Analysis of Face Recognition Methods

Ryan Karl
University of Notre Dame
Notre Dame, IN
rkarl@nd.edu

Pat Tinsley
University of Notre Dame
Notre Dame, IN
ptinsley@nd.edu

ABSTRACT

ACM Reference Format:
Ryan Karl and Pat Tinsley. 2020. Analysis of Face Recognition Methods. In . ACM, New York, NY, USA, 3 pages. 10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

For our project we investigated different modern face recognition techniques, specifically those based on Neural Networks and Local Binary Patterns to analyze their strengths and weaknesses under real world conditions. To test our methods, we downloaded the nd1C00002 dataset which contains 12,004 faces of 333 unique individuals and added 45 images each of ourselves.

2 LOCAL BINARY PATTERNS

Initially, the data set was split into positive samples (of ourselves) and negative samples (the remainder of the nd1C000002 data). Every image was read, resized, and converted to gray scale before running the Haar cascade-based face detector. [1] After the face detection phase, local binary patterns were computed using Python's scikit-learn package. [9] The local_binary_pattern method takes four arguments: the face crop, the radius, the number of points, and the method to determine the pattern. The last three had considerable effects on the model accuracy and were consequently studied very carefully. A grid search methodology revealed that the best parameters were radius 3 with 24 points using the 'nri-uniform' method; this method features non rotation-invariant uniform patterns.

For each of the images, a label was assigned; either 'PT' (for Patrick Tinsley), 'Ryan' (for Ryan Karl), or 'Unknown'. The positive samples (labeled 'PT' or 'Ryan') had 90 such images while the negative samples had 12004 images. After extracting the local binary patterns associated with each image, the data was split into training and testing, and a random forest classifier was fit to the training data. Support vector machines were explored, but ultimately abandoned due to poor performance. Additionally, decision trees and ensemble methods from scikit-learn were tested, but provided lack-luster precision and accuracy. The below results show the classifier's accuracy on the test data. Additionally, a confusion matrix shows the model's ability to generalize. However, when this model was deployed to detect faces in real time using the webcam, the accuracy suffered; most face detections were labeled as 'Unknown'. The next step of this project is to perform principal component analysis on our data feature set to discern which features are identity-salient.

	precision	recall	f1-score	support
PT	0.33	0.38	0.35	8
Ryan	0.50	0.56	0.53	9
Unknown	0.97	0.96	0.97	181
accuracy			0.92	198
macro avg	0.60	0.63	0.62	198
weighted avg	0.92	0.92	0.92	198

```
array([[  9,   0,   0],
       [  1,   4,   0],
       [  1,   0, 183]])
```

3 DEEP LEARNING METHODS

Feature Extraction Technique

For face detection, our method relies on a Histogram of Oriented Gradients based approach. [2] After inputting an image, we first convert it to gray scale, and then partition the image into 16x16 squares of pixels. In each square, we iterate over each individual pixel to analyze how dark it is in relation to neighboring pixels. After computing the gradient for each pixel to measure the flow of pixel intensity across the image in each square, we compute how many gradients point in each major direction (how many point right, etc…), and then replace the square in the image with the gradients that were the most common. After this procedure, we can scan each image and search for the section that looks the most similar to a known HOG pattern.

To account for noisy face images, which show faces at an angle or irregularly rotated, we apply a face landmark estimation algorithm that trains the network to locate 68 specific points that exist on any face — the top of the chin, etc. [5] Then we train a machine learning algorithm to be able to find these 68 specific points on any face. We can then apply affine transformations to face images (such as rotation, etc.) to center the facial components for classification. We decided to use the HOG based approach, because it is generally robust to changes in lighting. If we analyzed pixels directly, pictures of the same person would have radically different pixel values

under different levels of illumination. By only considering the gradient, regardless of the level of light, the same face should generally have the same encoding. [11] We also chose this method over other established methods such as Viola-Jones, because the HOG method is newer and has been observed in practice to outperform the Viola-Jones method. [7, 10]

Classification Technique

To train our Convolutional Neural Network for face recognition, we make use of a technique called deep metric learning to train it to generate 128 measurements (a real-valued feature vector also called an embedding) for each face. After loading an image from our training data, we then load another picture of the same person and a third picture of a different person. The algorithm then computes the measurements generated for each image, and adjusts the CNN so that the measurements generated for the images of the same person are closer, but the measurements for the second and third image are a further distance from each other. [8] After iterating over the training process many times the network eventually learns a unique mapping of embeddings to faces attributed to an individual person in the training data. Please note that we used a pre-trained network for our project, specifically, a ResNet network with 29 convolutional layers, based on the ResNet-34 network, that was trained on a dataset of roughly 3 million faces from 7485 unique individuals. [3] This model has an accuracy of 99.38% on the Labeled Faces in the Wild benchmark. [4]

We ran our own face images through their pre-trained network to get the measurements for our faces. After this process was complete, to search for the identity of the person's face in our face database who had the closest measurements to the test image during classification, we applied a k-NN model with majority voting to find the most likely identity of the face. This classifier was trained to take in the measurements from a new test image and determine which person in the database had the most similar measurements. We computed the Euclidean distance between the embedding and all faces in the training dataset. If the distance was below the threshold, then the faces matched and we returned True, but if the distance was above the tolerance threshold, the faces did not match and we returned False. We chose to use a k-NN based classifer due to its easy interpretability in our feature space. This was useful when training and testing our method, as we could better understand the distribution of different faces within the feature space and which images were found to have similar features to a class to which they were wrongly classified. Also, the k-NN algorithm is known to have a fast run time. [6]

Evaluation

To test our method, we ran the images from the nd1C00002 dataset (available at https://cvrl.nd.edu/projects/data/) through the framework to see if any were falsely classified as either of us, or if the system correctly classified them as unknown. We also ran 45 new images of each of us through the system to measure whether they were positively classified as one of us or not. Below is a confusion matrix that describes the accuracy of our CNN based approach. Note that the Accuracy was 0.9797, the Precision was 0.2673, the Specificity was 0.9797, the Recall was 0.9889, and the F1 score was 0.4208.

Prediction Outcome

		p	**n**	**Total**
Actual Value	**p′**	True Pos. 89	Fal. Neg. 1	P′
	n′	Fal. Pos. 244	True Neg. 11,760	N′
	Total	P	N	

Evaluation Discussion

In general, we observe that the CNN based method is highly accurate, and almost always classifies a known face from the training data to its owner. However, there is a large number of false positives, and this suggests that the features associated with each of our faces are too general and the network has not yet learned the unique features associated with our faces. Our method could perhaps be further improved if we gathered more training data in more adverse lighting conditions and at atypical angles to help teach the network how the features appear in noisy conditions. After further inspecting our training data, we noticed that most of the images have a face that is centered in the picture, and the subject is well illuminated. In the future, to improve the overall generalization ability of our method, we intend to increase the amount of training data of each of our faces under harsher conditions to better train the network to differentiate between our faces and those of other people.

4 DEEP LEARNING CODE

The python 2.7 code for this project is available at https://github.com/RyanKarl/Face_Classifier. Our project was built mostly using the dlib, OpenCV, imutils, and face_recognition libraries, but the full list of python packages on our system

is listed in the README. To run the Feature Extractor run the following command (quit with SPACE):

python encode_faces.py –dataset dataset –encodings encodings.pickle

To run the Face Classifier run the following command (quit with SPACE):

python recognize_faces_image.py –encodings encodings.pickle –image examples/True/IMG_1907.jpg

5 GROUP WORK

Ryan Karl worked on the Deep Learning portion of this paper while Patrick Tinsley worked on the LBP portion.

REFERENCES

[1] Timo Ahonen, Abdenour Hadid, and Matti Pietikainen. 2006. Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 12 (2006), 2037–2041.

[2] Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Identity mappings in deep residual networks. In *European conference on computer vision*. Springer, 630–645.

[4] Gary B. Huang, Manu Ramesh, Tamara Berg, and Erik Learned-Miller. 2007. *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*. Technical Report 07-49. University of Massachusetts, Amherst.

[5] Vahid Kazemi and Josephine Sullivan. 2014. One millisecond face alignment with an ensemble of regression trees. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 1867–1874.

[6] Sotiris B Kotsiantis, I Zaharakis, and P Pintelas. 2007. Supervised machine learning: A review of classification techniques. *Emerging artificial intelligence applications in computer engineering* 160 (2007), 3–24.

[7] Rainer Lienhart and Jochen Maydt. 2002. An extended set of haar-like features for rapid object detection. In *Proceedings. international conference on image processing*, Vol. 1. IEEE, I–I.

[8] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, et al. [n.d.]. Deep face recognition.

[9] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research* 12 (2011), 2825–2830.

[10] Paul Viola, Michael Jones, et al. 2001. Rapid object detection using a boosted cascade of simple features. *CVPR (1)* 1, 511-518 (2001), 3.

[11] Qiang Zhu, Mei-Chen Yeh, Kwang-Ting Cheng, and Shai Avidan. 2006. Fast human detection using a cascade of histograms of oriented gradients. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, Vol. 2. IEEE, 1491–1498.