Low-Complexity Approximate Convolutional Neural Networks

Renato J. Cintra, *Senior Member, IEEE*, Stefan Duffner, Christophe Garcia, and André Leite

***Abstract*— In this paper, we present an approach for minimizing the computational complexity of the trained convolutional neural networks (ConvNets). The idea is to approximate all elements of a given ConvNet and replace the original convolutional filters and parameters (pooling and bias coefficients; and activation function) with an efficient approximations capable of extreme reductions in computational complexity. Low-complexity convolution filters are obtained through a binary (zero and one) linear programming scheme based on the Frobenius norm over sets of dyadic rationals. The resulting matrices allow for multiplication-free computations requiring only addition and bit-shifting operations. Such low-complexity structures pave the way for low power, efficient hardware designs. We applied our approach on three use cases of different complexities: 1) a "light" but efficient ConvNet for face detection (with around 1000 parameters); 2) another one for hand-written digit classification (with more than 180 000 parameters); and 3) a significantly larger ConvNet: AlexNet with ≈1.2 million matrices. We evaluated the overall performance on the respective tasks for different levels of approximations. In all considered applications, very low-complexity approximations have been derived maintaining an almost equal classification performance.**

***Index Terms*— Approximation, convolutional neural networks (ConvNets), numerical computation, optimization.**

I. Introduction

SINCE their introduction in the 1990s by LeCun *et al.* [1], convolutional neural networks (ConvNets) have proven to be very powerful in many challenging computer vision tasks, such as hand-written character recognition [1], [2], embedded text detection and recognition [3], [4], automatic facial analysis [5]–[7], traffic sign recognition [8], pedestrian detection [9], vision-based navigation [10], and house numbers recognition [11], just to cite a few. Although state-of-the-art results have been reached in many different fields, ConvNets have become very popular only recently with the impressive results obtained by Krizhevsky *et al.* [12] in the recognition task, followed by Simonyan and Zisserman [13] who won the localization challenge at the Large-Scale Visual Recognition Challenge (ImageNet) 2014.

The main property of ConvNets is their capability for automatic extraction of complex and application-suitable features from raw data (e.g., pixels in computer vision). To do so, they integrate a pipeline of convolution and pooling layers generally followed by a multilayer perceptron, jointly performing local feature extraction and classification (or regression) in a single architecture, where all parameters are learned using the classical error back-propagation algorithm [14]. Traditionally, ConvNets are used for image processing tasks and, to this end, they are often applied on small regions of a bigger image in a sliding-window framework, for instance to detect an object. Because of weight sharing, each layer essentially performs convolution or pooling operations using small kernels inside a "retina," and when applied to a large image, the replication of the ConvNet operation over all positions in the image can be significantly optimized by performing each convolution over the full image at once, efficiently implementing a full image transformation pipeline.

However, during training and when applying the trained ConvNet, there is still a significant number of floating-point computations (multiplications and additions). In order to accelerate these operations, most current approaches and available software rely on parallel computing using GPUs [12], [15] facilitated to some extent by the inherently parallel architectures of ConvNets. The recent trend in "deep learning" to use more and more complex models with millions of parameters requires enormous amounts of computational resources, in particular for training the models but also for applying the learned ConvNets. Reducing the computational complexity of these models is thus of great interest to the research community as well as to industry. Moreover, such reduction in the complexity of ConvNets is necessary to implement them on devices with limited resources (such as mobile devices) in order to operate at acceptable speed, e.g., for real-time applications, and reduce the overall power consumption.

In this paper, we focus on drastically reducing the computational cost itself by proposing a post-training approximation scheme aiming at replacing all parameters of a ConvNet with low-complexity versions. That is, for the convolution filters, only additions and bit-shifting operations are performed—no multiplication is necessary. In addition, the activation function is sought to be replaced with the low-complexity alternatives. Formulated as an optimization problem, we adopt matrix approximation techniques based on the Frobenius norm error

Manuscript received March 8, 2016; revised January 7, 2017, November 4, 2017, and March 5, 2018; accepted March 5, 2018. This work was supported by CNPq, Brazil. *(Corresponding author: Renato J. Cintra.)*

R. J. Cintra was with the Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621 Lyon, France. He is now with the Signal Processing Group, Departamento de Estatística, Universidade Federal de Pernambuco, Recife 50670901, Brazil, and also with the Department of Electrical and Computer Engineering, University of Calgary, Calgary, AB T2N 1N4, Canada (e-mail: rjdsc@de.ufpe.br).

S. Duffner and C. Garcia are with the Université de Lyon, CNRS, INSA-Lyon, LIRIS, UMR5205, F-69621 Villeurbanne, France (e-mail: stefan.duffner@liris.cnrs.fr; christophe.garcia@liris.cnrs.fr).

A. Leite is with the Signal Processing Group, Departamento de Estatística, Universidade Federal de Pernambuco, Recife 50670901, Brazil (e-mail: leite@de.ufpe.br).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2018.2815435

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

and dyadic rational numbers represented in terms of canonical signed digit (CSD) encoding.

Indeed, a sound approximation theory is a necessary step to facilitate hardware development. This is illustrated in the case of image compression where the most efficient coding schemes are based on approximate matrices realized in dedicated hardware [16]–[19]. We aim at introducing an approach for approximating the CNNs in order to pave the way for future efficient dedicated hardware design.

This paper is organized as follows. Section II provides a literature review on the efficient numerical implementation of neural networks and, in particular, ConvNets. Section III details our approach for approximating the elements of a given ConvNet aiming at designing low-complexity structures. In Section IV, we present the results of our experimental evaluation of the proposed approach for two typical ConvNet architectures. We assess the approximate ConvNets relative to their exact counterparts in terms of several figures of merit. We conclude this paper in Section V.

II. Related Work

Although GPU implementations [20] allow for fast training and application of ConvNets on sufficiently equipped platforms, their integration on embedded systems for real-time applications may be more difficult due to the limited amount of available resources on these devices, which usually requires a good tradeoff between performance and code size. Several previous works have tackled this problem. In early works [21]–[23], weight parameters of neural networks have been represented as power-of-two integers. Thus, all multiplications can be operated as simple bit shifts. Direct training of these networks was also possible by keeping a floating-point version of each weight parameter, or otherwise use a technique called "weight dithering" [24]. Simard and Graf [25] extended this idea by encoding all other parameters as powers of two except the weights, i.e., neuron activations, gradients, and learning rates. Later, Draghici [26] conducted a broader analysis on the computational power of neural networks with reduced precision weights. Recently, Machado *et al.* [27] proposed a specific approximation scheme for sparse representation learning using only values of powers of two, and they integrated this quantization into the learning process. Finally, in the work of Courbariaux *et al.* [28], all weights are encoded as binary values (1-bit), and for the training a floating-point version is still needed. Similarly, Kim and Paris [29], proposes a completely binary neural network. However, an initial real-valued training phase is required.

More recently, special attention has been paid to hardware implementations of ConvNets, especially on field-programmable gate arrays (FPGAs). In [30], for example, a high-level optimization methodology is applied to the implementation of the convolutional face finder (CFF) face detector [6]. They propose algorithmic optimizations and advanced memory management and transform the floating-point computation into fixed-point arithmetic. Interestingly, such coarse approximation could furnish very similar detection rates and low false-alarm rates on referenced data sets, for a roughly sevenfold gain of speed. Later, this paper has been extended in [31], where the authors present for the first time several implementations of the CFF algorithm on FPGA, with a parallel architecture composed of a processing element ring and an FIFO memory, which constitutes a generic architecture capable of processing images of different sizes. Farabet *et al.* [32] also propose a scalable hardware architecture to implement the large-scale ConvNets, with a modular vision engine for large image processing, with FPGA and application-specific integrated circuit (ASIC) implementations. Chakradhar *et al.* [33] present a dynamically configurable FPGA coprocessor that adapts to complex ConvNet architectures exploiting different types of parallelism. A very low-complexity ASIC design of the ConvNets has been developed by Chen *et al.* [34], allowing for very high execution speeds and power consumption of state-of-the-art ConvNets. Finally, Zhang *et al.* [35] propose an FPGA design strategy and algorithmic enhancements to optimize the computational throughput and memory bandwidth for any given ConvNet architecture.

Other recent works have focused on the *algorithmic* and *memory* optimizations of large-scale ConvNets. For example, Mamalet and Garcia [36] proposed different strategies for simplifying the convolutional filters (fusion of convolutional and pooling layers, 1-D separable filters), in order to modify the hypothesis space and to speed up learning and processing times. These convolutions can also effectively be performed by simple multiplications of the filters with the respective input images in the frequency domain [37]. However, due to the overhead of the fast Fourier transform, there is only a computational gain with larger filter sizes, and if a given filter can be reused consecutively for many input images. Vanhoucke *et al.* [38] presented a set of different techniques to accelerate the computation of ConvNets on CPU, mostly for Intel and AMD CPUs, exploiting, for example, single instruction, multiple data instructions, memory locality, and fixed-point representations. Also, many recent works [39]–[46] have focused on reducing the complexity of convolution or fully connected layers of large-scale ConvNets by replacing the high-dimensional matrix or tensor multiplications with several low-rank matrix multiplications using different low-rank factorization methods, either at the test time or both for training and testing. Although large gains in computational and memory resources can be obtained on complex ConvNets, these optimizations do not focus on hardware implementation and low-power constraints.

As opposed to many previous works that integrate the approximation process into the learning [28], [47]–[50], our approach operates on the existing fully trained models, which originally may have been aimed for standard PCs or more powerful architectures. Thus, our approximation scheme allows to integrate these models into hardware with much fewer resources.

III. Approximation Approach

A. General Goal

Our goal is to derive low-complexity structures capable of reducing the computational costs of a given ConvNet. Ideally, the following two conditions are simultaneously expected to be satisfied.

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

1) The computational elements of the ConvNet (convolutional filters, subsampling coefficients, bias values, and sigmoid function calls) are replaced by the corresponding low-complexity structures.
2) The performance of the ConvNet is not significantly degraded.

However, addressing both the above-mentioned conditions proves to be a hard task. In particular, the large number of variables, the nonlinearities, and extremely long simulation times prevent such approach. Also, to the best of our knowledge, literature furnishes no mathematical result linking the approximation of individual ConvNet elements and the final ConvNet performance. Thus, we adopt a greedy-like heuristic that consists in individually simplifying each computational structure of a ConvNet in the hope of finding a resulting structure capable of good performance [51].

In a ConvNet, two main types of mathematical elements are found: 1) matrix structures and 2) activation functions. The matrix structures are represented by the convolution filter weights, subsampling operations, and bias values; whereas the activation function is usually a nonlinear function such as the threshold, piecewise linear, and sigmoid functions [52].

To approximate these two classes of elements, different tools are required. For the matrix-based structures, we selected matrix approximation methods as a venue to derive low-complexity computational elements [53]–[56]. For the activation, we separate methods capable of approximating functions with efficient digital implementation [57]–[59].

B. Low-Complexity Matrix Structures

In [54], [57], and [61]–[64], several methods for deriving approximations of discrete transform matrices—such as the discrete cosine transform [64]—were proposed. Let $\mathbf{M}$ be an $N \times N$ given matrix. For instance, $\mathbf{M}$ can be a convolutional filter. In this case, a computational instantiation of $\mathbf{M}$ applied to evaluate a single output pixel requires, in principle, N^2 floating point multiplications. A typical ConvNet may contain thousands of convolutional filters. For example, the classical architecture described in [12] contains 244 760 filters, which is nowadays considered a relatively small network. Therefore, to minimize such a significant computational cost, we aim at obtaining a low-complexity matrix $\hat{\mathbf{M}}$ capable of satisfying the following relation in an optimal sense: $\mathbf{M} \approx \hat{\mathbf{M}}$. The matrix $\hat{\mathbf{M}}$ is said to be an approximation for $\mathbf{M}$. Such approximate convolutional filters would allow the realization of computationally intensive ConvNets in the limited resources architectures.

In this paper, a low-complexity matrix is a matrix of dyadic rational entries. Dyadic rational numbers are the fractions of the form $m/2^n$, where n is a positive integer and m is an odd integer. Such numbers are suitable for binary arithmetic. Indeed, a multiplication by a dyadic rational consists of a multiplication by m followed by a right shift of n bits. Because m is an integer, we can take full advantage of fixed-point arithmetic. Indeed, m can be given a binary representation with the minimum number of adders, aiming at multiplicative irreducibility. Multiplicative irreducibility is attained whenever the minimum number of additions to implement a multiplication by m is equal to the number of ones in the binary representation of m [63]. Multiplicative irreducibility is often obtained when the CSD representation is considered [65]. Therefore, a multiplication by m can be converted into a sequence of additions and bit-shifting operations. As a consequence, low-complexity matrices are multiplierless, a very desirable property as floating-point operations are much more costly than additions and bit-shifting operations.

Standard methods for matrix approximation include: inspection [66], matrix parametrization [60], and matrix factorization [67]. However, since a typical ConvNet may contain from thousands to millions of filters, inspection-based approaches are not feasible. Methods based on the parametrization of the matrix elements are also ineffective, because: 1) the elements of convolutional filters are usually not clearly related, i.e., they do not satisfy identifiable mathematical relationships and 2) the elements are not repeated. In addition, ConvNet filters are not expected to satisfy properties, such as symmetry and orthogonality, which favors the derivation of approximations. Thus, methods based on matrix factorizations are less adequate.

C. Matrix Approximation by Linear Programming

We adopted a general approach to the problem of obtaining $\hat{\mathbf{M}}$ according to an optimization problem as described below

$$\hat{\mathbf{M}} = \arg\min_{\mathbf{T}} \operatorname{error}(\mathbf{M}, \mathbf{T}) . \tag{1}$$

The above-mentioned optimization problem can yield better approximate matrices if an expansion factor α is introduced [63]. By adopting the usual Frobenius norm [55] as an error measure, (1) can be recast according to the following mixed integer nonlinear programming (INLP) setup [68]:

$$(\alpha^*, \mathbf{T}^*) = \arg\min_{\alpha, \mathbf{T}} \|\mathbf{M} - \alpha \cdot \mathbf{T}\|^2 \tag{2}$$

where $\alpha > 0$ is the real-valued expansion factor and $\|\cdot\|$ is the Frobenius norm [55]. The choice of the Frobenius norm is justified by the following argument. An approximate CNN must have its elements numerically "close" to elements from the exact CNN. Therefore, a measure that takes into consideration distance in an energy-based manner (Euclidean distance sense) emerges naturally as a means to guarantee that the approximate filtering structures (e.g., convolution kernels) are close to the exact counterpart. The Frobenius norm satisfies the above rationale. This analysis is confirmed in [39].

To ensure that the candidate matrices $\mathbf{T}$ have low complexity, we limited the search space of the above problem to the matrices whose elements are defined over a sets of dyadic rationals $\mathscr{D}$. Some particular sets are [56], [66]

$$\mathscr{D}_1 = \{-1, 0, 1\}$$
$$\mathscr{D}_2 = \{-2, -1, 0, 1, 2\}$$
$$\mathscr{D}_3 = \{-4, -3, -2, -1, 0, 1, 2, 3, 4\}$$
$$\mathscr{D}_4 = \left\{-4, -3, -2, -1, -\frac{3}{4}, -\frac{1}{2}, -\frac{1}{4}, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 2, 3, 4\right\}$$
$$\mathscr{D}_5 = \left\{-7, -6, -5, -4, -3, -2, -1, -\frac{3}{4}, -\frac{1}{2}, -\frac{1}{4}, 0, \frac{1}{4}, \frac{1}{2}, \frac{3}{4}, 1, 2, 3, 4, 5, 6, 7\right\}$$

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

$$\mathscr{D}_6 = \left\{-4, -\frac{15}{4}, -\frac{7}{2}, -\frac{13}{4}, \dots, \frac{13}{4}, \frac{7}{2}, \frac{15}{4}, 4\right\}$$
$$\mathscr{D}_7 = \left\{-5, -\frac{19}{4}, -\frac{9}{2}, -\frac{17}{4}, \dots, \frac{17}{4}, \frac{9}{2}, \frac{19}{4}, 5\right\}$$
$$\mathscr{D}_8 = \left\{-7, -\frac{27}{4}, -\frac{13}{2}, -\frac{25}{4}, \dots, \frac{25}{4}, \frac{13}{2}, \frac{27}{4}, 7\right\}.$$

Sets $\mathscr{D}_6$, $\mathscr{D}_7$, and $\mathscr{D}_8$ possess uniformly spaced rationals.

A straightforward way of addressing (2) is as follows. Considering a given set of dyadic rationals $\mathscr{D}$, for each element of $\alpha \cdot \mathbf{T}$, we simply find the closest neighbor of such element in $\mathbf{D}$. Such approach can be efficiently implemented by means of binary search. However, this approach is only effective as long as (2) remains an unconstrained optimization problem. Alternatively, we can consider a more flexible approach based on integer linear programming (ILP).

For fixed values of α, the mixed INLP problem posed in (2) can be efficiently solved by means of binary (zero and one) linear programming. In other words, we aim at converting a nonlinear problem into a linear one. Indeed, let $m_{i,j}$, $i, j = 1, 2, \dots, N$, denote the entries of $\mathbf{M}$ and $r \in \mathscr{D}$ be a dyadic rational. We adopt the following binary decision variables:

$$x_{i,j}(r) = \begin{cases} 1, & \text{if } m_{i,j} = r \\ 0, & \text{otherwise.} \end{cases}$$

For binary (zero and one) variables, we have $y^2 = y$, where y is a dummy variable. This fact paves the way for the linearization of the above-mentioned optimization problem. Therefore, (2) can be rewritten according to the following binary linear programming problem [69]–[71]:

$$\begin{aligned} \min_{x_{i,j}(r)} \ & \sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{r\in\mathscr{D}} (r - \alpha \cdot m_{i,j})^2 \cdot x_{i,j}(r) \\ \text{s.t.} \ & \sum_{r\in\mathscr{D}} x_{i,j}(r) = 1, i, \quad j = 1, 2, \dots, N. \end{aligned} \tag{3}$$

The above-mentioned constraint is to ensure that each element $m_{i,j}$ is approximated by a unique dyadic rational in $\mathscr{D}$. The solution of the above problem is denoted as $x_{i,j}^{(\alpha)}$, $i, j = 1, 2, \dots, N$, $r \in \mathscr{D}$, being linked to the choice of α. Such binary (zero and one) solution can be employed to compute the actual entries $t_{i,j}^{(\alpha)}$ of the low-complexity matrix associated with the considered α according to

$$t_{i,j}^{(\alpha)} = \sum_{r\in\mathscr{D}} r \cdot x_{i,j}^{(\alpha)}(r). \tag{4}$$

The resulting low-complexity matrix is denoted by $\mathbf{T}_\alpha$. The approximation error is implied by (2) and can be computed according to

$$\text{Error}(\alpha) = \|\mathbf{M} - \alpha \cdot \mathbf{T}_\alpha\|^2.$$

Because a sequence of values for α is selected, the above problem is solved for each instantiation; furnishing the sequence of errors indexed by α: Error(α). Being a linear programming problem, each instantiation can be solved efficiently and very quickly by contemporary computational packages [72], [73]. State-of-the-art solvers can obtain solutions for ILP problems at an average computation complexity in $\mathscr{O}(N)$ [74] or $\mathscr{O}(N \log N)$ [69]–[71]. Finally, we determine the global optimum value α^* according to

$$\alpha^* = \arg\min_{\alpha} \text{Error}(\alpha) \tag{5}$$

which can be solved by the simple minimization over a vector of values. Associated with α^*, we also obtain $\mathbf{T}^* \triangleq \mathbf{T}_{\alpha^*}$, which is the global optimal low-complexity matrix. Therefore, the sought approximation $\hat{\mathbf{M}}$ is given by

$$\hat{\mathbf{M}} = \alpha^* \cdot \mathbf{T}^*. \tag{6}$$

The above-mentioned ILP approach allows the user to easily include constraints to the optimization problem. This is relevant for further investigation in this topic; in particular, when the specific mathematical properties are expected to be enforced on the resulting low-complexity matrices (for instance, 2-D filter normalization [75, p. 115]).

We emphasize that the solving method for (2) is only required to be efficient enough to cope with the time constraints at the design phase of the approximate neural network. In other words, solvers available in contemporary optimization packages are suitable; and the choice of the particular method for solving (2) is not critical for our approach. In addition, we note that the optimization solver is simply a step for obtaining the final neural network. Once the approximate structures are found, optimization solver is clearly not required anymore.

D. Example

To illustrate the procedure, we selected $\mathscr{D}_8$ and considered the search space of the expansion factor to be the interval $[0.25, 1]$ with a step of 10^{-3}. In addition, we consider the following particular convolutional filter employed in the ConvNet described in Section IV-B:

$$\mathbf{M}_0 = \begin{bmatrix} 1.5200701 & 1.0317051 & 0.7906240 & -0.2153791 & -0.2340538 \\ 1.3982610 & 2.1860176 & 2.0152923 & 1.5620477 & 0.8270900 \\ -0.6848867 & 0.7470516 & 1.6923728 & 1.2537112 & 1.1946758 \\ -1.2387477 & -0.5483563 & 0.1261987 & 0.8677799 & 0.7742613 \\ -1.4691808 & -1.2178997 & -0.2924347 & 0.2172496 & 0.1325074 \end{bmatrix}.$$

Solving (2) for the above matrix, we obtain

$$\begin{aligned} \alpha^* &= 0.30931 \\ \mathbf{T}^* &= \begin{bmatrix} 5 & 3.25 & 2.5 & -0.75 & -0.75 \\ 4.5 & 7 & 6.5 & 5 & 2.75 \\ -2.25 & 2.5 & 5.5 & 4 & 3.75 \\ -4 & -1.75 & 0.5 & 2.75 & 2.5 \\ -4.75 & -4 & -1 & 0.75 & 0.5 \end{bmatrix} \\ &= \frac{1}{4} \cdot \begin{bmatrix} 20 & 13 & 10 & -3 & -3 \\ 18 & 28 & 26 & 20 & 11 \\ -9 & 10 & 22 & 16 & 15 \\ -16 & -7 & 2 & 11 & 10 \\ -19 & -16 & -4 & 3 & 2 \end{bmatrix}. \end{aligned}$$

Fig. 1(a) depicts the Frobenius norm error for varying values of α [see (2)]. For very small α, the values of $\alpha \cdot \mathbf{M}$ are close to zero. Therefore, the discrete entries of the candidate matrices $\mathbf{T}$ are unable to provide a good approximation. As α

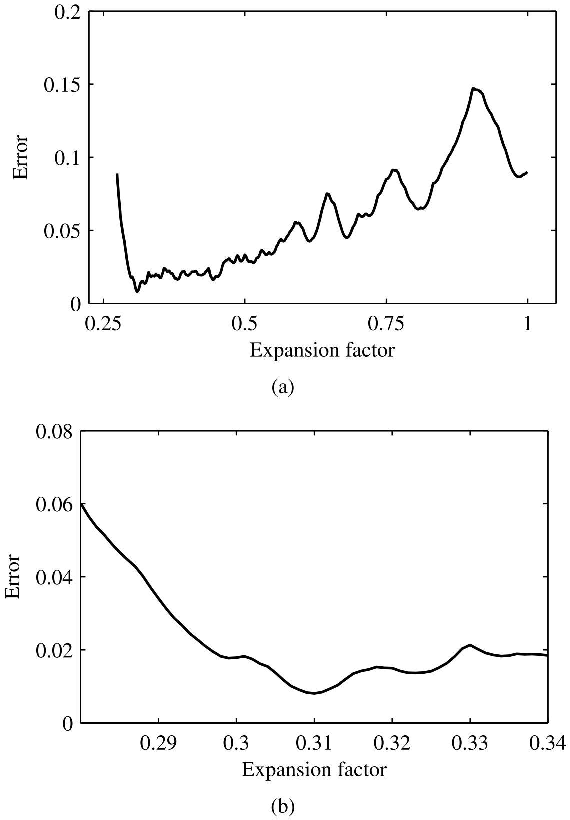

Fig. 1. Approximation error for the particular matrix $\mathbf{M}_0$. (a) Error curve over the considered search interval for α. (b) Detailed view around the optimum value of α.

increases, a similar effect happens. However, for intermediate values, the minimum can be found. Fig. 1(b) shows details in the vicinity of the optimum. The curves shown in Fig. 1 are piecewise concatenations of parabolae. This is due to the quadratic nature of the coefficients $(r - \alpha \cdot m_{i,j})^2$ of the linear programming problem in (3). Each parabola is linked to a particular approximate candidate $\mathbf{T}$.

Note that the low-complexity matrix $\mathbf{T}^*$ is expressed in terms of small integers, which can be given simple binary expansions (e.g., $22 = 2^6 - 2^4 - 2^2$). Similarly, we have that $\alpha^* = 0.30931 \approx 2^{-2} + 2^{-4} - 2^{-8} = 0.30859375$.

Therefore, considering (6), the actual fully multiplierless approximation is furnished by

$$\hat{\mathbf{M}} = (2^{-4} + 2^{-6} - 2^{-10}) \cdot \begin{bmatrix} 20 & 13 & 10 & -3 & -3 \\ 18 & 28 & 26 & 20 & 11 \\ -9 & 10 & 22 & 16 & 15 \\ -16 & -7 & 2 & 11 & 10 \\ -19 & -16 & -4 & 3 & 2 \end{bmatrix}.$$

E. Activation Function Approximations

Although there are several types of activation functions, we focus our analyses on the continuous tanh-sigmoid function, which is defined according to the hyperbolic tangent function [52]. As indicated in [53] and [77], the mathematical expression for the tanh-sigmoid functon is given by

$$\phi(x) = a \cdot \tanh(b \cdot x) \tag{7}$$

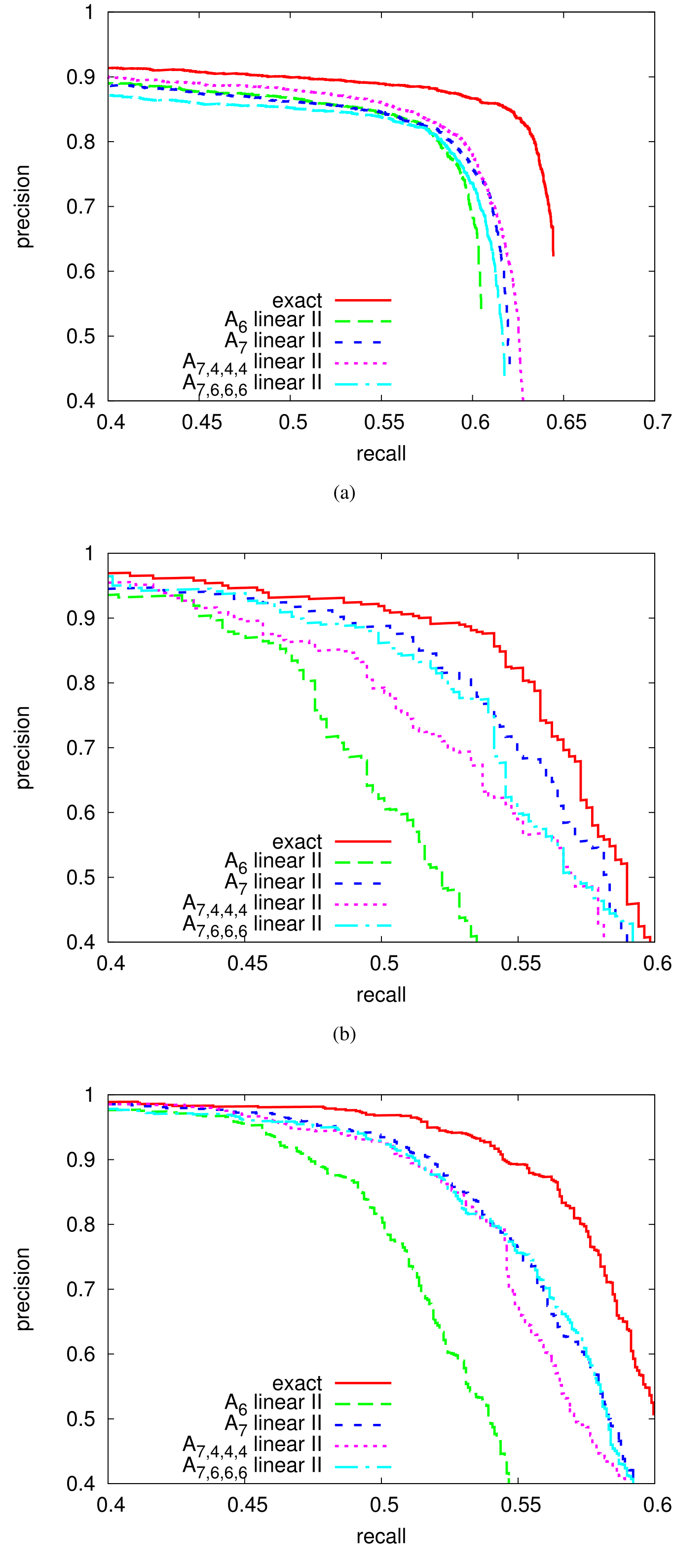

Fig. 2. ROC curves for (a) face detection data set and benchmark (FDDB), (b) annotated faces in the wild (AFW), and (c) Pascal data sets comparing the face detection performance of the original (exact) ConvNet model with different approximations.

where $a = 1.7159$ and $b = 2/3$. This particular activation function has been originally proposed by LeCun [77] and adopted in several working models as the CFF [6].

TABLE I
APPROXIMATIONS FOR THE TANH-SIGMOID

ASG-based	$\sigma_1(x) = \hat{a} \cdot \begin{cases} -1 + \frac{1+\frac{\lvert\lfloor x \rfloor\rvert - x}{2}}{2^{\lvert x \rvert}}, & x < 0, \\ 1 - \frac{1+\frac{\lvert\lfloor x \rfloor\rvert - x}{2}}{2^{\lvert x \rvert}}, & x \geq 0. \end{cases}$	PLAN-based	$\sigma_2(x) = \hat{a} \cdot \begin{cases} -1, & x < -5, \\ \frac{x}{16} - \frac{89}{128}, & -5 \leq x < -\frac{19}{8}, \\ \frac{x}{4} - \frac{1}{4}, & -\frac{19}{8} \leq x < -1, \\ \frac{x}{2}, & -1 \leq x < 1, \\ \frac{x}{4} + \frac{1}{4}, & 1 \leq x < \frac{19}{8}, \\ \frac{x}{16} + \frac{11}{16}, & \frac{19}{8} \leq x < 5, \\ 1, & x \geq 5. \end{cases}$
Linear I	$\sigma_3(x) = \hat{a} \cdot \begin{cases} -1, & x < -4, \\ \frac{x}{4}, & -4 \leq x < 4, \\ 1, & x \geq 4. \end{cases}$	Linear II	$\sigma_4(x) = \hat{a} \cdot \begin{cases} -1, & x < -2, \\ \frac{x}{2}, & -2 \leq x < 2, \\ 1, & x \geq 2. \end{cases}$
Quadratic I	$\sigma_6(x) = \hat{a} \cdot \begin{cases} -1, & x < -4, \\ (\frac{x}{4}+1)^2 - 1, & -4 \leq x < 0, \\ 1 - (\frac{x}{4}+1)^2, & 0 \leq x < 4, \\ 1, & x \geq 4. \end{cases}$	Quadratic II	$\sigma_6(x) = \hat{a} \cdot \begin{cases} -1, & x < -2, \\ (\frac{x}{2}+1)^2 - 1, & -2 \leq x < 0, \\ 1 - (\frac{x}{2}+1)^2, & 0 \leq x < 2, \\ 1, & x \geq 2. \end{cases}$

In [58], [60], and [79]–[82], approximations for the related sigmoid function given by $y = 1/(1 + e^{-x})$ were examined, including the Alippi and Storti–Gajani (ASG) approximation [81], the piecewise linear approximation of a nonlinear function (PLAN) [82], and simple linear [57] and quadratic [58] approximations. Based on these approximations, we derived expressions for the tanh-sigmoid approximations, as shown in Table I. We adopted an 8-bit representation for a resulting in the following approximate value: $\hat{a} = 7/4$.

F. Complexity

As a consequence of the above approximations, we have substantial savings in computation costs. Indeed, a single call of the original $N \times N$ matrix $\mathbf{M}$ requires N^2 multiplications of floating-point entries per pixel. On the other hand, the proposed approximation $\hat{\mathbf{M}}$ contains only small integers that can be very efficiently encoded with a minimal number of adders [65]. Similarly, the expansion factor α^* can be given a truncated rational approximation in the form of dyadic rationals. The same rationale also applies to the remaining computational structures of the original ConvNet. Thus, the final resulting structure is fully *multiplierless*—only additions and bit-shifting operations are required. In terms of hardware realization, the number of arithmetic operations translates into chip area and power consumption [83], [84]. Thus, in limited resource scenarios (e.g., embedded systems and wireless sensors), approximations may provide an effective way of porting large ConvNets into physical realization.

To summarize, the proposed approximation approach consists of the following:

1) finding approximate convolutional filter by solving (1) for each exact convolutional filter from a given ConvNet;
2) converting scaling factors, subsampling coefficients, and bias values into CSD representation aiming at the minimization of computation costs and multiplicative irreducibility;
3) approximating the activation function to a simple function.

TABLE II
ARITHMETIC COST FOR CFF-BASED MODELS

Model	Operation			
	Mult.	Add.	CSD Add.	Bit-shifting
Exact	882	843	-	-
A_1	0	843	235	346
A_2	0	843	251	362
A_3	0	843	377	488
A_4	0	843	457	568
A_5	0	843	506	617
A_6	0	843	756	867
A_7	0	843	842	953
A_8	0	843	1028	1139

IV. EXPERIMENTS

We studied the effectiveness of the proposed approximation approach on two classical computer vision problems: 1) a binary and 2) a multiclass classification problem. The first application is face detection, where the ConvNet classifies image regions as face or nonface. The second one is handwritten digit recognition, where the trained model is used to classify a given image patch into one of the ten digits “0” to “9.” For each of the two applications, we trained a ConvNet in a classical way and evaluated its performance in terms of precision and recall, for the given application.

The first ConvNet is relatively small, whereas the second model (for digit recognition) contains much more parameters. We aim at demonstrating that our proposed approach is able to effectively process larger networks.

After approximating the parameters of the models, we compared their performance with their respective original, exact

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE III
AVERAGE PRECISION FOR CFF WITH THE FDDB TEST SET AND DIFFERENT APPROXIMATIONS RELATIVE TO THE EXACT MODEL

	Exact	ASG	PLAN	Linear I	Linear II	Quadratic I	Quadratic II
Exact	1.000	0.953	0.894	0.002	0.988	0.887	0.938
A_1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
A_2	0.001	0.001	0.000	0.000	0.001	0.000	0.002
A_3	0.549	0.311	0.432	0.005	0.510	0.426	0.180
A_4	0.523	0.602	0.365	0.001	0.558	0.383	0.602
A_5	0.490	0.098	0.517	0.000	0.657	0.510	0.073
A_6	0.938	0.914	0.817	0.002	0.888	0.797	0.949
A_7	0.960	0.943	0.847	0.002	**0.963**	0.848	0.935
A_8	0.821	0.792	0.648	0.001	0.810	0.650	0.793
$A_{7,3,3,3}$	0.917	0.902	0.886	0.003	0.947	0.888	0.851
$A_{7,4,4,4}$	0.967	0.931	0.855	0.000	**0.976**	0.861	0.928
$A_{7,6,6,6}$	0.959	0.907	0.862	0.004	**0.959**	0.863	0.933

TABLE IV
AVERAGE PRECISION FOR CFF WITH THE AFW TEST SET AND DIFFERENT APPROXIMATIONS RELATIVE TO THE EXACT MODEL

	Exact	ASG	PLAN	Linear I	Linear II	Quadratic I	Quadratic II
Exact	1.000	0.839	0.829	0.000	1.041	0.797	0.383
A_1	0.000	0.000	0.000	0.000	0.000	0.000	0.000
A_2	0.001	0.001	0.000	0.000	0.000	0.000	0.000
A_3	0.220	0.041	0.260	0.004	0.199	0.248	0.014
A_4	0.422	0.356	0.317	0.000	0.457	0.336	0.275
A_5	0.220	0.015	0.314	0.000	0.217	0.251	0.002
A_6	0.864	0.794	0.715	0.000	0.893	0.694	0.700
A_7	0.978	0.844	0.753	0.000	**0.985**	0.755	0.614
A_8	0.698	0.553	0.576	0.000	0.565	0.544	0.169
$A_{7,3,3,3}$	0.955	0.525	0.835	0.004	0.914	0.816	0.184
$A_{7,4,4,4}$	1.020	0.827	0.755	0.000	**0.954**	0.761	0.576
$A_{7,6,6,6}$	0.967	0.881	0.787	0.000	**0.979**	0.788	0.827

versions. Note that we do not aim at improving the state of the art in face detection or hand-written digit recognition. Indeed, current literature presents concrete example of complex models, such as multiview or part-based detectors, for face detection [85] and huge ensemble classifiers for digit recognition [86]. Our goal is to demonstrate—based on common representative models—that the complexity of a given trained ConvNet model can be reduced significantly by approximating its parameters while maintaining a very similar performance.

Hereafter, an approximate network based on dyadic set $\mathscr{D}_i$ is referred to as A_i.

A. Binary Classification

Our first set of experiments employs a ConvNet that was trained for face detection in gray-scale images. Thus, such network is a binary classifier that decides whether the given input image is a face or not. As a working model, we selected the classical face detector called CFF proposed by Garcia and Delakis [6]. This model is a relatively "light" ConvNet with an input size of 32×36 and six layers: four layers alternating convolution and average pooling operations, with 4, 4, 14, and 14 maps, respectively, followed by 14 neurons and one single final output neuron. The first convolution layer contains four filters of size 5×5, the second one contains 20 filters of size 3×3, and the 14 neurons of the first neuron layer are treated as convolutions of size 6×7, each neuron being connected to only one map. Pooling maps contain a single coefficient, and all maps and neurons have an additional bias. The entire ConvNet has 951 trainable parameters in total. A thorough description of this particular ConvNet is provided in [6]. The employed activation function is the exact continuous tanh-sigmoidal function as detailed in (7).

After training the ConvNet as described in [6], we approximated all the convolution filter matrices with low-complexity versions. We created several approximations using the different sets of dyadic rationals described in Section III-C: $\mathscr{D}_1, \mathscr{D}_1, \ldots, \mathscr{D}_8$. We also replaced all average pooling coefficients and bias terms with their closest CSD representation using 8 bits, being 7 bits for the fractional part.

Table II lists the arithmetic costs of the exact ConvNet compared to its approximations. Floating-point multiplications, direct additions, additions due to the CSD expansion,

This article has been accepted for inclusion in a future issue of this journal. Content is final as presented, with the exception of pagination.

TABLE V

AVERAGE PRECISION FOR CFF WITH THE "PASCAL FACES" TEST SET AND DIFFERENT APPROXIMATIONS RELATIVE TO THE EXACT MODEL

	Exact	ASG	PLAN	Linear I	Linear II	Quadratic I	Quadratic II
Exact	1.000	0.933	0.774	0.001	1.039	0.747	0.893
A_1	0.001	0.001	0.000	0.000	0.001	0.001	0.001
A_2	0.006	0.006	0.004	0.000	0.006	0.003	0.009
A_3	0.234	0.128	0.239	0.000	0.234	0.230	0.098
A_4	0.357	0.375	0.259	0.001	0.361	0.258	0.370
A_5	0.433	0.143	0.386	0.000	0.449	0.345	0.101
A_6	0.862	0.844	0.656	0.000	0.894	0.638	0.847
A_7	0.917	0.879	0.707	0.001	**0.971**	0.702	0.889
A_8	0.725	0.700	0.520	0.001	0.697	0.517	0.617
$A_{7,3,3,3}$	0.899	0.752	0.755	0.004	0.899	0.752	0.645
$A_{7,4,4,4}$	0.930	0.889	0.710	0.000	**0.967**	0.708	0.906
$A_{7,6,6,6}$	0.931	0.906	0.725	0.004	**0.970**	0.722	0.916

Fig. 3. Some CFF face detection results on the AFW data set. Top: exact model. Bottom: approximation $A_{7,3,3,3}$. Despite the very coarse approximation, the results are very close. In the second last image, the approximate model even detects an additional face, missed by the original CFF. However, in the last example, a false detection is produced.

and bit-shifting operations were counted. The exact structure requires both floating-point multiplications and additions. In contrast, the approximate methods completely eliminate the need for multiplications at the expense of much simpler operations: additions and bit-shifting operations. Because the approximate quantities can be easily represented in fixed-point arithmetic representation, it is suitable for hardware implementation. In addition, the hardware implementation of bit-shifting operations requires virtually no cost, because it can be implemented by simple physical wiring. As a result, we have a very favorable tradeoff: multiplications are exchanged for additions.

In order to analyze the effect of the approximation on the actual performance of the ConvNet, we evaluated the different versions on three standard face detection benchmarks: FDDB [87] (2845 images), AFW [88] (205 images), and Pascal Faces data set [89] (851 images); and we used the improved annotation and evaluation protocol proposed by Mathias *et al.* [85].

Tables III–V show the average precision rates for different combinations of sigmoid and weight matrix approximations relative to the exact model for the three data sets.

Overall, the "Linear II" sigmoid approximation provides the best results, followed by "ASG," "Quadratic I," and "PLAN." In terms of weight matrix approximations, A_1–A_5 generally give unsatisfactory results, and A_7 performs best. Also, it is interesting to note that the finer approximation A_8 gave worse results than A_7. We further evaluated some variants, where different layers of the ConvNet have been approximated with different sets of dyadic rationals. For such mixed approximate structures, we have denoted them by $A_{i,j,k,l}$, where the subscripts indicate the selected dyadic set for each layer. In other words, indices i, j indicate that the dyadic sets $\mathscr{D}_i$ and $\mathscr{D}_j$, respectively, are employed in the first two convolution layers; similarly, indices k, l correspond to the adoption of the dyadic sets $\mathscr{D}_k$ and $\mathscr{D}_l$, respectively, for the two final fully connected layers. We found that the first layer requires a finer approximation than the other layers. This allowed us to maintain a good performance with very low-complexity approximations (e.g., A_3 and A_4) for these later layers. This can be explained according to the following: 1) by its own very nature, the layers have different degrees of importance; 2) errors in initial layers tend to propagate

TABLE VI
ARITHMETIC COST FOR THE MNIST-BASED MODELS

Model	Operation			
	Mult.	Add.	CSD Add.	Bit-shifting
Exact	183375	178110	-	-
A_1	0	178110	12740	23325
A_2	0	178110	12722	23307
A_3	0	178110	49127	59712
A_4	0	178110	61228	71813
A_5	0	178110	65211	75796
A_6	0	178110	141401	151986
A_7	0	178110	158595	169180
A_8	0	178110	188417	199002

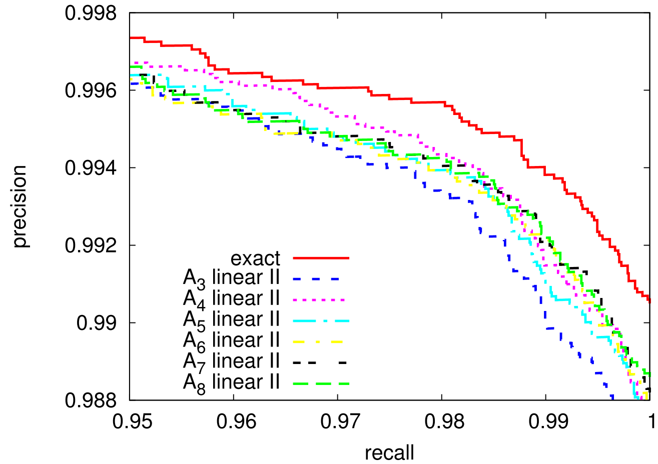

Fig. 4. ROC curves for the MNIST hand-written digit classification test set comparing the performance of the original (exact) ConvNet model with its approximations.

through the succeeding layers; and 3) error propagation can potentially be amplified along the layers. Fig. 2 shows the receiver operating characteristic (ROC) curves of the best performing approximations for the three data sets.

These results are quite impressive given the fact that we considerably reduced the precision of each parameter of the ConvNet, and given the highly nonlinear classification problem, where the frontier between the face and nonface classes can be very thin and complex.

Fig. 3 shows some face detection results from the exact model (top) and the approximation $A_{7,3,3,3}$ (bottom), i.e., a finer approximation for the first layer and a very coarse one for the rest of the layers. The results are almost identical.

B. Multiclass Classification

We studied a second case where a ConvNet has been trained for a classical multiclass classification problem: the MNIST hand-written digit recognition data set [90]. To show that the proposed approximations can also be applied to larger networks, we trained a ConvNet with a different architecture containing again six layers but much more maps and around 180 000 parameters and more than 5300 matrices in total. The input is a 32×32 gray-scale image, and the network is composed of five convolution maps (5×5 kernels) followed by the five average pooling maps (connected one to one), 50 convolution maps (3×3, fully connected), 50 average pooling maps (connected one-to-one), 100 neurons (6×6 matrices, fully connected), and the 10 final output neurons corresponding to the 10 digits to classify.

After having trained this ConvNet model on the MNIST data set, we approximated all the convolution filters, the fully connected layer matrices, and the activation functions which are based on the tanh-sigmoid function. Again, all pooling coefficients and bias terms were replaced by their closest CSD representation using 8 bits. The computation cost of the exact and approximate structures is shown in Table VI. Similar to the previous experiment, the approximate models have totally eliminated the multiplicative costs. Floating-point arithmetic is not required; being fixed-point arithmetic adequate. The cost of the extra additions due to the CSD representation is very low compared to the multiplicative cost required by the exact model. The cost of bit-shifting operations is negligible.

Table VII shows the relative classification rates on the MNIST test set for the different approximations, and Fig. 4 depicts the respective ROC curves of the best-performing approximations (combined for the 10 classes).

The results show that the approximations, although very coarse, have a very small effect on the overall performance of the ConvNet. In a multiclass setting, the trained ConvNet, at least in this particular case, is much more robust to the loss in precision of the weights induced by our approximation scheme compared to the binary classifier. For example, a very coarse approximation, such as $A_{3,3,1,1}$, leads to a relative performance decrease of less than 1%.

C. Large-Scale Deep Neural Networks

Finally, we applied our approximation approach to a deeper and more complex network architecture, the well-known AlexNet proposed by Krizhevsky *et al.* [12], and the ImageNet data set [13] for image classification with 1000 classes. This model contains more than 1.2 million matrices and 5096 vectors. We approximated all convolution filter matrices of the fully trained eight-layer ConvNet using two different sets of dyadic rationals for different layers, a very coarse set $\mathscr{D}_9$ and a slightly finer set $\mathscr{D}_{10}$

$$\mathscr{D}_9 = \left\{-2, -1, -\frac{1}{2}, -\frac{1}{8}, 0, \frac{1}{8}, \frac{1}{2}, 1, 2\right\}$$
$$\mathscr{D}_{10} = \left\{-2, -1, -\frac{1}{2}, -\frac{1}{4}, -\frac{1}{8}, 0, \frac{1}{8}, \frac{1}{4}, \frac{1}{2}, 1, 2\right\}.$$

Again, all other coefficients are approximated by their closest 8-bit CSD representation. The pooling layers do not have any coefficient here, and only linear and rectified linear units are used as activation function, which are already of very low complexity and thus do not require any approximation.

We used the ImageNet 2012 validation set to evaluate our different approximations. And, as usual in the literature,

TABLE VII

MEAN CLASSIFICATION RATES FOR THE MNIST TEST SET AND DIFFERENT APPROXIMATIONS RELATIVE TO THE EXACT MODEL

	Exact	ASG	PLAN	Linear I	Linear II	Quadratic I	Quadratic II
Exact	1.0000	1.0000	0.9847	0.9680	0.9978	1.0000	1.0000
A_1	0.9684	0.9684	0.9588	0.9260	0.9615	0.9684	0.9684
A_2	0.9643	0.9643	0.9627	0.8805	0.9573	0.9643	0.9643
A_3	0.9961	**0.9961**	0.9848	0.9655	**0.9944**	**0.9961**	**0.9961**
A_4	0.9973	**0.9973**	0.9863	0.9700	**0.9969**	**0.9973**	**0.9973**
A_5	0.9976	**0.9976**	0.9866	0.9666	**0.9969**	**0.9976**	**0.9976**
A_6	0.9991	**0.9991**	0.9868	0.9701	**0.9973**	**0.9991**	**0.9991**
A_7	0.9992	**0.9992**	0.9846	0.9680	**0.9977**	**0.9992**	**0.9992**
A_8	0.9994	**0.9994**	0.9848	0.9675	**0.9981**	**0.9994**	**0.9994**
$A_{3,3,1,1}$	0.9931	**0.9931**	0.9749	0.9625	**0.9924**	**0.9931**	**0.9931**
$A_{3,1,1,1}$	0.9891	0.9891	0.9684	0.9580	0.9866	0.9891	0.9891
$A_{4,4,1,1}$	0.9937	**0.9937**	0.9780	0.9618	**0.9943**	**0.9937**	**0.9937**
$A_{4,1,1,1}$	0.9885	0.9885	0.9655	0.9572	0.9872	0.9885	0.9885

TABLE VIII

CLASSIFICATION ACCURACY AND TOP-5 ACCURACY FOR IMAGENET AND DIFFERENT APPROXIMATIONS RELATIVE TO THE EXACT ALEXNET MODEL

	Absolute		Relative	
	Accuracy	Top-5	Accuracy	Top-5
Exact	0.5682	0.7995	1.0000	1.0000
A_9	0.4862	0.7288	0.8558	0.9117
A_{10}	**0.5463**	**0.7820**	**0.9616**	**0.9782**
$A_{10,9,9,9,9,9}$	0.5423	0.7794	0.9544	0.9750
$A_{10,10,9,9,9,9}$	0.5442	0.7796	0.9578	0.9751

we compute the classification accuracy as well as the top-5 accuracy for the 50 000 test images. Table VIII shows the results. The approximation A_{10} with the set $\mathscr{D}_{10}$ gives the best performance, with a relative decrease in accuracy of only 3.84% and 2.18% on the top-5 accuracy. However, as the following line shows, we can achieve almost the same performance using the coarser set $\mathscr{D}_9$ for all convolution layers except the first one. This again suggests that a finer approximation of the first layer is required to prevent a drastic performance drop.

V. CONCLUSION

We presented a novel scheme for approximating the parameters of a trained ConvNet, notably the convolution filters, neuron weights, as well as pooling and bias coefficients. Activation functions were also approximated. The particularity of the matrix approximations is that they allow for an extremely efficient implementation—software or hardware—using only additions and bit-shifts, and no multiplication. We thoroughly evaluated the impact of this parameter approximation measuring the overall performance of ConvNets on three different use cases: one smaller ConvNet for face detection, a larger ConvNet for hand-written digit classification, and a much more complex, deep ConvNet for large-scale image classification.

For all three models, our proposed scheme was able to produce low-complexity approximations without a significant loss in performance.

These results suggest that huge reductions in computational complexity of trained ConvNet models can be obtained, and extremely efficient hardware implementations can be realized. Further studies need to be undertaken to analyze the impact of this type of approximations for more use cases and different architectures.

REFERENCES

[1] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

[2] K. Chellapilla, S. Puri, and P. Simard, "High performance convolutional neural networks for document processing," in *Proc. 10th Int. Workshop Frontiers Handwriting Recognit. (IWFHR)*, 2006, pp. 1–7.

[3] M. Delakis and C. Garcia, "Text detection with convolutional neural networks," in *Proc. 3rd Int. Conf. Comput. Vis. Theory Appl. (VISAPP)*, vol. 2. Funchal, Madeira, Portugal, Jan. 2008, pp. 290–294.

[4] K. Elagouni, C. Garcia, F. Mamalet, and P. Sébillot, "Text recognition in multimedia documents: A study of two neural-based OCRs using and avoiding character segmentation," *Int. J. Document Anal. Recognit.*, vol. 17, no. 1, pp. 19–31, 2014.

[5] R. Vaillant, C. Monrocq, and Y. Le Cun, "An original approach for the localization of objects in images," *IEE Proc. Vis., Image Signal Process.*, vol. 141, no. 4, pp. 245–250, Aug. 1994.

[6] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1408–1423, Nov. 2004.

[7] M. Osadchy, M. L. Miller, and Y. LeCun, "Synergistic face detection and pose estimation with energy-based models," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2005, pp. 1017–1024.

[8] D. Ciresan, U. Meier, J. Masci, and J. Schmidhuber, "Multi-column deep neural network for traffic sign classification," *Neural Netw.*, vol. 32, pp. 333–338, Aug. 2012.

[9] P. Sermanet, K. Kavukcuoglu, S. Chintala, and Y. LeCun, "Pedestrian detection with unsupervised multi-stage feature learning," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Portland, OR, USA, Jun. 2013, pp. 3626–3633.

[10] R. Hadsell *et al.*, "Learning long-range vision for autonomous off-road driving," *J. Field Robot.*, vol. 26, no. 2, pp. 120–144, Feb. 2009.

[11] P. Sermanet, S. Chintala, and Y. LeCun, "Convolutional neural networks applied to house numbers digit classification," in *Proc. 21st Int. Conf. Pattern Recognit., (ICPR)*, Tsukuba, Japan, Nov. 2012, pp. 3288–3291.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Proc. 25th Adv. Neural Inf. Process. Syst.*, Lake Tahoe, NV, USA, 2012, pp. 1106–1114.

[13] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *Comput. Res. Repository*, pp. 1–14, Apr. 2014.

[14] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, Oct. 1986.

[15] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, and Y. LeCun, "OverFeat: Integrated recognition, localization and detection using convolutional networks," *Comput. Res. Repository*, pp. 1–16, Dec. 2013.

[16] P. K. Meher, S. Y. Park, B. K. Mohanty, K. S. Lim, and C. Yeo, "Efficient integer DCT architectures for HEVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 24, no. 1, pp. 168–178, Jan. 2014.

[17] Y.-J. Chen, S. Oraintara, and T. X. Nguyen, "Video compression using integer DCT," in *Proc. Int. Conf. Image Process.*, vol. 2. Sep. 2000, pp. 844–845.

[18] A. Joshi, A. K. Mishra, and R. Patrikar, "Design of real-time video watermarking based on integer dct for H.264 encoder," *Int. J. Electron.*, vol. 102, no. 1, pp. 141–155, 2015.

[19] V. A. Coutinho, R. J. Cintra, F. M. Bayer, P. A. M. Oliveira, R. S. Oliveira, and A. Madanayake, "Pruned discrete Tchebichef transform approximation for image compression," in *Circuits, Systems, and Signal Processing*. New York, NY, USA: Springer, 2018.

[20] D. Scherer, H. Schulz, and S. Behnke, "Accelerating large-scale convolutional neural networks with parallel graphics multiprocessors," in *Proc. Int. Conf. Artif. Neural Netw.*, 2010, pp. 82–91.

[21] B. A. White and M. I. Elmasry, "The digi-neocognitron: A digital neocognitron neural network model for VLSI," *IEEE Trans. Neural Netw.*, vol. 3, no. 1, pp. 73–85, Jan. 1992.

[22] M. Marchesi, G. Orlandi, F. Piazza, and A. Uncini, "Fast neural networks without multipliers," *IEEE Trans. Neural Netw.*, vol. 4, no. 1, pp. 53–62, Jan. 1993.

[23] H. K. Kwan and C. Z. Tang, "Multiplierless multilayer feedforward neural network design suitable for continuous input-output mapping," *Electron. Lett.*, vol. 29, no. 14, pp. 1259–1260, Jul. 1993.

[24] J. Vincent and D. Myers, "Weight dithering and wordlength selection for digital backpropagation networks," *BT Technol. J.*, vol. 10, no. 3, pp. 124–133, 1992.

[25] P. Y. Simard and H. P. Graf, "Backpropagation without multiplication," in *Proc. Adv. Neural Inf. Process. Syst.*, 1994, pp. 232–239.

[26] S. Draghici, "On the capabilities of neural networks using limited precision weights," *Neural Netw. J. Int. Neural Netw. Soc.*, vol. 15, no. 3, pp. 395–414, Apr. 2002.

[27] E. L. Machado *et al.*, "Computational cost reduction in learned transform classifications," *Comput. Res. Repository*, pp. 1–7, Apr. 2015.

[28] M. Courbariaux, Y. Bengio, and J.-P. David, "BinaryConnect: Training deep neural networks with binary weights during propagations," in *Proc. Annu. Conf. Neural Inf. Process. Syst.*, 2015, pp. 3123–3131.

[29] M. Kim and P. Smaragdis, "Bitwise neural networks," in *Proc. ICML Workshop Resour.-Efficient Mach. Learn.*, 2015. [Online]. Available: https://arxiv.org/pdf/1601.06071.pdf

[30] F. Mamalet, S. Roux, and C. Garcia, "Real-time video convolutional face finder on embedded platforms," *EURASIP J. Embedded Syst.*, vol. 2007, no. 1, p. 021724, Mar. 2007, doi: 10.1155/2007/21724.

[31] N. Farrugia, F. Mamalet, S. Roux, F. Yang, and M. Paindavoine, "Fast and robust face detection on a parallel optimized architecture implemented on FPGA," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 19, no. 4, pp. 597–602, Apr. 2009.

[32] C. Farabet, B. Martini, P. Akselrod, S. Talay, Y. LeCun, and E. Culurciello, "Hardware accelerated convolutional neural networks for synthetic vision systems," in *Proc. Int. Symp. Circuits Syst. (ISCAS)*, Paris, France, May/Jun. 2010, pp. 257–260.

[33] S. Chakradhar, M. Sankaradas, V. Jakkula, and S. Cadambi, "A dynamically configurable coprocessor for convolutional neural networks," in *Proc. Int. Symp. Comput. Archit.*, 2010, pp. 247–257.

[34] T. Chen *et al.*, "DianNao: A small-footprint high-throughput accelerator for ubiquitous machine-learning," in *Proc. 19th Int. Conf. Archit. Support Programm. Lang. Oper. Syst. (ASPLOS)*, 2014, pp. 269–284.

[35] C. Zhang, P. Li, G. Sun, Y. Guan, B. Xiao, and J. Cong, "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. ACM/SIGDA Int. Symp. Field-Programm. Gate Arrays-(FPGA)*, 2015, pp. 161–170.

[36] F. Mamalet and C. Garcia, "Simplifying convnets for fast learning," in *Proc. Artif. Neural Netw. Mach. Learn. (ICANN)*, Lausanne, Switzerland, 2012, pp. 58–65.

[37] M. Mathieu, M. Henaff, and Y. LeCun, "Fast training of convolutional networks through FFTs," in *Proc. Int. Conf. Learn. Represent.*, 2014, pp. 1–9.

[38] V. Vanhoucke, A. Senior, and M. Z. Mao, "Improving the speed of neural networks on CPUs," in *Proc. Deep Learn. Unsupervised Feature Learn. NIPS Workshop*, 2011, p. 4.

[39] K. Osawa and R. Yokota, "Evaluating the compression efficiency of the filters in convolutional neural networks," in *Artificial Neural Networks and Machine Learning* (Lecture Notes in Computer Science), vol. 10614, A. Lintas, S. Rovetta, P. Verschure, and A. Villa, Eds. Cham, Switzerland: Springer, 2017, pp. 459–466.

[40] J. Xue, J. Li, and Y. Gong, "Restructuring of deep neural network acoustic models with singular value decomposition," in *Proc. Interspeech*, 2013, pp. 2365–2369.

[41] T. N. Sainath, B. Kingsbury, V. Sindhwani, E. Arisoy, and B. Ramabhadran, "Low-rank matrix factorization for deep neural network training with high-dimensional output targets," in *Proc. ICASSP*, May 2013, pp. 6655–6659.

[42] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, and N. De Freitas, "Predicting parameters in deep learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 2148–2156.

[43] E. Denton, W. Zaremba, J. Bruna, Y. LeCun, and R. Fergus, "Exploiting linear structure within convolutional networks for efficient evaluation," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 1269–1277.

[44] Z. Yang *et al.*, "Deep fried convnets," in *Proc. Int. Conf. Comput. Vis.*, 2014, pp. 1476–1483.

[45] M. Jaderberg, A. Vedaldi, and A. Zisserman, "Speeding up convolutional neural networks with low rank expansions," in *Proc. Brit. Mach. Vis. Conf.*, Sep. 2014, pp. 1–13.

[46] V. Lebedev, Y. Ganin, M. Rakhuba, I. Oseledets, and V. Lempitsky, "Speeding up convolutional neural networks using fine-tuned CP decomposition," in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–18.

[47] M. Courbariaux, I. Hubara, D. Soudry, R. El-Yaniv, and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to $+1$ or -1," *Comput. Res. Repository*, Mar. 2016.

[48] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "XNOR-Net: Imagenet classification using binary convolutional neural networks," *CoRR*, Mar. 2016.

[49] M. Courbariaux, Y. Bengio, and J.-P. David, "Low precision arithmetic for deep learning," *Comput. Res. Repository*, pp. 1–9, Dec. 2014.

[50] D. Miyashita, E. H. Lee, and B. Murmann, "Convolutional neural networks using logarithmic data representation," *CoRR*, 2016.

[51] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein, *Introduction to Algorithms*, 3rd ed. Cambridge, MA, USA: MIT Press, Jul. 2009.

[52] S. O. Haykin, *Neural Networks*. Upper Saddle River, NJ, USA: Prentice-Hall, 1999.

[53] R. J. Cintra, "An integer approximation method for discrete sinusoidal transforms," *Circuits Syst. Signal Process.*, vol. 30, pp. 1481–1501, Dec. 2011.

[54] P. A. M. Oliveira, R. J. Cintra, F. M. Bayer, S. Kulasekera, and A. Madanayake, "A discrete Tchebichef transform approximation for image and video coding," *IEEE Signal Process. Lett.*, vol. 22, no. 8, pp. 1137–1141, Aug. 2015.

[55] G. A. F. Seber, *A Matrix Handbook for Statisticians*. Hoboken, NJ, USA: Wiley, 2007.

[56] C. J. Tablada, F. M. Bayer, and R. J. Cintra, "A class of DCT approximations based on the Feig–Winograd algorithm," *Signal Process.*, vol. 113, pp. 38–51, Aug. 2015.

[57] M. T. Tommiska, "Efficient digital implementation of the sigmoid function for reprogrammable logic," *IEE Proc.-Comput. Digit. Techn.*, vol. 150, no. 6, pp. 403–411, Nov. 2003.

[58] M. Zhang, S. Vassiliadis, and J. G. Delgado-Frias, "Sigmoid generators for neural computing using piecewise approximations," *IEEE Trans. Comput.*, vol. 45, no. 9, pp. 1045–1049, Sep. 1996.

[59] K. Basterretxea, J. M. Tarela, and I. del Campo, "Approximation of sigmoid function and the derivative for hardware implementation of artificial neurons," *IEE Proc.-Circuits, Devices Syst.*, vol. 151, no. 1, pp. 18–24, Feb. 2004.

[60] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A low-complexity parametric transform for image compression," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2011, pp. 2145–2148.

[61] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "A multiplication-free transform for image compression," in *Proc. 2nd Int. Conf. Signals, Circuits Syst.*, Monastir, Tunisia, 2008, pp. 1–4.

[62] S. Bouguezel, M. O. Ahmad, and M. N. S. Swamy, "Low-complexity 8×8 transform for image compression," *Electron. Lett.*, vol. 44, no. 10, pp. 1249–1250, Oct. 2008.

[63] V. Britanak, P. C. Yip, and K. R. Rao, *Discrete Cosine and Sine Transforms*. Orlando, FL, USA: Academic, 2007.

[64] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE Trans. Comput.*, vol. C-23, no. 1, pp. 90–93, Jan. 1974.

[65] A. G. Dempster and M. D. Macleod, "Constant integer multiplication using minimum adders," *IEE Proc.-Circuits, Devices Syst.*, vol. 141, pp. 407–413, Oct. 1994.

[66] R. J. Cintra, F. M. Bayer, and C. J. Tablada, "Low-complexity 8-point DCT approximations based on integer functions," *Signal Process.*, vol. 99, pp. 201–214, Jun. 2014.

[67] F. M. Bayer and R. J. Cintra, "DCT-like transform for image compression requires 14 additions only," *Electron. Lett.*, vol. 48, no. 15, pp. 919–921, Jul. 2012.

[68] J. Lee and S. Leyffer, *Mixed Integer Nonlinear Programming* (The IMA Volumes in Mathematics and its Applications). New York, NY, USA: Springer, 2011.

[69] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. New York, NY, USA: Dover, 1998.

[70] D. Bienstock and G. Nemhauser, *Integer Programming and Combinatorial Optimization* (Lecture Notes in Computer Science). New York, NY, USA: Springer, 2004.

[71] J. K. Lenstra and A. H. G. R. Kan, "Computational complexity of discrete optimization problems," *Ann. Discrete Math.*, vol. 4, pp. 121–140, Jan. 1979.

[72] *MATLAB, Version 7.10.0 (R2010a)*, Math-Works Inc., Natick, MA, USA, 2010.

[73] *R Core Team, R: A Language and Environment for Statistical Computing*, R Found. Statist. Comput., Vienna, Austria, 2013.

[74] N. Megiddo, "Linear programming in linear time when the dimension is fixed," *J. ACM*, vol. 31, no. 1, pp. 114–127, Jan. 1984.

[75] W. Burger and M. J. Burge, "Digital image processing: An algorithmic introduction using Java," in *Texts in Computer Science*, London, U.K.: Springer, 2016.

[76] Y. LeCun, "Generalization and network design strategies," *J. ACM*, vol. 31, no. 1, pp. 114–127, Jan. 1984.

[77] Y. Le Cun, "Generalization and network design strategies," in *Proc. Int. Conf. Connectionism Perspective*, Oct. 1988, pp. 143-155.

[78] D. Larkin, A. Kinane, V. Muresan, and N. O'Connor, "An efficient hardware architecture for a neural network activation function generator," in *Proc. Adv. Neural Netw. ISNN*, 2006, pp. 1319–1327.

[79] O. Temam, "A defect-tolerant accelerator for emerging high-performance applications," in *Proc. 39th Annu. Int. Symp. Comput. Archit.*, Jun. 2012, pp. 356–367.

[80] J. Schlessman, "Approximation of the sigmoid function and its derivative using a minimax approach," M.S. thesis, Lehigh Univ., Bethlehem, PA, USA, Aug. 2002. [Online]. Available: https://preserve.lehigh.edu/cgi/viewcontent.cgi?article=1752&context=etd

[81] C. Alippi and G. Storti-Gajani, "Simple approximation of sigmoidal functions: Realistic design of digital neural networks capable of learning," in *Proc. IEEE Int. Symp. Circuits Syst.*, Singapore, Jun. 1991, pp. 1505–1508.

[82] H. Amin, K. M. Curtis, and B. R. Hayes-Gill, "Piecewise linear approximation applied to nonlinear function of a neural network," *IEE Proc.-Circuits, Devices Syst.*, vol. 144, no. 6, pp. 313–317, Dec. 1997.

[83] R. E. Blahut, *Fast Algorithms for Digital Signal Processing*. Cambridge, U.K.: Cambridge Univ. Press, 2010.

[84] U. S. Potluri, A. Madanayake, R. J. Cintra, F. M. Bayer, S. Kulasekera, and A. Edirisuriya, "Improved 8-point approximate DCT for image and video compression requiring only 14 additions," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1727–1740, Jun. 2014.

[85] M. Mathias, R. Benenson, M. Pedersoli, and L. Van Gool, "Face detection without bells and whistles," in *Proc. Eur. Conf. Comput. Vis.*, 2014, pp. 720–735.

[86] D. C. Cireşan, U. Meier, and J. Schmidhuber, "Multi-column deep neural networks for image classification," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, 2012, pp. 3642–3649.

[87] V. Jain and E. Learned-Miller, "FDDB: A benchmark for face detection in unconstrained settings," Univ. Massachusetts, Amherst, MA, USA, Tech. Rep. UM-CS-2010-009, 2010.

[88] X. Zhu and D. Ramanan, "Face detection, pose estimation, and landmark localization in the wild," in *Proc. Int. Conf. Comput. Vis. Pattern Recognit.*, Jun. 2012, pp. 2879–2886.

[89] J. Yan, X. Zhang, Z. Lei, and S. Z. Li, "Face detection by structural models," *Image Vis. Comput.*, vol. 32, no. 10, pp. 790–799, 2014.

[90] Y. LeCun, C. Cortes, and C. J. Burges. (2015). "The MNIST database of handwritten digits." [Online]. Available: http://yann.lecun.com/exdb/mnist/

Renato J. Cintra (SM'10) received the B.Sc., M.Sc., and D.Sc. degrees in electrical engineering from the Universidade Federal de Pernambuco (UFPE), Recife, Brazil, in 1999, 2001, and 2005, respectively.

From 2014 to 2015, he was a Visiting Researcher with the Institut national des sciences appliquées de Lyon, Lyon, France. He is currently a Visiting Professor with the University of Calgary, Calgary, AB, Canada. He is also an Associate Professor with the Departamento de Estatística, UFPE. His current research interests include theory and methods for digital signal and image processing, numerical analysis, and applied probability.

Dr. Cintra is an Society for Industrial and Applied Mathematics Member. He serves as an Associate Editor for the IEEE GEOSCIENCE AND REMOTE SENSING LETTERS, the *Circuits, Systems, and Signal Processing* (Springer), the *IET Circuits, Devices & Systems*, and the *Journal of Communication and Information Systems* (SBrT/Comsoc).

Stefan Duffner received the bachelor's degree in computer science from the Konstanz University of Applied Sciences, Konstanz, Germany, in 2002, and the master's degree in applied computer science and the Ph.D. degree in computer science from the University of Freiburg, Freiburg im Breisgau, Germany, in 2004 and 2008, respectively. His Ph.D. thesis was on face image analysis with statistical machine learning methods, notably convolutional neural networks, with the Orange Labs, Rennes, France.

He was a Post-Doctoral Researcher with the Idiap Research Institute, Martigny, Switzerland, in the field of computer vision and video analysis for visual object tracking applications, for four years. He is currently an Associate Professor with the Laboratory of Image Informatics and Information Systems, IMAGINE Team, Institut National des Sciences Appliquées de Lyon, Villeurbanne, France.

Christophe Garcia was the Head of the Laboratory of Image Informatics and Information Systems, IMAGINE Research Team, Villeurbanne, France, from 2011 to 2015. He is currently a Full Professor with the Institut National des Sciences Appliquées de Lyon, where he is currently a Deputy Director of the Laboratory of Image Informatics and Information Systems and the Vice-President of the Research in Information and Digital Society. He has published over 140 articles in international conferences and journals. He holds 17 industrial patents. His current research interests include the areas of convolutional neural networks, neural networks, pattern recognition, and computer vision.

Dr. Garcia has served as an Associate Editor for the *Journal of Visual Communication and Image Representation* (Elsevier), *Image and Video Processing* (Hindawi), and *Pattern Analysis and Application* (Springer-Verlag).

André Leite received the B.Sc., M.Sc., and D.Sc. degrees in electrical engineering from the Universidade Federal de Pernambuco (UFPE), Recife, Brazil, in 2003, 2005, and 2009, respectively.

Since 2010, he has been with the Department of Statistics, UFPE. His current research interests include dynamical systems, control theory and optimization, fundamentals of probability, and decision theory.