IOSR Journal of Engineering (IOSRJEN) **www.iosrjen.org**
ISSN (e): 2250-3021, ISSN (p): 2278-8719
PP 67-69

Role of Face Recognition in Machine Learning

Dr.J.Savitha
Associate Professor & Head, Department of Information Technology,
Sankara College of Science and Commerce, Coimbatore.

Abstract: *Face Recognition is an challenging area in research. Many Researchers found innovative ideas in this area. This paper deals with the role of face recognition in machine language.*
Index items: *Facebook, machine learning, face detection*

I. INTRODUCTION

Facebook has developed an uncanny ability to recognize the friends in photographs. In the old days, Facebook used to make you to tag your friends in photos by clicking on them and typing in their name. Now as soon as the photo is uploaded, Facebook tags everyone for *like magic*.

This technology is called face recognition. Facebook's algorithms are able to recognize friends' faces after they have been tagged only a few times. It's pretty amazing technology—Facebook can recognize faces with 98% accuracy which is pretty much as good as humans can do.

II. HOW TO USE MACHINE LEARNING ON A VERY COMPLICATED PROBLEM

So far machine learning is used to solve isolated problems that have only one step—estimating the price of a house,generating new data based on existing data and telling if an image contains a certain object. All of those problems can be solved by choosing one machine learning algorithm, feeding in data, and getting the result. But face recognition is really a series of several related problems:

1. First, look at a picture and find all the faces in it
2. Second, focus on each face and be able to understand that even if a face is turned in a weird direction or in bad lighting, it is still the same person.
3. Third, be able to pick out unique features of the face that you can use to tell it apart from other people— like how big the eyes are, how long the face is, etc.
4. Finally, compare the unique features of that face to all the people you already know to determine the person's name.

As a human, brain is wired to do all of this automatically and instantly. In fact, humans are *too good* at recognizing faces and end up seeing faces in everyday objects. Computers are not capable of this kind of high-level generalization (*at least not yet…*), and have to teach them how to do each step in this process separately. *pipeline* is to solve each step of face recognition separately and pass the result of the current step to the next step.

III. FACE RECOGNITION—STEP BYSTEP

Step 1: Finding all the Faces

The first step in pipeline is *face detection*, where we locate the faces in a photograph. Face detection is a great feature for cameras. When the camera can automatically pick out faces, it can make sure that all the faces are in focus before it takes the picture. But use it for a different purpose—finding the areas of the image to pass on to the next step in pipeline.

Face detection went main stream in the early 2000's when Paul Viola and Michael Jones invented a way to detect faces that was fast enough to run on cheap cameras. However, much more reliable solutions exist now. The method invented in 2005 called Histogram of Oriented Gradients—or just ***HOG*** for short is used.To find faces in an image, start by making the image black and white .

Fig:1 Posing and Projecting Faces

The goal is to figure out how dark the current pixel is compared to the pixels directly surrounding it. Draw an arrow showing in which direction the image is getting darker.Looking at just this one pixel and the pixels touching it, the image is getting darker towards the upper right.Repeat the process for **every single pixel** in the image, and end up with every pixel being replaced by an arrow. These arrows are called *gradients* and they show the flow from light to dark across the entire image.

Break up the image into small squares of 16x16 pixels each. In each square, count up the gradients point in each major direction (how many point up, point up-right, point right, etc…). Then replace that square in the image with the arrow directions that were the strongest.The end result is turned original image into a very simple representation that captures the basic structure of a face in a simple way.The original image is turned into a HOG representation that captures the major features of the image regardless of image brightnesss.

Humans can easily recognize that both images are of Will Ferrell, but computers would see these pictures as two completely different people.To account for this, try to warp each picture so that the eyes and lips are always in the sample place in the image. This will make it a lot easier for us to compare faces in the next steps.To do this, use an algorithm called **face landmark estimation**.

The basic idea is we will come up with 68 specific points (called *landmarks*) that exist on every face—the top of the chin, the outside edge of each eye, the inner edge of each eyebrow, etc. Then we will train a machine learning algorithm to be able to find these 68 specific points on any face.The 68 landmarks we will locate on every face. This image was created by Brandon Amos of CMU who works on OpenFace.

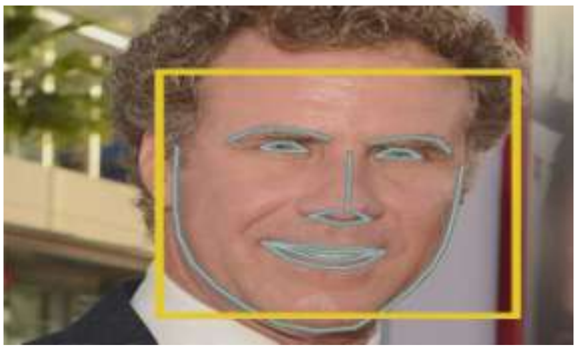

fig 2: face landmarks

Now no matter how the face is turned, able to center the eyes and mouth are in roughly the same position in the image. This will make the next step a lot more accurate.

Step 3: Encoding Faces

The simplest approach to face recognition is to directly compare the unknown face found in Step 2 with all the pictures of people that have already been tagged. Find a previously tagged face that looks very similar to unknown face, it must be the same person.

There's actually a huge problem with that approach. A site like Facebook with billions of users and a trillion photos can't possibly loop through every previous-tagged face to compare it to every newly uploaded picture. That would take way too long. They need to be able to recognize faces in milliseconds, not hours.

Extract a few basic measurements from each face. Then measure the unknown face the same way and find the known face with the closest measurements. For example, measure the size of each ear, the spacing between the eyes, the length of the nose, etc.

The most reliable way to measure a face

It turns out that the measurements that seem obvious to humans (like eye color) don't really make sense to a computer looking at individual pixels in an image. Researchers have discovered that the most accurate approach is to let the computer figure out the measurements to collect itself. Deep learning does a better job than humans at figuring out which parts of a face are important to measure.

The solution is to train a Deep Convolutional Neural Network (just like we did in Part 3). But instead of training the network to recognize pictures objects ,train it to generate 128 measurements for each face.The training process works by looking at 3 face images at a time:

1. Load a training face image of a known person
2. Load another picture of the same known person
3. Load a picture of a totally different person

Then the algorithm looks at the measurements it is currently generating for each of those three images. It then tweaks the neural network slightly so that it makes sure the measurements it generates for #1 and #2 are slightly closer while making sure the measurements for #2 and #3 are slightly further apart.

After repeating this step millions of times for millions of images of thousands of different people, the neural network learns to reliably generate 128 measurements for each person. Any ten different pictures of the same person should give roughly the same measurements.

Machine learning people call the 128 measurements of each face an **embedding**. The idea of reducing complicated raw data like a picture into a list of computer-generated numbers comes up a lot in machine learning (especially in language translation). The exact approach for faces we are using was invented in 2015 by researchers at Google but many similar approaches exist.

Encoding our face image

This process of training a convolutional neural network to output face embeddings requires a lot of data and computer power. Even with an expensive NVidia Telsa video card, it takes about 24 hours of continuous training to get good accuracy.

But once the network has been trained, it can generate measurements for any face, even ones it has never seen before! So this step only needs to be done once. Lucky , the fine folks at OpenFace already did this and they published several trained networks which can be directly used. Run the face images through their pre-trained network to get the 128 measurements for each face. The network generates nearly the same numbers when looking at two different pictures of the same person.

Step 4: Finding the person's name from the encoding

This last step is actually the easiest step in the whole process. Find the person in database of known people who has the closest measurements to test image.It can be done by using any basic machine learning classification algorithm. No fancy deep learning tricks are needed. Use a simple linear SVM classifier, but lots of classification algorithms could work.Train a classifier that can take in the measurements from a new test image and tells which known person is the closest match. Running this classifier takes milliseconds. The result of the classifier is the name of the person.

IV. EXECUTION

Let's review the steps we followed:

1. Encode a picture using the HOG algorithm to create a simplified version of the image. Using this simplified image, find the part of the image that most looks like a generic HOG encoding of a face.
2. Figure out the pose of the face by finding the main landmarks in the face. Once we find those landmarks, use them to warp the image so that the eyes and mouth are centered.
3. Pass the centered face image through a neural network that knows how to measure features of the face. Save those 128 measurements.
4. Looking at all the faces we've measured in the past, see which person has the closest measurements to our face's measurements. That's our match!

V. CONCLUSION

This paper even put together a pre-configured virtual machine with face_recognition, OpenCV, TensorFlow and lots of other deep learning tools pre-installed and this factor enforces to the future enhancement.

REFERENCES

[1]. Mattew Turk and Alex Pentland," Eigenfaces for Recognition," SPIE Vol.1192 IRCVVIn (i989), 22- 32.

[2]. Kirby and Sirovich, 1990. Application of Karhunen- Loeve procedure for the characterization of human faces. IEEE Trans. pattern analysis and machine intelligence, 12:103-108.

[3]. Turk, M.A. and A.L. Pentland, 1991. Face recognition using Eigen faces. Proc. IEEE computer society Conf Computer Vision and pattern recognition, pp: 586-591.

[4]. Kyungim Baek, Bruce A. Draper, J. Ross Beveridge, Kai She, "PCA vs. ICA: A Comparison on the FERET Data Set", Proceedings of the 6th Joint Conference on Information Science (JCIS), 2002, pp. 824-827.

[5]. T. Chen, W.Yin, X.-S. Zhou, D. Comaniciu, T. S. Huang, Total Variation Models for Variable Lighting Face Recognition and Uneven Background Correction", IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 28(9), 2006, pp.1519- 1524.