Facial Recognition using HOG and Neural Networks

Gaurav Bhambhani
Computer Science Department
Nirma University
Ahmedabad, India
18bce072@nirmauni.ac.in

Ishan Tewari
Computer Science Department
Nirma University
Ahmedabad, India
18bce080@nirmauni.ac.in

Labdhi Sheth
Computer Science Department
Nirma University
Ahmedabad, India
18bce101@nirmauni.ac.in

***Abstract*—In this paper we have proposed a novel method for solving the challenge of Face Recognition using Histograms of Oriented Gradients (HOGs) which has been proven to be an effective descriptor for face recognition in general.It has recently gained a lot of attention since it can also be incorporated in low-cost digital cameras and computers, as well as because of its applications in biometrics and surveillance.**

***Index Terms*—Face Recognition, HOG, ResNet, CNN, Face Detection, Viola Jones, Histogram of Oriented Gradients, kNN.**

I. Introduction

The histogram distribution of oriented gradient directions is used as a feature in the HOG feature descriptor. Since the magnitude of gradients is high around edges and corners, and since edges and corners pack in a lot more knowledge about object shape than flat regions, gradients (x and y derivatives) of an image are useful. As humans, our brain is wired in a way that it recognises faces automatically and almost instantly. Computers, however do not possess such a high-level of generalization, and hence have to be taught this procedure of recognising faces, separately, by building a pipeline which chains together several different machine learning algorithms in order to make a prediction. The basic pipeline architecture for detecting faces is: Finding all the faces present in the image, which is done by converting the image into black and white since we don't require color data to detect faces, then scan through each and every pixel and the pixels surrounding it to know how dark that pixel is compared to the pixels surrounding it which gives us the gradient. Doing this is really advantageous as really dark and really light images of the same person give out the exact same representation. After isolating the faces in the image, we analyze all the data of the facial features we can gather, by wrapping the eyes and lips in each picture so that they always are fixed on the same sample place, using the concept of landmarking, where we mark 68 specific points that exist on every face, which makes it a lot easier for us to compare faces in the next few steps.We now compare the image given as input against all the other known faces, where we extract a few basic measurements from each face and find the known face with the closest measurements. Finally, we make a prediction, which is done by simply finding the image of the person with the closest measurements to the input/test image.

II. Related Work

The task of Face Recognition is basically divided into 2 parts.

1) Face Detection
2) Model Training and Prediction

A. Face Detection

Face Detection can be defined as the task of detecting the location of one or multiple faces in any given image. Face detection could be a tough task in image analysis that has on a daily basis a lot of and a lot of applications. It's one of the primary and therefore the most significant steps of Face Analysis. Face detection isn't straightforward as a because of its countless variations of appearance of the image, like pose variation, occlusion, image orientation, illuminating condition, etc. [10] There are many methods for face detection out of which, 3 are mentioned here.

1) Model based on Skin Colour: Detection of skin color in color images is a very popular and useful technique for face detection. Color is an important feature of human faces [1].

- Step-1: Convert the RGB Image to HSV Image as it works better where colour description plays an important role [3][4].
- Step-2: After converting into HSV, we phase the colour image into skin and non-skin region. Different color areas have different ranges of pixels that represent skin region and non-skin region [3][5].
- Step-3: After segmentation procedure, morphological operators are enforced with a structuring part.
- Step-4: After applying the morphological operators [6], the standard deviation of the area is calculated and rectangles are drawn in the skin regions. If by chance, any unwanted regions are detected, then it is then removed.

Advantages:

- This technique is in a able to properly find all faces within the pictures with nearly at right scale.
- It is quite robust to noise and form variations.
- It has an accuracy of about 80-82

Disadvantages:

Fig. 1. RGB to HSV Conversion

- Moderate false detection rate.
- Sometimes non face complexion region is additionally detected.
- Several objects have skin-tone colours, like some styles of animal skin, sand, wood, fur, etc., which might be erroneously detected.

2) Viola Jones Face Detection:: The Viola–Jones object detection framework [2] is an object detection framework which offers sturdy and competitive object detection rates in real-time. It was proposed in 2001 by Paul Viola and Michael Jones.

- Step-1: We create a new representation of an image [2], known as the integral image, which allows the features used by the detector to be computed very rapidly.

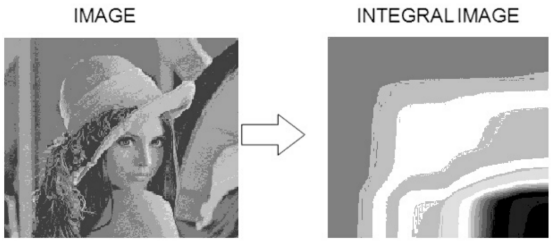

Fig. 2. Conversion to Integral Image

- Step-2: From the huge amount of features computed in the Step-1 we want to select only a few features that would enable us to detect faces with great accuracy. For this, we use Adaboost Algorithm [7] to select principal features and to train classifiers that would be using them.
- In the third step, we combine a lot of advanced classifiers into a cascade structure [9] that dramatically increases the speed of the detector by focusing attention on promising face-like regions of the image.

Advantages:

- Speedy image processing with high detection rates.
- Higher accuracy.
- Quite low false positive rate.

Disadvantages:

- Less effective on non-frontal faces.
- Quite long training time.

3) Model based on Histogram of Oriented Gradients (HOG): This model uses the concept of histogram orientation from a region of high brightness to low brightness. Hence the name Histogram of Oriented Gradients.

- Convert the image to Grayscale as we don't need colour data to find faces [9].

Fig. 3. Conversion to Grayscale [9]

- For every pixel, our goal is to make an arrow showing in which direction the image grows from brighter to darker.
- As saving the detail for every pixel becomes quite costly, we do it for a square of 16X16 instead [9].

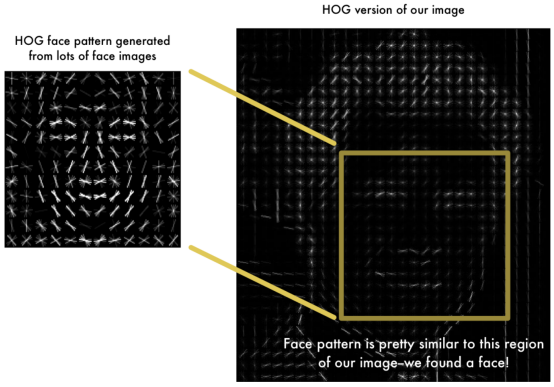

Fig. 4. Converted HOG Image [9]

Advantages:

- Quite fast as low processing is required.
- Fairly Accurate

Disadvantages:

- Not one of the most accurate ways to detect faces.
- Lowered efficacy on non-frontal faces.

As seen above, after carefully examining the advantages and disadvantages of many methodologies, we went with the HOG Method as it was fairly accurate and took lesser time for detection.

B. Model Training and Prediction

We considered two types of training for our network.

1) **CNN Based Model:**
   - In a CNN based model, the ResNet34 Architecture [12] can be used in combination with the Viola-Jones Face Detector. We can use this model which was trained on Labelled Faces in the Wild (LFW) dataset [10].
   - This trained model is then used to predict to whom the face belongs to.
   - It gives an accuracy of about 99.38
2) **Classifier Based Model:**
   - In the Classifier Based Model, possibly the features can be detected using the HOG method to detect the face and then applying some feature detection on it.
   - These features are then used to classify to whom the face belongs to.
   - Classifiers such as KNN, SVM, Naive Bayes, etc can be used [12].

As seen above, after trying out different types of training and prediction options, we went with the Classifier based model using KNN as a classifier.

III. Proposed Work

A. Tools Used

1) Python 3.7: Python is used to design and develop a face recognition program. Python is a high-performance language for technical computing. It incorporates calculation, representation, and programming in a simple-to-utilize climate where issues and arrangements are communicated in recognizable numerical documentation. It is a programming language and furthermore has a virtual programming environment. Similarly as a refined mini-computer, we can perform activities from the order line. Or on the other hand we can make projects and capacities that perform tedious work, similarly to some other coding language.
2) Pycharm: PyCharm is an integrated development environment (IDE) utilized in PC programming, explicitly for the Python language. It is created by the Czech organization JetBrains. Pycharm was used as an IDE to perform the functions.
3) Visual Studio code: Visual Studio Code is a freeware source-code editor made by Microsoft for Windows, Linux and macOS. We use this to install "Desktop development with C++" so as to support the library "cmake".
4) Google Images: Google Images is a search service additional to the normal Google Search owned by Google that allows users to search the World Wide Web for image content. This platform was introduced on July 12, 2001. We use this platform to create our dataset.
5) JavaScript console: The JavaScript console is a command line interface in the browser which is capable to execute snippets of code. We use this tool also to generate our dataset.

B. Approach

1) Input Image Acquisition:
   - Firstly, we go to google images platform and search for the images we want in our database. In our case, we built our dataset of friends' characters.
   - Then, we run the JavaScript snippet to simulate right click and download in the browser.
   - This way, we get a file containing the urls of the images presented in the search
   - Then we run a python script to loop through the urls and download the images.
   - Images for each person is stored in their own folder.
   - Finally, we repeat this process for every person we want to add in our dataset. This is how we built our dataset.
2) Face Detection:
   - We have used face_recognition.face_locations functions using CNN model for our implementation.
   - This function returns back a 2-D array of bounding boxes of human faces in an image [15].
   - "CNN" and "HOG" are the two methods that can be used for face detection, CNN is a slower but accurate process whereas HOG is a faster but less accurate process.
   - In the Histogram of gradient (HOG) method we convert the image into black and white image and then replace each pixel value with gradient vector showing the direction in which it gets darker.
   - While in CNN we use ResNet-34. The weights are learned from Labelled Faces in the Wild dataset and find a bounding box around the face by finding the left, top, right and bottom coordinates.
3) Feature extraction:
   - Now that we have identified our face, we need to learn its features while the face might be in different position, lightings, skin texture and resolutions.
   - Thus to overcome these problems we add landmarks in the images unique to them.
   - The basic idea is we will come up with 68 specific points[13] using machine learning.
4) Encoding faces:
   - Given a case of recognising several unknown faces from the known faces we need to use Deep learning to encode the image in a 128 (embedding points) vector.
   - This encoding process using convolutional neural network to fetch 128 embedding points for unknown image from the known image runs over millions of times to give the vector output and thus is a very

computationally tedious task, it might even take a day to just encode the dataset of small size.

5) Classification:
   - In this process we find the person's name from the known images.
   - A lot of classifiers have been trained to perform this task like SVM, PCA, etc.
   - For our approach we have used the k-NN approach where we use the Euclidean distance or cosine similarity between the encoded vectors so as to recognize the person from the image.

IV. Implementation Details

Our proposed implementation for face recognition is based on OpenCV, Python and deep learning. In our implementation we have used both HOG and CNN methods to compare between the results generated by them. Deep metric learning is the method used, this method differs from classification method as we use real valued feature vectors instead of trying to output a single label. While in HOG we find the gradient vector for each pixel so as to understand the direction of the darkness.

A. Libraries Used

1) OpenCV: This library is used for real-time computer vision. In our implementation we use this library to show the final output.
2) cmake: CMake is an open-source, cross-platform family of tools designed to build, test and package software[13]. 'face_recognition' library needs cmake base.
3) dlib: Dlib is a modern C++ toolkit containing machine learning algorithms and tools for creating complex software in C++ to solve real world problems.[14] Thus installed Visual Code studio for the same. It provides the GPU support for 'face_recognition' library.
4) face_recognition: Built using dlib's state-of-the-art face recognition built with deep learning to recognize and manipulate faces from Python [15].
5) os: To extract the path of the dataset.
6) pickle: To save the features extracted into a pickle file for further use.
7) imutils: A series of convenience functions to make basic image processing functions[16].

B. Implementation Steps

1) encode_faces.py: We load our dataset in the program and iterate through the imagePaths. While iterating through the imagePath we perform the following tasks:
   - Read the image in cv2 and convert the image from BGR to RGB for the face_reconition library to understand.
   - Then we detect faces in the images using face_recognition.face_locations, using CNN and HOG model one by one. The function returns a 2-D array representing the bounding box around the faces.
   - Using the RGB images and the 2-D array representing the boxes we encode the face using face_recognition.face_encodings function which returns a 128-d vector.
   - We store all these values in a list called knownEncodings and knownNames.

   Using these lists we generate a dictionary type of variable to store all the collected information a pickle file called encodings_file.pkl.
2) recognize_faces_image.py: In this program we read the pickle file into data. An unknown image input is taken and encoding vector is found for it too. We loop over these encoded values and perform the following tasks:
   - find matches of the embedding points from the data using face_recognition.compare_faces. This function returns a list of true/false values as per the matchings of the encodings.
   - Initial name is assigned as Unknown.
   - If we find any True in the matchings then we find the matchedIndex.
   - If the image from the data is in the matchedIndex then we fetch the name and count the number of times it has been matched. Thus using k-NN and votes we find the matchedIndexes.

   From all the matches found, find the one with the most number of counts and assign that name of the face. The name is displayed using OpenCV.

V. Results and Discussion

Our implementation used both cnn and hog method to compare the results:

Fig. 5. Both of them give the perfect output

VI. Conclusion

At present, recognition of an individual face use human spectators in numerous security applications Face recognition frameworks are additionally utilized related to restricted human obstruction in certain applications. It is profoundly attractive that the face recognition frameworks ought to have the option to give high dependability and precision under

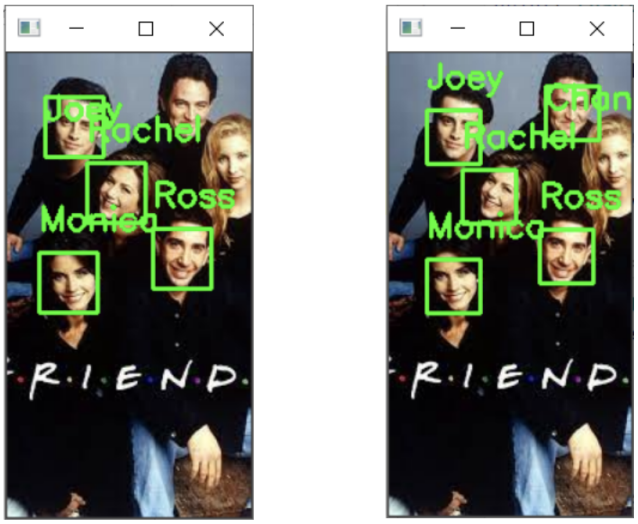

Fig. 6. HOG (right) outperforms CNN (left). HOG detects all the faces

Fig. 7. CNN (left) outperforms HOG (right). HOG doesn't display output while CNN does

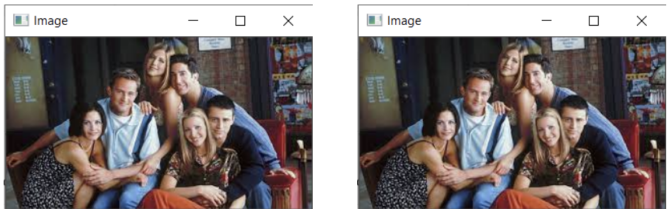

Fig. 8. Both display no result

changing conditions. All things considered, numerous calculations are not strong to high security applications like terrorists doing boundary crossing. Our approach used CNN and HOG methods to understand the working of both the methods under different situations. As per the results optioned it is quite clear that the ReNet-34 is gives better results than gradient vectors but again it is a very computationally and resource demanding method. Future scope of this topic would be to find a more efficient method to detect images with every possible lighting and resolutions to detect as many faces as possible by a better trained neural network which is not very computationally expensive.

REFERENCES

[1] H.A. Rowley, S. Baluja, and T. Kanade, "Neural Networks Based Face Detection", IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 1, pp. 22-38, 1998.

[2] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features",Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, CVPR , Vol.1, pp.I-511 - I-518, 2001.

[3] Vandana S. BhatandJagadeesh D. Pujari, "Face detection system using HSV color model and morphing operations ",International Journal of Current Engineering and Technology,Proceedings of National Conference on 'Women in Science Engineering, Special issue.1, pp.200- 204,2013.

[4] S.Chitra, G. Balakrishnan, "Comparative Study for Two Color Spaces HSCbCr and YCbCr in Skin Color Detection", Applied Mathematical Sciences, Vol.6, no.85, pp.4229 - 4238, 2012.

[5] Devendra Singh Raghuvanshi,Dheeraj Agrawal, "Human Face Detection by using Skin Color Segmentation, Face Features and Regions Properties", International Journal of Computer Applications, Vol. 38, No.9,pp.14-17,2012.

[6] SmitaTripathi ,Varsha Sharma ,Sanjeev Sharma, "Face Detection using Combined Skin Color Detector and Template Matching Method", International Journal of Computer Applications,Vol.26,No.7,pp.5-8,2011

[7] Y. Amit, D. Geman, and K. Wilder,"Joint induction of shape features and tree classifiers",IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 19 ,Issue 11,pp.1300-1305,1997

[8] M.H. Yang, D. J. Kriegman, N. Ahuja, "Detecting faces in images: A survey", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 24 (1), pp.34-58,2002.

[9] https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78.

[10] http://vis-www.cs.umass.edu/lfw/

[11] https://www.pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

[12] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016.

[13] https://cmake.org/

[14] http://dlib.net/

[15] https://face-recognition.readthedocs.io/en/latest/face_recognition.html

[16] https://pypi.org/project/imutils/