rate from 78.97% to 95.77% on LFW using CNNs.

A CNN consists of several layers which can be classified into three groups:

1. Convolutional layer
2. Max-Pooling Layer
3. Fully-Connected layer

The following parameters can be tuned to optimize the classifier:

- Classification threshold
- Training parameters
  - Training length (number of epochs)
  - Batch size (how many images to look at once during a single training step)
  - Solver type (what algorithm to use for learning)
  - Learning rate (how fast to learn; this can be dynamic)
  - Weight decay (prevents the model being dominated by a few “neurons”)
  - Momentum (takes the previous learning step into account when calculating the next
- Neural network architecture
  - Number of layers
  - Layer types
    - Convolutional Layer
    - Fully-connected Layer
    - Pooling Layer
  - Layer parameters

- Preprocessing parameters
  - image size
  - dimension

Benchmark

Using the synthesized large training datasets, Hu et. al., (reference 2) has reported that the face verification rate to 95.77% on LFW using CNNs.

III. Methodology

Data Preprocessing

To reduce the data set size, raw images were further resized into a smaller format (80 pixel x 80 pixel x 3) with OpenCV with INTER_AREA interpolation. Images data for both categories were read into separate numpy arrays and saved as pickle files named as authorized_image_data.pickle and unauthorized_image_data.pickle, respectively.

Image data then extracted from corresponding pickle files and corresponding labels were added for each image depends on which category it originates from. For example, an image from a "authorized" folder is labeled as 1 while an image from a "unauthorized" folder is labeled as 0. The image data is then converted into numpy arrays.

To train a supervised model, data is randomly divided into training and testing groups with test size set as 0.25. For validation purpose, the training data set is further divided into training and validation groups with train size set as 0.7. All image data is presented as float32 data type and is centered by dividing by 255. Image label vectors were converted into binary matrics. Input shape for image data is 80 x 80 x 3 which represents 19200 features.

In total, there are 24076 pieces of images with 45% are "authorized" images. After dividing, there are 12639 train samples, 5418 validation samples and 6019 test samples. All three groups have similar mean value (~ 0.42) and standard deviation (~ 0.27).

Implementation

In this project, we have selected the convolutional neural network (CNN) which has been proved particularly useful for imaging classification and face recognition. The architecture of CNN has been tuned many times. I found the architecture with 8 layers returns the best accuracy. Below is the detailed 8 layers (denoted as L1, ..., L8) which starts with input image and ends with softmax loss layer (Figure 2).

Following each Maxpooling, a drop-out with probability of 0.25 is applied. For the fully-connected dense layer L7, a drop-out with probability of 0.5 is applied. The final full-connected dense layer takes the outputs of L7 as its input, produce outputs equal to 2 classes through a fully connected architecture, then passes these outputs through a softmax function. With softmax loss layer, each input image is expected to have only one label.

In order to make the deep neural network training more robust to facial pose variation, real time data augmentation is used. Specifically, I have randomly rotated image in the range 0 to 20 degree and

randomly flipped image vertically. Since some of the images collected from LFW were shifted, to avoid any duplication, width_shift and height_shift have been disabled during data augmentation. I have also arbitrary set the channel_shift_range to 0.2 and fill_mode as "nearest".In total, there are 9,422,882 trainable parameters.

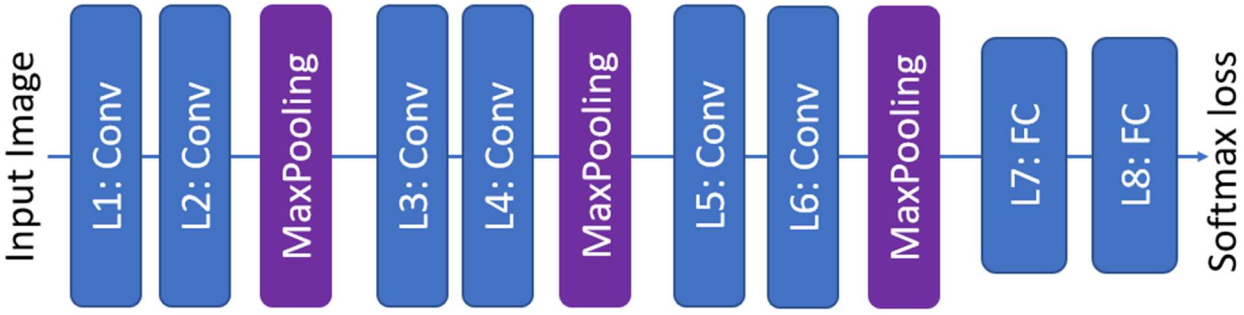

Figure 2. Architecture of CNN

Conv: Convolutional layer, FC: Fully connected layer

Refinement

Initial tuning has been focus on the epoch number, the problem associated with epoch number is that it takes longer time to test larger epoch number. so I have stick to epoch number with no more than 40. Although as epoch number increase, we usually get better accuracy, it always over shot and miss the best epoch number. After the initial testing, I turn to early stopping method by setting the callbacks argument value equals earlyStopping with 'val_acc' as the monitor.

To find the best combination of batch size and optimizer, I have tested batch size ranging from 40 to 90 with Adam, sgd or adamax optimizer. Unfortunately, larger batch sizes are beyond the capability of my computer. The arguments of each optimizer were set as default as indicated in keras document. The result suggested that optimizer sgd is very sensitive to the batch size. The accuracy of train data set with sgd optimizer ranges from 84.72% to 96.65% when batch size changing from 40 to 90. On the contrary, both adam and adamax are not very sensitive to the batch size, the accuracy of train data set for both optimizers range from 98.66% to 99.38%, and from 98.85% to 99.21%. The difference among the accuracy of train data set for both optimizers are within 1.0%.

When compiling the model, I have tested two loss functions which are categorical-cross entropy and binary_crossentropy. I have found that the former returns better accuracy. I have also tested two metrics which are accuracy and fmeasure. It turns out that accuracy returns better result.

IV. Results

Model Evaluation and Validation

When training, models were further cross validated with validation data set. When evaluate the model, accuracy was selected as the default metrics. From the returned verbose information, we can see that accuracy of validation data set shows similar trend as the accuracy of training data set. When using optimizer sgd, the accuracy of validation data set was sensitive to the change of batch size (from 84.03%

to 97.73%), while using either adam or adamax optimizer, the accuracy of validation data set hardly changed (adam: from 98.93% to 99.59%, adamax: from 98.91% to 99.54%). Of all optimizers I have tested, batch size 40 returns the best accuracy for validation data set, and adam optimizer returns the best accuracy for validation data set. It is worth to note that the accuracy returned by adam and adamax optimizers does not show significant difference.

Justification

The optimized models were further evaluated with test data set. As I expected, the accuracy for the test data set shows the similar trend as the accuracy for both validation and training data set. When using sgd and adamax as the optimizer, batch size 40 returns the best accuracy for test data set. However, when using adam as the optimizer, batch_size 70 returns the best accuracy for test data set although batch_size 40 has the best accuracy for validation data set. With adam as the optimizer and 70 batch has the best accuracy for test data set among all test conditions.

Finally, to verify the robustness of the model, I have tested the best model (optimizer: adam, batch size: 70) with 20 new images which have 10 authorized and 10 unauthorized images. It is worth to mention that the time points of authorized images were span for about 10 years with different background. Unauthorized images were collected from online. The person of the unauthorized images either were at similar age or have similar facial sobel as the authorized person. Interesting, this model predicts all images correctly with 100% accuracy (Figure 4 and PredictNewImage.ipynb).

V. Conclusion

Free-Form Visualization

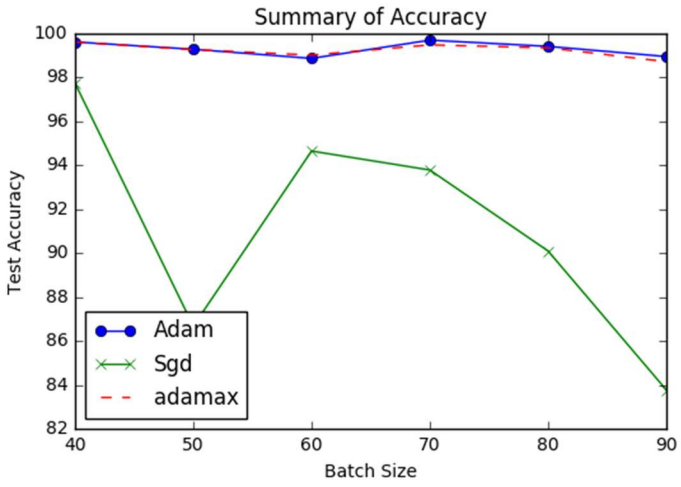

Figure 3. The summary of accuracy of test data

Test accuracy (shown in percentage (%)) predicted with model trained under different combination of data batch size ([40-90]) and optimizers (adam, Adamax, and Sgd). The accuracy derived from Sgd optimizer is very sensitive to the batch sizes. On contrast, the accuracy derived from adam and adamax optimizers is not sensitive to the batch sizes.

Figure 4. Prediction of new Images with the final model

Top row numbers show the predicted values of the top row corresponding authorized images. Bottom row number show the predicted values of the bottom row corresponding unauthorized images. NOTE: all images were correctly predicted with the model trained with batch size at 70 and a adam optimizer.

Reflection

In this project, I have established a face recognition model with the state-of-the-art CNN algorithm. The accuracy of the test data set is 99.70%. The whole process of this project can be divided into the following steps:

1. Problem initialization

2. Working Environment preparation

3. Data collection and exploration

4. Data processing and splitting

5. Model training and Parameter tuning

6. Model testing

Among all these steps, the step 2, 3 and 5 are most challenging for me.

To set up the working environment, I need to install TensorFlow with GPU in my Windows 10 desktop. To accelerate the video image capturing and image processing, I have also installed OpenCV and dlib. These installation steps are trivial but very time consuming and sometimes can be very desperate. Thanks for the internet, I am happy that I am able to make them work.

As for Data Collection, it is also a time-consuming step. Originally, I proposed to use images from Faces94 (http://cswww.essex.ac.uk/mv/allfaces/faces94.html). These images contain only the head part. Moreover, the background for all images is basically same with pure green color background. I decided to give up using these images because they may not represent a real-world situation. Fortunately, I was able to find the images from [LFW](http://vis-www.cs.umass.edu/lfw/) with more diverse background and variation. So I decided to use this image database as my control (unauthorized) data. To collect the authorized image data, I initially manually took my photos with camera which is a very time-consuming step. It seems a daunting task to collect over 10000 images manually. After googling on the internet, I was able to find that OpenCV enable one to automatically capture image from videos. So I then turn to this automatically procedure. Finally, I have collected more than 10000 images

from 20 shot videos taken with different dress styles, background and emotions. I think these images should be more close to a real-world scenario.

Finally, the model training and tuning step is one of the most challenging and rewarding parts. As a student starting before Oct, 2016. I did not learn any deep learning knowledge in our machine learning nanodegree program. So I have to learn it myself. I have followed the CS231n: Convolutional Neural Networks for Visual Recognition course (http://cs231n.stanford.edu/) offered by Stanford and Deep Learning (https://classroom.udacity.com/courses/ud730) course offered by Udacity. For parameters tuning, it was quite complicated with so many parameters to be tuned and a not so powerful computer.

Improvement

Although the accuracy of the test data set was reached up to 99.70%, it might not be true when applying this model with more diverse images or exposing it to more robust testing. To achieve better accuracy and more reliable models, the following methods might be worth to give a shot.

- Collecting more data. As we all know that the key point of training a good CNN model is the data size, putting more training data should has positively results on this regard.
- Using synthesis images. As more than 10000 images for each category in hand, one can use these images as the start point to synthesize more images with synthesized background as has been reported by Hu (reference 2).
- Cropping only face landmark. This method could be very technically challenging and the face landmark captured may not be a real face.
- Using more powerful computer. This method might be very costly for a student but should not be a problem for a dedicated company.

VI. References

1. Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments. *University of Massachusetts, Amherst, Technical Report 07-49*, October, 2007.
2. Guosheng Hu, Xiaojiang Peng, Yongxin Yang, Timothy Hospedales, and Jakob Verbeek. Frankenstein: Learning Deep Face Representations using Small Data. 2016 (https://hal.inria.fr/hal-01306168)
3. Hironsan. Deep Learning Enables You to Hide Screen when Your Boss is Approaching (http://ahogrammer.com/2016/11/15/deep-learning-enables-you-to-hide-screen-when-your-boss-is-approaching/)