Face Recognition and Classification Using GoogleNET Architecture

R. Anand, T. Shanthi, M. S. Nithish and S. Lakshman

Abstract Face recognition is the most important tool in computer vision and an inevitable technology finding applications in robotics, security, and mobile devices. Though it is a technology of the past, state-of-the-art machine learning (ML) techniques have made this technology game-changing and even surpass human counterparts in terms of accuracy. This paper focuses on applying one of the advanced machine learning tools in face recognition to achieve higher accuracy. We created our own dataset and trained it on the GoogleNet (inception) deep learning model using the Caffe and Nvidia DIGITS framework. We achieved an overall accuracy of 91.43% which was fairly high enough to recognize the faces better than the conventional ML techniques. The scope of the application of deep learning is enormous and by training a huge volume of data with massive computational power, accuracy greater than 99% can be achieved. This paper will give a glimpse of deep learning, from creation of dataset to training and deploying the models, and the method can be applied for dataset corresponding to any field, be it medicine, agriculture or manufacturing, reducing the human effort and thus triggering the revolution of automation.

1 Introduction

Face recognition has been, for the past few years, the frontrunner in many technological fields like robotics, biometrics and security. Though face recognition dates back to past few decades, it has gained more interest recent years due to the state-of-the-

R. Anand · T. Shanthi · M. S. Nithish · S. Lakshman (✉)
Department of Electronics and Communication Engineering, Sona Signal and Image Processing Research Center, Sona College of Technology, Salem, India
e-mail: lakshman2497@gmail.com

R. Anand
e-mail: anand.r@sonatech.ac.in

T. Shanthi
e-mail: shanthi@sonatech.ac.in

K. N. Das et al. (eds.), *Soft Computing for Problem Solving*,
Advances in Intelligent Systems and Computing 1048,
https://doi.org/10.1007/978-981-15-0035-0_20

art accuracy achieved. The accuracy is fairly high enough to compete with human counterparts and even surpass them in some cases. Such high accuracies became possible only after using Convolutional Neural Nets (CNN) in face recognition [1]. The CNN then evolved to deep learning which is simply a more sophisticated neural net. Therefore, the seriousness with which face recognition algorithms are being developed has gone over the roof and more sophisticated models are developed and deployed by the researchers both in big corporate tech giants and in academia. Each year, one model created by some organization/team replaces another of the previous year in terms of accuracy and complexity. Models have achieved accuracies of a whopping 99.63% (FaceNet [2] by Google) which is more accurate than a human. In order to trigger an AI revolution and disrupt a wide range of tech sectors the tools for developing and deploying deep learning models have been simplified by tech titans like google, amazon, and facebook. With the huge computational power available they develop many deep learning models and give us the flexibility to customize it to our applications. In this paper, we take one of the advanced and a more versatile model called the GoogleNet or the Inception model. We created our own dataset by scrapping images from the internet and organizing it to a dataset. We will focus on changing the input layer of the inception model, customizing it to our dataset. The model has been trained using nvidia DIGITS and has achieved the desired accuracies.

2 Literature Review

Eigenfaces [3, 4] and Fisherfaces [5] were some of the first methods to propose an idea for recognizing faces. These two methods worked with gray scale images which are nothing more than a series of numbers, each number corresponding to some intensity level. So these methods treated images as vector. By treating the images as samples of data, we can perform a principal components analysis [6] and obtain the eigenvectors which make up the basis of the vector space. Each face has unique features and these features are what eigenvectors represent. The eigenvectors actually represent the strongest characteristics of the faces in the dataset. When a dataset is provided for training, the eigenvectors are extracted from the set as a whole. Whenever it is provided with a new image, it detects a face, extracts the eigenvectors and matches it with existing eigenvectors and thus determines to which person the face belongs. The change in illumination in faces within the same dataset was one of the major problems that these methods faced. When one face's illumination differed from the others', it made it hard to extract the eigenvectors. This problem was overcome by the Local Binary Pattern Histogram (LBPH) method. The LBPH [7, 8] method takes a different approach than the eigenfaces method. In LBPH each image is analyzed independently. The LBPH method is somewhat simpler, in the sense that we characterize each image in the dataset locally; and when a new unknown image is provided, we perform the same analysis on it and compare the result to each of the images in the dataset. In LBPH, an image is taken and each pixel in it is taken and assigned as the central value (or threshold value) and the pixels surrounding it are given a binary value based on the threshold value. When all pixels are converted to binary values, a histogram [9] is formed. But the problem with LBPH is that it does not read each and every feature of the face, as CNN does, which results in relatively lower accuracy.

Neural network is a technique that mimics the neurons in the human brain, which is the most advanced and powerful system ever known to man. They started to gain popularity when the closeness with which they resembled the neurons came to light and the accuracy of their output surpassed that of its predecessors. Le et.al proposed a model in which convolution operations are performed prior to the neural nets, which was named LeNet. Convolution operations are performed to extract certain features from the data provided, therefore assigning weights to the neural nets. This operation reduced the time as well as the complexity of the following layers. This model by Le was in the end of '90 s when the computing power of systems was not so high and the dataset for training was not properly confined, due to which the model was not widely used. But in the last decade, with computational powers of computers being so high, the model was again brought to light and modified according to the needs. One such model which surfaced during the last decade was AlexNet [10] which contained seven layers and provided a decent accuracy. After this, there were many attempts to develop the perfect model. One such model worth mentioning is ZF Net. Then the following year, there was ZF NET [11] (or Zeiler Fergus Net) which was a slight modification to AlexNet. Instead of using 11×11 sized filters in the first layer (which is what AlexNet implemented), ZF Net used filters of size 7×7 and a decreased the stride value. The reason behind this modification is that a smaller filter size in the first convolution layer helps to retain a lot of original pixel information in the input volume. A filtering of size 11×11 proved to be skipping a lot of relevant information, especially as this is the first convLayer. But not until 2015, when GoogleNet [12] was created and was high accuracy achieved. GoogleNet used 17 layers in its model of which 10 were inception layers. An inception layer is one in which operations like convolution, concatenation, and maxpooling are performed in parallel. This approach of using inception layers helped GoogleNet achieve the highest-achieved accuracy of 99.63%.

3 Dataset Creation

Dataset creation is very important in deep learning as the output of the training model, viz., the accuracy of the prediction/classification depends hugely on the amount of data fed for training. So, the more the data provided for training, the more accurate the prediction. In case of face recognition, in order to train a deep learning model the following has to be done.

- Provide as many unique faces as possible
- Provide the maximum no. of faces for each individual.

We created our own dataset taking the faces of celebrities concerning sportspersons and actors. The work involved was the collection of seven individuals' faces with each photo being almost close to the dimension 256 × 256. The dimension should not be too small because the bottleneck structure of GoogleNet would reduce the dimension further and the model will not provide the desired accuracy. After the collection was done, we put together all of the photos and made sure that no two faces of the same individual were identical and nor were below the required dimensions.

4 Proposed Work

The inception model provides state-of-the-art accuracy but the model is flexible, as we can change the hyperparameters in any layer of the model customizing it to our needs. In this paper, we attempt to modify the architecture more suitable to the dataset created. The size of the input images is 256 × 256. In the first layer of the architecture, convolution operation is performed with a filter size of 34 × 34 and a stride of value 2. This reduces the dimension of the image to 112 × 112. The size of the output due to convolution operation can be obtained as follows.

$$N_{out} = \frac{N_{in} + 2p - k}{s} + 1 \tag{1}$$

wherem N_{in} = input size, N_{out} = output size, p = convolution padding size, k = convolution kernel size and s = convolution stride size. Further maxpooling with a patch size of 3 × 3 and stride of value 1.5 reduces the dimension to 56 × 56. This is succeeded by another convolution and pooling operation followed by inception layers. The structures of the inception layers were not changed and it was the same as in GoogLeNet. This is similar to the bottleneck approach wherein each layer reduces the dimensions. The last few layers of the inception consist of fully connected layers made of neural nets. The linear and softmax functions are applied to each perceptron to include nonlinearities. The last layer of the neural net consists of seven neurons, which outputs different probabilities to different persons depending on the input image. The neuron with highest probability represents the person which model has correctly classified. The architecture flow of our proposed method which is shown in Table 1.

4.1 Caffe and DIGITS

The dataset prepared above was trained on the GoogLeNet deep learning model or Inception v1. The inception model is a very robust state-of-the-art model with very high accuracy. Hence, a number of deep learning platforms give support for this model. One such platform is the Caffe [13] and we used it because of the simplicity

Table 1 Architecture flow of our proposed method

Type	Patch size/stride	Output size
Convolution	34 × 34/2	112 × 112 × 64
Max pool	3 × 3/2	56 × 56 × 64
Convolution	3 × 3/1	56 × 56 × 192
Max pool	3 × 3/2	28 × 28 × 192
Inception (3a)		28 × 28 × 256
Inception (3b)		28 × 28 × 480
Max pool	3 × 3/2	14 × 14 × 480
Inception (4a)		14 × 14 × 512
Inception (4b)		14 × 14 × 512
Inception (4c)		14 × 14 × 512
Inception (4d)		14 × 14 × 528
Inception (4e)		14 × 14 × 832
Max pool	3 × 3/2	7 × 7 × 832
Inception (5a)		7 × 7 × 832
Inception (5b)		7 × 7 × 1024
Avg pool	7 × 7/1	1 × 1 × 1024
Drop out (40%)		1 × 1 × 1024
Linear		1 × 1 × 7
Softmax		1 × 1 × 7

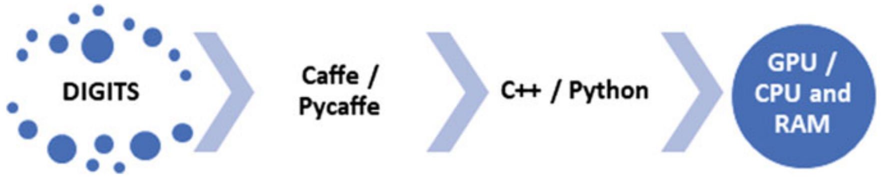

Fig. 1 Hierarchy of deep learning platforms and hardware

and ease of use. No surprise that it was developed for the above-stated purpose in UC Berkely. Caffe is a deep learning framework originally developed for developing deep learning applications using C++. With the support of Pycaffe the caffe framework supports python. The Caffe framework was cloned from github and installed on the system. DIGITS is another deep learning platform that is unparalleled in the ease of deployment and rapid prototyping of any deep learning model. The DIGITS platform was developed by GPU manufacturer Nvidia. The platform works best on the systems equipped with Nvidia GPU's but also supports CPU based systems. Unlike Caffe, the DIGITS is not a standalone platform. The DIGITS platform depends on the Caffe which in turn operates over python. The flow of Hierarchy of deep learning platforms and hardware is shown in Fig. 1.

Like Caffe, the DIGITS was also installed and the correct path to the Caffe for the operation of DIGITS was specified. When run, the DIGITS run as a local server in any browser of that system relying on the underlying CPU/GPU and memory. After running DIGITS the browser displays a user interface where the dataset can be loaded and the required deep learning model can be selected. All the parameters of the model during training can be visualized in the form of graphs where all metrics are available. After the required level of training at any point in time the resulting model can be downloaded.

4.2 Training Process

Training a deep learning model for a given dataset falls under supervised learning. For more sophisticated models like inception the number of hyperparameters are more and training such models require huge computational power and thousands of hours of training. To reduce this complexity, training method called transfer learning is adopted. In transfer learning, the inception model is pretrained by Google with accuracy close enough. Then the weight of the fully connected layers in the inception model is adjusted for the given dataset to achieve the required accuracy. Our hardware was a traditional laptop PC with Intel core i3 CPU and 4GB RAM running Ubuntu v16. The DIGITS was launched and the path to the dataset consisting of RGB images of different labels was specified. A total of 350 images were used for training, out of which 25% of the images were reserved for validation. The dataset is now prepared for training over a deep learning model. Then a classification model was created with the required dataset and the numbers of iterations were specified around 100. The classification model was selected as GoogLeNet with 256 $\times$ 256 architecture. The training was initiated and the model was trained for the given dataset. The PC was kept in a well-ventilated and cool environment with proper power backup and monitored time to time for accuracy and loss. The training was carried over 50 hours straight. All the parameters available as graph were visualized and the accuracy began to saturate after 100 iterations. After 113 iterations the training was stopped and the resultant model was downloaded. Now the downloaded model can be integrated with any other python program and the required operations can be performed.

For every iteration, the model understands the dataset better and the results can be inferred from the graph. The losses drastically reduce after a few iterations and then the slope of the loss decreases. The losses become low after a hundred iterations. Similarly, accuracy is low for the early iterations and the accuracy begins to saturate after hundred iterations. Table 2 shows the parameters that have evolved throughout the training process.

Table 2 Evaluation of parameters with different Epochs

Parameters	Epoch #25	Epoch #50	Epoch #75	Epoch #100	Epoch #113
Accuracy (val)	36.25	37.5	46.25	42.5	50
Accuracy top-5 (val)	86.25	90	86.25	88.75	92.5
Loss (train)	1.91165	1.4028	0.7599	0.88255	0.5612
Loss1/loss (train)	1.57042	1.1918	0.64438	1.0050	0.76085
Loss2/loss (train)	1.56094	1.216523	0.65539	1.03687	0.5499

5 Results and Discussion

Fig. 2 which is fed as input (Testing Image) to the trained model of our method. The model correctly predicts the person as Messi (person in the image is the football legend Lionel Messi). The Table 3 shows the prediction accuracy of test images.

The image passes through various layers of the inception model and the visualization of the image in each of the layers can be viewed below. The bottleneck approach of inception model is evident from the data obtained. In each of the layers, the dimension of the image is reduced. The convolution operation looks for features such as shades for eyes, forehead, and cheeks. The first layer uses kernel size of 34 × 34 with a stride of 2. Pooling and normalization further reduce the dimension of the image. The second convolution operation uses a kernel size of 3 × 3 with a stride of

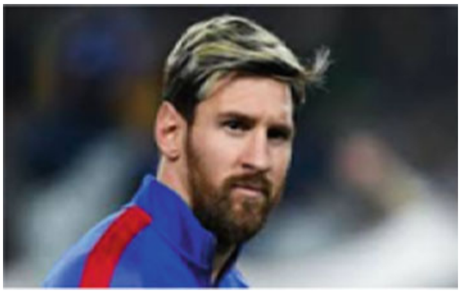
Fig. 2 Testing image

Table 3 Prediction rate

Testing image name	Prediction accuracy (%)
Messi	86.52
Federer	13.33
Vidyut	0.13
Tyson	0.01
Ajith	0.01

Table 4 Confusion matrix

Predicted class/Actual class	Ajith	Federer	Messi	Surya	Tyson	Vidyut	robertjr	Per-class accuracy (%)
Ajith	7	0	1	1	0	0	1	70
Federer	0	6	1	1	0	1	1	60
Messi	1	1	7	1	0	0	0	70
Surya	0	1	0	8	0	1	0	80
Tyson	0	1	0	0	9	0	0	90
Vidyut	1	3	0	0	0	6	0	60
Robertjr	0	0	0	0	0	1	9	90

2. After these operations a number of inception layers start training on the images and also reduce the dimensionality. In the final layer, classifier with a softmax activation classifies the individuals.

5.1 Confusion Matrix

Once the testing is over the confusion matrix can be created for the obtained model. In confusion matrix a separate set of images, not the ones used for training, belonging to the identities or persons for which the model has been trained is used to evaluate the model. All the images are grouped into folders, each folder consisting of images of respective persons. The same is created as a dataset in DIGITS. Now, the dataset is converted into a text file that consists of the image paths of all the images. Now the trained model is selected and the text file is given as input to classify all the images at once. Now the model classifies the images and creates a table as shown in Table 4.

The first column lists the individuals who have been used to evaluate the model. Per-class accuracy gives how accurate the model was correctly recognizing the person. The overall accuracy obtained by the model was 91.43% for Top-five predictions.

6 Conclusion

Deep learning is more than yet another machine learning technique. It has a huge scope and the potential to disrupt every sector we know ranging from medicine, defense, manufacturing, mobile services. Face recognition is one such application of deep learning and has already disrupted the field of computer vision. Since face recognition is the simplest application, i.e., to classify the images, it acts as a benchmark to test and experiment all the new machine learning models being developed.

By doing the above work, we explored the different frameworks and platforms in deep learning and knew how to customize any deep learning model depending on our applications. This paperwork gives a clear view of deep learning from the creation of dataset to training using a model and deploying it in real time. These steps can be applied to datasets obtained from any field and corresponding actions like classification, decision-making, and control can be made. Deep Learning is the future and it will disrupt all the fields with automation.

References

1. Lawrence, S., et al.: Face recognition: A convolutional neural-network approach. IEEE Trans. Neural Netw. **8**(1) 98–113 (1997)
2. Schroff, F., Kalenichenko, D., Philbin, J.: Facenet: a unified embedding for face recognition and clustering. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
3. Zhang, Jun, Yan, Yong, Lades, Martin: Face recognition: eigenface, elastic matching, and neural nets. Proc. IEEE **85**(9), 1423–1435 (1997)
4. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces versus fisherfaces: recognition using class specific linear projection. Yale University New Haven United States (1997)
5. Turk, M.A., Pentland, A.P.: Face recognition using eigenfaces. In: Proceedings CVPR'91, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE (1991)
6. O'Rourke, N., Psych, R., Hatcher, L.: A step-by-step approach to using SAS for factor analysis and structural equation modeling. Sas Institute (2013)
7. Turk, Matthew, Pentland, Alex: Eigenfaces for recognition. J. Cogn. Neurosci. **3**(1), 71–86 (1991)
8. Yang, Bo, Chen, Songcan: A comparative study on local binary pattern (LBP) based face recognition: LBP histogram versus LBP image. Neurocomputing **120**, 365–379 (2013)
9. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2005 CVPR 2005, vol. 1. IEEE (2005)
10. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems (2012)
11. Zeiler, M.D., Fergus, R., Visualizing and understanding convolutional networks. In: European Conference on Computer Vision. Springer, Cham (2014)
12. Szegedy, C., et al.: Going deeper with convolutions. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (2015)
13. Jia, Y., et al.: Caffe: convolutional architecture for fast feature embedding. In: Proceedings of the 22nd ACM International Conference on Multimedia. ACM (2014)