Deep Learning on a Raspberry Pi for Real Time Face Recognition

O. Dürr, Y. Pauchard, D. Browarnik, R. Axthelm, and M. Loeser

School of Engineering, Zurich University of Applied Sciences, Winterthur, Switzerland

Abstract

In this paper we describe a fast and accurate pipeline for real-time face recognition that is based on a convolutional neural network (CNN) and requires only moderate computational resources. After training the CNN on a desktop PC we employed a Raspberry Pi, model B, for the classification procedure. Here, we reached a performance of approximately 2 frames per second and more than 97% recognition accuracy. The proposed approach outperforms all of OpenCV's algorithms with respect to both accuracy and speed and shows the applicability of recent deep learning techniques to hardware with limited computational performance.

Categories and Subject Descriptors (according to ACM CCS): I.5.1 [Pattern Recognition]: Models—Neural Nets I.5.2 [Pattern Recognition]: Design Methodology—Classifier Design and Evaluation I.5.4 [Pattern Recognition]: Applications—Computer Vision

1. Introduction

Automatically recognizing faces and identifying individuals is a problem of large interest and there exists numerous approaches to solve this problem. Recently, *deep neural networks* and in particular *convolutional neural networks* (CNNs) have shown impressive classification performance in face recognition tasks [TYRW14]. While these approaches are promising, their classification pipelines usually require considerable computational resources. By exploiting the asymmetry of CNNs, where only the training phase is computationally expensive whereas classification itself is cheap, we show that fast and accurate face recognition can also be achieved on small computers such as the Raspberry Pi. For benchmarking we compare the CNN approach to standard OpenCV (version 2.4.9) classifiers.

2. Methods

2.1. Data Sets

In order to evaluate the entire pipeline on a single hardware, we utilized the Raspberry Pi camera module *Pi NoIR* to generate both test and training sets for 6 individuals with a total of approximately 40 pictures per capita. For each person we created two distinct sets of images. The first set—a collection of indoor images—served as training set whereas the second set, taken in an outdoor environment, was exclusively used for testing purposes. Figure 1 shows typical examples of training and test images.

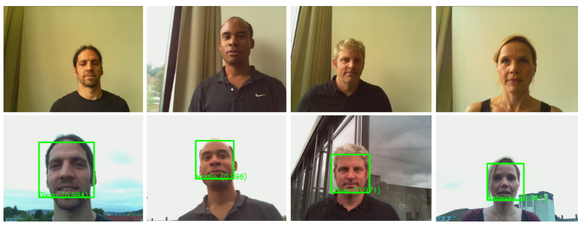

Figure 1: *The top row shows examples of the training set. The bottom row depicts images from the test set, where the faces have already been detected and classified.*

2.2. Face Detection and Preprocessing

As a first step in the face detection pipeline the original high-definition color images, taken with the Raspberry Pi camera, were converted to 8-bit gray-scale and downscaled to a resolution of 640×480 pixels. We then applied OpenCV's standard Viola-Jones algorithm to detect the faces on the scaled images.

DOI: 10.2312/egp.20151036

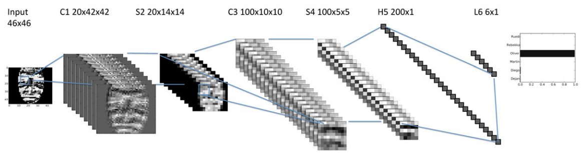

Figure 2: *Schematic representation of the CNN featuring maxpooling, convolutional and hidden layers. The 46×46 source image is shown on the left, and the prediction result, computed in the regression layer, is shown on the right.*

Method	Accuracy	Classification Time [msec]	Enrollment Rate N_e/N	Total Time per Face [msec]
CNN (p_0=0.85)	99.59%	105 ± 8	250 / 278	529 ± 64
CNN (p_0 = 0.0)	97.48%	105 ± 8	278 / 278	529 ± 64
Fisherfaces (no al.)	88.50%	54 ± 11	278 / 278	511 ± 89
Fisherfaces (al.)	96.87%	535 ± 89	192 / 278	1006 ± 118

Table 1: *Accuracy and performance for various approaches. The time for classification also includes the time for preprocessing.*

2.3. Convolutional Neural Network Classifier

As a novel approach, we used a CNN for classification. The entire neural network approach was implemented in Python using the deep-learning framework theano [BLP*12].

Prior to applying the CNN classifier, each detected face was scaled to 48×48 pixels and an ellipsoidal region around the center of the detected face was masked. As opposed to standard CNN approaches (see e.g. [TYRW14]) we applied the *local binary pattern* (LBP) operator [OPH96] prior to feeding the image to the CNN. By encoding each pixel depending its 8 local neighbors the LBP operator decreased the illumination dependence and reduced the image size to 46 × 46 pixels .

The architecture of our CNN is depicted in Figure 2. The newly obtained 46 × 46-images served as input. In the first convolutional layer C_1, 20 kernels of size 5×5 were applied, resulting in 20 42×42 images, from which the maximum of 3×3 neighboring pixels was taken (maxpooling, S_2). As a next step, the results were fed into the second convolutional layer C_3, using 100 5 × 5 filters. Next, a 2 × 2 - maxpooling (S_4) was done, resulting in 100 5×5 images. These 2500 pixels were then taken as an input for a fully connected hidden layer H_5 with an output of 200 neurons, which was then fed into a multinomial logistic regression with 6 outputs representing the 6 persons. The final output of the multinomial logistic—shown in Figure 2—is the likelihood that the input image belongs to a given person. An animation of the CNN classifier can be found under `http://youtu.be/oI1eJa-UWNU`.

2.4. Training of the Neural Network

As the training of the neural network is computationally expensive, it was carried out on a NVIDIA GeForce GTX 780 GPU (40 minutes). The trained model was then transferred to a Raspberry Pi. During the learning phase we maximized the log-likelihood of the training data with a standard batch gradient descent method using a batch size of 10 and a learning rate of $\alpha = 0.1$ which we continuously decreased by a factor of 0.993 during 1000 epochs.

In order to augment the training set we generated new images by randomly modifying the original images in one of the following ways: rotation (from 3 to 6 degrees), translations (from 1 to 2 pixels), rescaling (by a factor between 0.9 to 1.1) and randomly blackening 20% of the pixels.

3. Results and Conclusion

As a benchmark we chose various OpenCV classifiers and the highest performance was achieved using the Fisherface classifier. Aligning the images prior to classification increased the classification accuracy from 87.5% to 96.9%. Yet, as the alignment procedure was based on detecting both eyes, the number of enrolled images decreased from $N = 278$ to $N_e = 192$. Without any pre-processing the CNN featured an initial recognition rate of 24%. Applying the LBP operator reduced the dependence on illumination, boosting the recognition rate to 82%. Artificially augmenting the training data further increased the hit rate to 97.5% on all 278 test images. This accuracy could be increased even further by rejecting those images for which the best estimate, returned by the regression layer, was below a given threshold p_0. A timing analysis on the Raspberry Pi revealed that the Viola-Jones algorithm required 423 ± 64ms to detect a face in a given image. The CNN required another 105 ± 8ms for face recognition whereas the Fisherface approach needed 535 ± 89ms for the same task (see Table 1).

In conclusion, our approach leads to significant improvements with respect to both speed and accuracy compared to OpenCV. Thus, recent deep-learning techniques are also applicable on hardware with limited resources.

References

[BLP*12] BASTIEN F., LAMBLIN P., PASCANU R., BERGSTRA J., GOODFELLOW I. J., BERGERON A., BOUCHARD N., BENGIO Y.: Theano: new features and speed improvements. Deep Learning and Unsupervised Feature Learning NIPS 2012 Workshop, 2012. 2

[OPH96] OJALA T., PIETIKÄINEN M., HARWOOD D.: A comparative study of texture measures with classification based on featured distributions. *Patt. Rec. 29*, 1 (1996), 51–59. 2

[TYRW14] TAIGMAN Y., YANG M., RANZATO M., WOLF L.: DeepFace: Closing the gap to human-level performance in face verification. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on* (2014), pp. 1701–1708. 1, 2