Embedded facial image processing with Convolutional Neural Networks

Franck Mamalet, Sébastien Roux, Christophe Garcia
Orange Labs, France
Email: firstname.name@orange-ftgroup.com

Abstract— **This paper presents an embedded facial image analysis framework based on Convolutional Neural Networks (*ConvNets*). This robust framework has been proposed by Garcia, Delakis and Duffner on general purpose workstations without any constraints on computational and memory resources. We show that *ConvNets*, which consist of a pipeline of convolution and subsampling operations followed by a Multi Layer Perceptron, are particularly well suited for implementation on embedded processors. We present a set of high-level optimizations, such as automatic fractional transformation, convolution and subsampling fusion and memory requirement optimizations that can be applied to these algorithms without any loss in performance, leading to a speedup factor up to 700 compared to the reference implementation. This work leads to a face processing library able to handle the complete framework and its applications on mobile phones.**

I. Introduction

Facial image processing is an area of research dedicated to the extraction and analysis of information about human faces, information which is shown to play a central role in social interactions including recognition, emotion and intention. Over the last decade, it has become a very active research field due to the large number of possible applications, such as model-based video coding, image retrieval, surveillance and biometrics, and intelligent human-computer interaction.

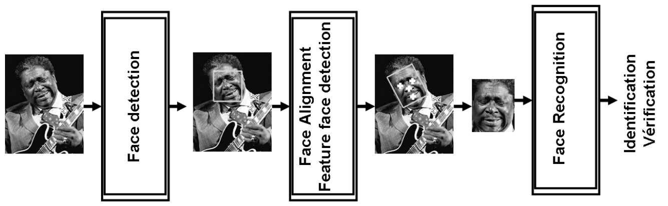

Fig. 1. Generic framework for facial analysis

Fig. 1 presents a general framework for face analysis. It first relies on face detection in images, followed by facial feature detection (eyes, mouth, nose) in order to align the face bounding box and make the last step of face recognition more robust. Many algorithms have been proposed in the literature to handle these key steps and provide robust systems in order to cope with the variability of facial image appearances due to lighting conditions, poses and expressions, image noise and partial occlusions. These techniques are commonly classified in two main groups [1], [2]:

- Local methods: simple visual features are used like color, edges, geometric features or local template matching-based approaches [3], [4].
- Global methods: faces and features are treated as a whole and models are learnt from a training database [5]–[8].

The latter methods are well known to be more robust to the images variabilities, and show better performances on standard face databases.

Facial analysis also has many applications in embedded systems. Among them we can cite automatic focus in digital cameras, enhanced mobile videoconference, biometry, and intelligent user interfaces. However, advanced algorithms are usually developed on PCs without any implementation restrictions in mind. Low memory capacities, low CPU frequency and lack of specialized hardware such as a floating point unit are some of the major differences between a PC and an embedded platform. Thus, embedding such applications on power constraint systems is a challenging task and requires strong algorithmic, memory and software optimizations.

In the literature, we can find many implementations of facial analysis algorithms on various kinds of embedded platforms (processors, DSP, FPGA or Asic). If most of them are based on simple local methods, several implementations of global methods have been published: Tang *et al.* [9], [10] have proposed an embedded version of the Viola and Jones [7] Adaboost face detector, and also an eye detector, on mobile phones. The Adaboost technique was also used in [11] for implementing a face detector on a DSP. Other papers focus on FPGA or Asic targets, such as the Viola and Jones algorithm implementation in [12], the Rowley *et al.* [5] neuron based face detector implementation in [13].

Such real time embedded implementations often require a trade off between detection robustness, fast runtime and small code size, and a major side effect of embedding facial analysis methods is often the reduction of the algorithm efficiency.

In this paper we will focus on a neural-based framework that has been proposed by Delakis, Duffner and Garcia [14], [15], which robustness has been proven with respect to noise and partial occlusions as well as variations in lighting and pose. This framework, which relies on Convolutional Neural Networks (*ConvNets*) introduced by LeCun *et al.* [16], will be detailed in section II. We will then present in section III a set of optimizations for embedded software implementations that we have introduced and applied on several *ConvNet* based algorithms, using a methodology based on iterations of high-level code optimizations and profiling presented in [17],

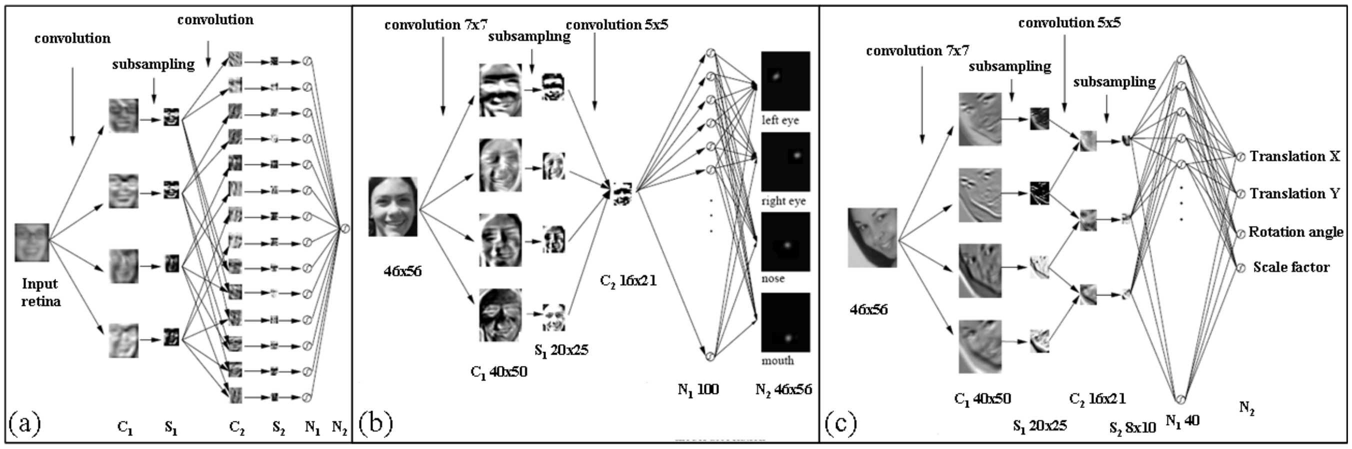

Fig. 2. The facial analysis Convolutional Neural Network framework: (a) CFF face detector, (b) Facial feature detector, (c) Face aligner

[18]. Before concluding, we will describe our *ConvNet* based embedded face analysis library and its performances.

II. Facial image processing with *ConvNets*

This section will give a short description of the entire *ConvNet* based framework that has been described in [15]. *Fig.* 2-a presents the general architecture of a *ConvNets*, which consists in a pipeline of convolution, subsampling and neuronal operations. This pipeline performs automatic feature extraction in an input retina (*ConvNet*'s input plane), and the classification of the extracted features, in a single integrated scheme. The three main types of layers in *ConvNets* are:

- Layers C_i are called convolutional layers, and contain a certain number of planes. Each element in a plane receives an input from a small neighborhood in the planes of the previous layer. Each plane can be considered as a feature map that has a fixed feature detector, which corresponds to a pure convolution with a mask applied over the planes in the previous layer. A bias is added to the results of each convolutional mask. Multiple planes are used in each layer so that multiple features can be detected.
- Layers S_i are called subsampling layers and consist in local averaging and subsampling operations. More precisely, each unit computes the average of four inputs, multiplies it by a trainable coefficient, adds a trainable bias, and passes the results through a hyperbolic tangent function, used as an activation function. This subsampling operation divides by two each dimension of the input map and increases the degrees of invariance to translation, scale, and deformation of the learnt patterns.
- Layers N_i are called classification layers, and are applied after feature extraction and input dimensionality reduction of C_i and S_i layers. These layers correspond to a multilayer perceptron.

In [14], Garcia and Delakis have presented a face detection method called Convolutional Face Finder (CFF) which is able to localize multiple faces with a minimum size of 20 pixels, rotated up to $\pm 20°$ in the image plane and turned up to $\pm 60°$. The CFF (see *Fig.* 2-a) consists in four 5×5 convolution maps in C_1 layer, fourteen 3×3 maps in C_2 layer, 14 neurons in N_1 layer, and a single neuron in N_2 layer classifying input retina as face or non-face. The CFF has still the best published performances on standard face databases with a very low false alarm rate (88% on CMU database with only 2 false alarms).

In [15], [19], Duffner *et al.* present a facial feature detector, which is able to detect four facial features in face images: the eyes, the tip of the nose and the center of the mouth. The architecture of this *ConvNet* is presented in *Fig.* 2-b: it has no S_2 layer, and 4 output maps consisting of arrays of neurons with the same dimensions as the retina. C_1 layer comprises four 7×7 convolution maps, C_2 a single 5×5 convolution map, N_1 a hundred neurons connected to the C_2 feature map. This network has been learnt to highlight a feature position using a 2-dimensional Gaussian function centered at the feature position. The network is applied at several translations and scales to find the maximum output score for each feature. This leads to a robust feature detector, able to cope with considerable variations in pose, illumination, facial expressions as well as noise and partial occlusions. It achieves a detection rate of 98.2% on the AR database with a maximum error of 15% of the bounding box width.

In [15], [20], Duffner and Garcia present a global face alignment method (see *Fig.* 2-c) which consists of a network very similar to the face detector, using four 7×7 convolution maps in C_1, three 5×5 maps in C_2, and four output neurons. These neurons are trained to estimate simultaneously translation in both axis, rotation and scale parameters of the transformation the input face image has undergone. A non-aligned detected face is then presented to the network which produces an estimation of the underlying transformation. To improve the correction an iterative scheme is applied where at each iteration only a certain portion (10%) of the correction is applied to the bounding box giving a new input retina. Experimental results show that 94% of the face images of the BioID database are aligned with an error of less than 10%

of the face bounding box width.

Garcia *et al.* have also proposed in [15], a face recognition scheme that is based on a non-linear projection and reconstruction with a *ConvNet*, and a nearest neighbor classification using the N_1 layer output values. The projection network comprises four 7×7 C_1 convolution maps, three 5×5 C_2 maps, no S_2 layer, 60 neurons in N_1, and an output layer consisting of an array of neurons with the same dimensions as the input retina. This network has been trained to reconstruct for each input face a reference image of this person. The recognition rate of this method on Yale database is 93.3%.

III. Optimizations for embedded *ConvNets*

A. Automatic fractional transformation

The first step towards embedding these algorithms is to transform the floating point computations into fractional ones using Q_{15} and Q_{31} arithmetic [21]. The main advantage of the *ConvNets* is that the results of each extraction layer pass through a hyperbolic tangent function. This limits the data dynamics, thus reduces the risk for common issues of fixed point computations such as arithmetic dynamic expansion and saturation. We have developed an automatic tool able to parse the floating point description of a given *ConvNet* and generate the corresponding fixed point network, coding all coefficients with half-words (16 bits).

The following equations describe the fractional transformation process for a C_i $N\times N$ convolution map. Eq. 1 is the classical formula for a floating point convolution (plus bias) computation with $K = [0, N-1]^2$ (w being the convolution kernel, in the input plane, and b the bias); A normalization parameter i_0 is chosen such as the sum of the fractional coefficients W (Eq. 2) is lower than 2^{15}. Thus the convolution accumulation will fit on 32 bits, and the fractional output $O_{(x,y)}$ will be lower than the maximum fractional input $IN_{(x,y)}$ (Eq. 3).

$$o_{(x,y)} = \sum_{(u,v)\in K} w_{(u,v)} in_{(x+u,y+v)} + b \tag{1}$$

$$i_0 \text{ such as } 2^{i_0-1} \leq \sum_{(u,v)\in K} \|w_{(u,v)}\| + \|b\| < 2^{i_0}$$

$$W_{(u,v)} = 2^{15-i_0} w_{(u,v)} \text{ , } B = 2^{31-i_0} b \text{ , } IN = 2^{16} in \tag{2}$$

$$O_{(x,y)} = (\sum_{(u,v)\in K} W_{(u,v)} IN_{(x+u,y+v)} + B) 2^{-16} \tag{3}$$

The hyperbolic tangent function is computed with a Look Up Table with saturation, avoiding any floating point computation. Our previous papers [17], [18] have shown that this fractional transformation does not reduce the performance (detection rate and precision) of the studied *ConvNet* networks.

B. Convolution and subsampling fusion

In [17], we have proposed an algorithmic optimization that can be applied in any convolution-subsampling pair in *ConvNets*: we can notice that, within subsampling layers, there is no overlapping between input data to produce two neighbor subsampled elements (Eq. 4). The output element value $s_{(i,j)}$ of a S_i layer can be expressed as follows:

$$s_{(x,y)} = \alpha(\sum_{(m,n)\in[0,1]^2} o_{(2x+m,2y+n)}) \tag{4}$$

$$= \alpha(\sum_{(m,n)} \sum_{(u,v)\in K} w_{(u,v)} in_{(2x+m+u,2y+n+v)} + b)$$

$$= \alpha(\sum_{(u,v)\in[0,N+1]^2} \tilde{w}_{(u,v)} in_{(2x+u,2y+v)} + \tilde{b}) \tag{5}$$

These equations 4 and 5 prove that we can fuse any $N \times N$ convolution C_i followed by a subsampling S_i operations, into a single $(N+1)\times(N+1)$ convolution CS_i with horizontal and vertical steps of two pixels. Table I gives the computational and memory access complexities for each version. The gain achieved by that algorithmic optimization is huge regarding the computational coast, *e.g.* 65% for a 5×5 convolution.

TABLE I
Complexity of convolution and subsampling fusion

	Number of Mac instructions
C_N + S	$4 * N^2 + 4$
C_{N+1}	$(N+1)^2$
Gain	$(3 * N^2 - 2 * N + 3)/(4 * N^2 + 4)$

C. Memory optimizations

When *ConvNets*' input retina is directly connected to the input image (no pre-processing), it is well-known that computations can be done on the whole image instead of at each retina position, factorizing several convolution and subsampling operations that are common to successive retinas. This has been done on the CFF [14], and can also be applied in the feature detector to merge processings that differ only by horizontal or vertical translations on the input image.

An other constraint in embedded systems is the amount of available memory and the memory cache size, which requires for effectiveness to promote local computing.

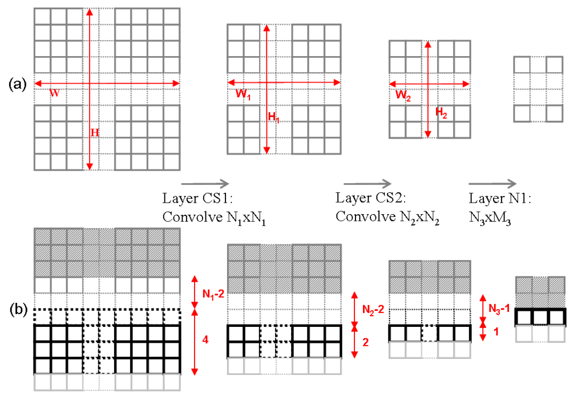

Fig. 3. *ConvNet* dataflow (a) global scheme, (b) line by line processing

Fig. 3-a represents the global processing of a $W \times H$ input image (simplified with only one map per layer). W_i, H_i are the output sizes of CS_i layers (for instance, $W_1 = (W - N_1 - 2)/2$). *Fig.* 3-b points out the possibility to compute N1 layer output line by line: Slashed (resp. unslashed) grey parts are unused (resp. re-used) previously computed data, whereas dark rectangles are newly computed data. In order to produce one output line of N_1 layer, we need N_3 lines of CS_2 output map among which only one new line has to be produced. Processing this line requires N_2 CS_1 map lines (two new ones). CS_1 has thus to be applied on $N_1 + 2$ input lines. This line by line processing has the same computational complexity, and drastically reduces the amount of memory required to compute the whole image, and avoid many cache misses.

IV. Embedded facial analysis library

The optimizations presented in the previous section have been applied to the facial analysis framework presented in [15]. The target platforms are ARM based mobile phones or digital assistants, and computation times are evaluated on a Xscale PXA27x based platform running at 624MHz.

The CFF algorithm has been implemented on this platform and optimized according to section III, and Table II presents a comparison between the original and optimized version for QCIF image processing (see [17] for details).

TABLE II
Reference and optimized CFF performances comparison

	Reference	Optimized
Detection Rate on CMU database	87.99%	88.2%
Processing rate	0.3 fr./s	6.5 fr./s
Memory footprint	3800 kByte	220 kByte

Another algorithmic optimization has been presented in [17] to enable face tracking in video, enabling to process up to 16 fr./s. To compare with Tang *et al.* [9] claims to proceed 3.6 fr./s on an ARM9E processor.

The facial feature detector, *Fig.* 2-b, has been implemented on the ARM platform using the same kind of optimizations. In [18], we have shown that the exhaustive computation of output neurons (to find the maximum location), can be replaced by a "three step" fast search algorithm, with no loss in feature detection precision. All the optimizations have induced a speed up factor of 700 in execution time, reaching a 12.8 faces/s processing rate. Tang *et al.* [10] are able to process 2.5 faces/s.

We have also optimized the face alignment method (*Fig.* 2-c), fractional transformation and convolution-subsampling fusion enable processing up to 7.4 faces/s.

The non-linear projection of the face recognizer method presented in [15] can also be optimized, fusing C_1 and S_1 layers and using fractionnal arithmetics. However, until now, we had not evaluated the full recognizer process and its performances on standard databases.

This embedded facial framework has been integrated into a single library that enables the detection of faces in images (or video with tracking), align bounding boxes and find facial features (face recognition integration is in progress). The whole framework is able to process up to three QCIF frames every second (with one face). Many applications have been implemented using this library on mobile phone, such as real-time centering of faces in video, automatic face cropping in pictures and intelligent picture browsing.

V. Conclusions

In this paper, we have described a set of optimizations that can be applied to Convolutional Neural Networks for their implementation on embedded processors, and its application to a *ConvNet* based facial analysis framework described in [15]. These optimizations enable us to reach a speed up factor up to 700, and to propose an embedded library which can robustly detect, track, align and find facial features on mobile phones at up to 3 QCIF frames per second. Further works will consist in implementing and testing the face recognizer, and also to explore others domain with embedded *ConvNet* networks such as mobile phone text detection and recognition.

References

[1] E. Hjelmås and B. K. Low, "Faces detection : A survey," *Computer Vision and Image Understanding*, vol. 83, pp. 236–274, 2001.
[2] M.-H. Yang, D. J. Kriegman, and N. Ahuja, "Detecting faces in image : A survey," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 24, 2002.
[3] E. Saber and A. M. Tekalp, "Face detection and facial feature extraction using color, shape and symmetry-based cost functions," in *Proc. ICPR'96*, 1996.
[4] R. Brunelli and T. Poggio, "Face recognition: Features versus templates," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 15, pp. 1042–1052, 1993.
[5] H. A. Rowley, S. Baluja, and T. Kanade, "Neural network-based face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 20, 1996.
[6] K. K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. PAMI*, vol. 20, 1998.
[7] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. Computer Vision and Pattern Recognition, CVPR'01.*, 2001.
[8] T. Cootes and C. Taylor, "Locating faces using statistical feature detectors," *Proc. FG'96*, 1996.
[9] X. Tang, Z. Ou, T. Su, and P. Zhao, "Cascade adaboost classifiers with stage features optimization for cellular phone embedded face detection system," in *Proc. ICNC'05*, 2005.
[10] D. Chen, X. Tang, Z. Ou, and N. Xi, "A hierarchical floatboost and mlp classifier for mobile phone embedded eye location system," in *Proc. ISNN'06*, 2006.
[11] J.-B. Kim, Y. H. Sung, and S.-C. Kee, "A fast and robust face detection based on module switching network," *Proc. FG'04*, 2004.
[12] K. Khattab, J. Dubois, and J. Miteran, "Cascade boosting-based object detection from high-level description to hardware implementation," *EURASIP Journal on Embedded Systems*, vol. 2009, 2009.
[13] T.Theocharides, G.Link, N.Vijaykrishnan, M.J.Irwin, and W.Wolf, "Embedded hardware face detection," *Proc. VLSID'04*, 2004.
[14] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, 2004.
[15] C. Garcia and S. Duffner, "Facial image processing with convolutional neural networks," in *Progress in Pattern Recognition*, 2007.
[16] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," in *Proc. of the IEEE*, 1998.
[17] F. Mamalet, S. Roux, and C. Garcia, "Real-time video convolutional face finder on embedded platforms," *EURASIP Journal on Embedded Systems*, 2007.
[18] S. Roux, F. Mamalet, C. Garcia, and S. Duffner, "An embedded robust facial feature detector," in *Proc. MLSP'07*, 2007.
[19] S. Duffner and C. Garcia, "A connexionist approach for robust and precise facial feature detection in complex scenes," in *Proc. ISPA'05.*, Sept. 2005.
[20] ——, "Robust face alignment using convolutional neural networks," in *Proc. VISAPP'08*, 2008.
[21] A. Bateman and I. Paterson-Stephens, *The DSP handbook : algorithms, applications and design techniques*. Prentice Hall, 2002.