Training Lightweight Deep Convolutional Neural Networks using Bag-of-Features Pooling

Nikolaos Passalis and Anastasios Tefas

Abstract

Convolutional Neural Networks (CNNs) are predominantly used for several challenging computer vision tasks achieving state-of-the-art performance. However, CNNs are complex models that require the use of powerful hardware, both for training and deploying them. To this end, a quantization-based pooling method is proposed in this paper. The proposed method is inspired from the Bag-of-Features (BoF) model and can be used for learning more lightweight deep neural networks. Trainable Radial Basis Function (RBF) neurons are used to quantize the activations of the final convolutional layer, reducing the number of parameters in the network and allowing for natively classifying images of various sizes. The proposed method employs differentiable quantization and aggregation layers leading to an end-to-end trainable CNN architecture. Furthermore, a fast linear variant of the proposed method is introduced and discussed, providing new insight for understanding convolutional neural architectures. The ability of the proposed method to reduce the size of CNNs and increase the performance over other competitive methods is demonstrated using seven datasets and three different learning tasks (classification, regression and retrieval).

Index Terms

Bag-of-Features, Convolutional Neural Networks, Pooling Operators, Lightweight Neural Networks

I. Introduction

Convolutional Neural Networks (CNNs) are predominately used for several challenging computer vision tasks, e.g., action recognition [1], image classification [2]–[4], and supervised hashing [5], [6]. A typical CNN is composed of a convolutional feature extraction block followed by a fully connected block. The convolutional feature extraction block consists of a series of convolutional layers and pooling layers. Several different architectures for the feature extraction block have been proposed in the literature [7]–[11]. After extracting the feature maps from the convolutional feature extraction block, they are flattened into a feature vector that is used by the fully connected block to perform the task at hand. Note that the parameters of a CNN are learned using the well-known back-propagation algorithm [12].

Many CNN architectures require the input images to have a predefined size and are incapable of handling arbitrary sized images, since the dimensionality of the output of the convolutional feature extraction block depends on the input image size. Furthermore, the cost of feed-forwarding the network also depends on the size of the input images. Therefore, the cost of the feed-forward process cannot be adjusted to fit the available computational resources without re-training the network. This renders applications on embedded and mobile systems, e.g., real-time systems on drones, that often requires adjusting to the available computational resources, especially difficult. Also, often the size of the feature vector extracted from the feature extraction block is large, leading to exceptionally large fully connected layers. For example, it is worth noting that for the well-known VGG-16 network [10], the 90% of the network's parameters are spent on the fully connected block, while only 10% of the network's parameters are used on the previous 13 layers of the network.

To overcome some of the previous limitations global pooling operators have been proposed [8], [13]. Global pooling methods are capable of extracting a relatively small feature vector from the feature extraction layers, which is fixed and does not depend on the size of the network's input. However, even though global pooling allows a network to handle images of arbitrary size, its ability to provide scale invariance is limited, as it is also shown in Section IV. This can be attributed to the fact that the scale invariance is not achieved by learning a scale-invariant pooling layer along with the convolutional filters, but merely by learning scale-invariant convolutional filters. Learning both the convolutional layers and the pooling layer can often provide better scale-invariance, while also reducing the number of parameters needed, as demonstrated in this paper. Finally, note that compression methods, e.g., [14]–[16], can be used in conjunction with the aforementioned methods to further reduce the size of CNNs.

A novel layer inspired by the Bag-of-Feature model (also known as Bag-of-Visual-Words (BoVW) model) [17], [18], is proposed in this paper. The proposed layer works as a trainable quantization-based pooling layer and allows for overcoming the aforementioned limitations. The choice of the BoF model for this task was motivated by the fact that the BoF model was originally proposed to resolve similar problems, i.e., to provide a compact representation, while handling objects that are composed of a variable number of feature vectors, and provide better scale and position invariance. This can be better understood by considering the way that the BoF model works: First, a feature extractor, e.g., SIFT extractor [19], is employed

Nikolaos Passalis and Anastasios Tefas are with the Department of Informatics, Aristotle University of Thessaloniki, Thessaloniki, 54124, Greece, e-mail: passalis@csd.auth.gr, tefas@aiia.csd.auth.gr

Manuscript received April 19, 2005; revised August 26, 2015.

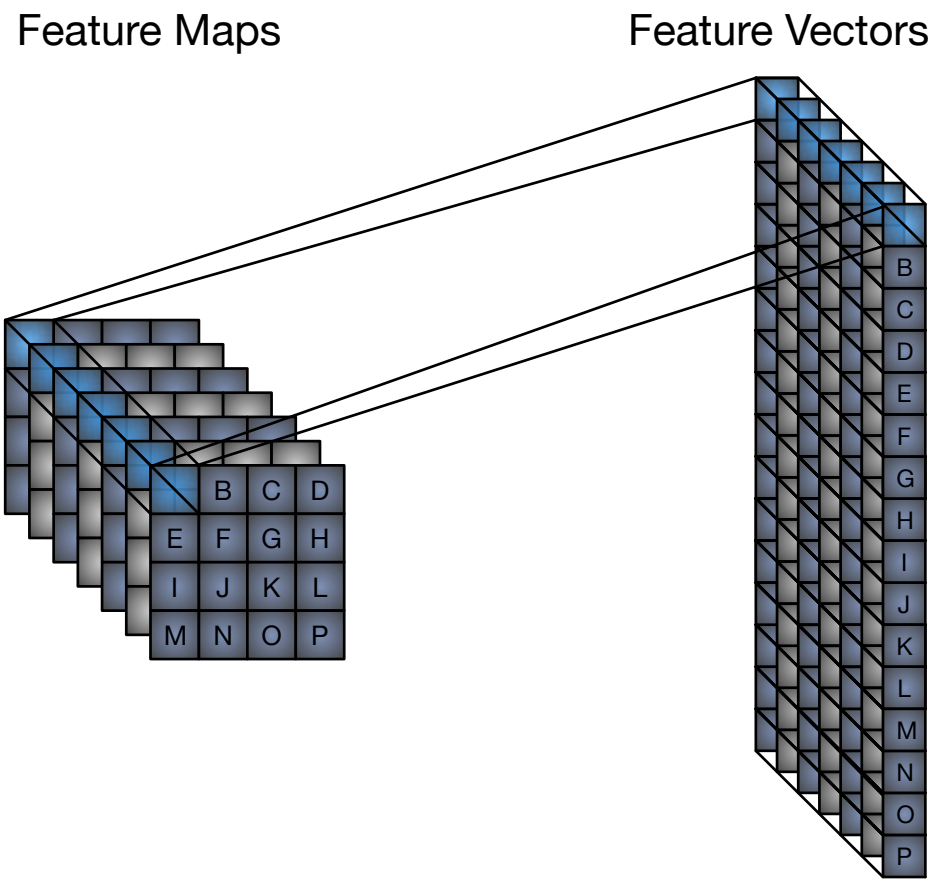

Fig. 1. Transforming feature maps into feature vectors that can be used with BoF-based techniques.

to extract multiple feature vectors from an image. The number of the extracted feature vectors might vary according to the type of the feature extractor that it is used. Then, a quantization process follows, where the extracted feature vectors are assigned to a predefined number of bins, that are called *codewords*. The set of all codewords is also called *codebook* or *dictionary*. The representation of an object is compiled by counting the number of feature vectors in each bin and then extracting the final histogram representation of the corresponding object. Note that a feature map can be readily converted into a set of feature vectors, as shown in Figure 1. Therefore, the proposed BoF pooling is used to extract a compact histogram representation from the last convolutional layer, instead of having the fully connected layer to directly handle the extracted feature maps, reducing the size of the subsequent layers and allowing the network to operate with arbitrary sized images.

The main contribution of this work is the proposal of a neural formulation of the BoF model, called Convolutional BoF (CBoF), that works as a trainable pooling layer. In this way, a unified end-to-end trainable convolutional architecture, that can be optimized using the back-propagation algorithm, is formed. It is demonstrated that the proposed method can a) reduce the number of parameters needed in a network, b) allow the network effectively handle images of any size, and c) provide better distribution-shift invariance that competitive methods, such as the Spatial Pyramid Pooling (SPP) [8]. Note that the BoF model ignores most of the spatial information, which is expressed by the position from which each feature vector was extracted, and can be used to further increase the accuracy of the models. Therefore, a spatial pyramid scheme is also proposed to retain some spatial information from the extracted feature vectors. Furthermore, note that it is straightforward to combine the proposed CBoF model with fully convolutional networks [20], simply by setting the appropriate stride/window for the CBoF pooling.

This paper is an extended version of our previous work presented in [21]. We further extend our previous work by providing a detailed derivation of a faster linear approximation of the proposed (highly non-linear) pooling technique, that can be directly used for embedded applications where devices with limited resources are used [22], [23]. The proposed method is also evaluated on a real problem, where lightweight models that can be used to aid drone-based cinematography tasks are needed. Furthermore, we present additional experiments as well as a more thorough analysis of the proposed method. To demonstrate the generality of the proposed method, the proposed pooling layer was also combined and evaluated with a deep supervised hashing technique [6]. Finally, apart from providing a practical way to train and deploy lightweight CNNs, this paper aims at linking the CNNs with the extensive BoF-based work that has been done before the deep learning era and providing new

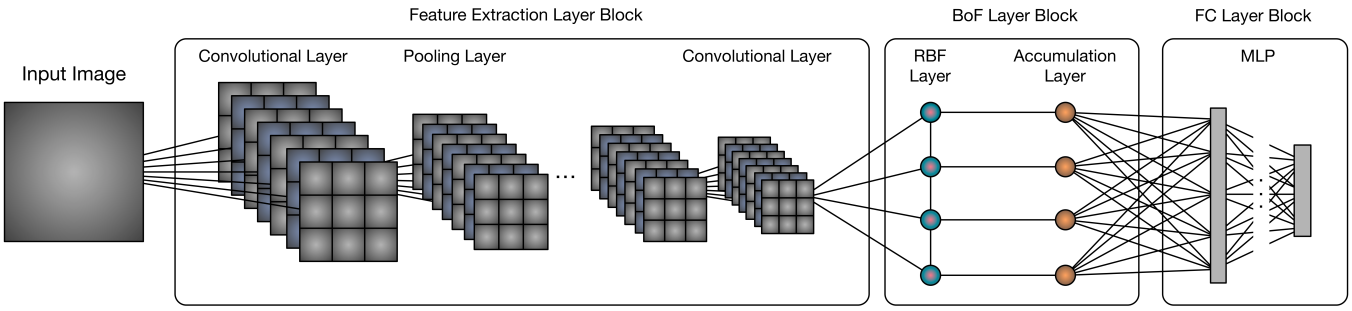

Fig. 2. Convolutional BoF Model

insight for understanding convolutional neural architectures. The proposed method can be extended beyond CNNs to any other type of deep differentiable feature extractors, e.g., recurrent models [24]. It also provides a practical link between CNNs and the existing literature on dictionary-based image representation [25]–[28]. An open-source implementation of the proposed method is publicly available at https://github.com/passalis/cbof.

The paper is structured as follows. The related work is briefly introduced in Section II. The proposed methods are presented and discussed in detail in Section III. Next, the proposed methods are evaluated and compared to other competitive methods using seven different datasets and three different evaluation setups in Section IV. Finally, conclusions are drawn in Section V.

II. Related Work

Reducing the size of CNNs and increasing the inference speed is an increasingly important issue with many recent methods proposed to tackle this problem [8], [14]–[16], [29]–[31]. Some of these works employ pruning or compression methods to reduce the size and increase the speed of CNN models [14]–[16], [31], [32]. However, these methods mainly aim to compress an already trained CNN model, instead using a smaller/faster CNN architecture at the first place. Furthermore, they inherit the limitations of the used architecture, i.e., they might not be able to work efficiently with smaller input images and adapt to the available computational resources without re-training. It is worth noting that vector quantization is also employed by some of these methods, e.g., [16]. However, in contrast to these works, that use quantization merely for reducing the size of a neural layer, the proposed CBoF method utilizes a differentiable quantization scheme that is actually part of the network architecture and allows for training the network in and end-to-end fashion. Thus, CNN models with less parameters can be directly trained (instead of compressing them after training), while providing a way for natively handling differently sized images. Furthermore, group convolutions can be also used to make networks more compact [33], while sparse representations can be also utilized to reduce the storage requirements and increase the inference speed [34]. Of course, the proposed CBoF method can be combined with these techniques, e.g., group convolutions [33], or use depth-wise separable convolutions, e.g., [30], to increase the speed of the feed-forward process and reduce the model size.

Several global pooling operators have been used to allow CNNs work with images of any size [8], [13], [35]. Spatial global pooling [8], [35], has been used to overcome the loss of spatial information that occurs when naive global pooling operators, such as Global Max Pooling (GMP) [13], are used. Both the proposed approach and the aforementioned methods allow a CNN to handle images of any size and reduce the number of required parameters. However, the proposed CBoF method significantly increases the accuracy of the networks and requires less parameters, as experimentally demonstrated in Section IV. The proposed method *learns* how to perform pooling, which allows for increasing the distribution shift invariance of the network, as well as for more efficiently compressing the extracted feature maps.

Cross Channel Parametric Pooling (CCPP) [36], was proposed as a way to recombine various channels of the extracted feature maps and reduce the complexity of the network by performing dimensionality reduction. In contrast to the CCPP technique the proposed method is capable of extracting a constant length representation and allowing the network to natively operate with arbitrary sized input images. The Vector of Locally Aggregated Descriptors (VLAD) [37], and bilinear pooling [38], can also be used to perform CNN pooling. However, for these methods the size of the extracted representation is still bound to the original dimensionality of the extracted feature vectors. On the other hand, the BoF-based pooling completely decouples the size of the learned representation from the architecture of the used convolutional feature extractor allowing for learning significantly smaller networks. Some recently proposed methods attempt to reduce the dimensionality of the representation extracted using bilinear pooling [39], [40], even though they still yield significantly larger representations than the proposed approach (one to two orders of magnitude larger than the representations used in this work).

The proposed method is also related to traditional supervised dictionary learning methods [41]–[45]. In [44], and its extension for large-scale retrieval [46], a neural formulation of the BoF model is employed to learn how to aggregate the extracted feature vectors. However, these methods are designed to work with fixed handcrafted extractors instead of trainable convolutional feature extractors. In [47], the features extracted from a CNN were combined with the BoF representation. However, in [47]

the CNN was pretrained and it was not possible to fine-tune (or train from scratch) the resulting architecture. To the best of our knowledge, this is the first method that allows for combining a BoF-based pooling layer with CNNs and allows for the *supervised* end-to-end training of the resulting unified architecture using the back-propagation algorithm leading to extracting smaller representations and deploying more lightweight networks.

III. Proposed Method

The proposed Convolutional BoF layer is used between the regular feature extraction layer block and the fully connected classification block of a CNN. Figure 2 illustrates the structure of a CNN that includes the proposed BoF pooling layer. In the next subsection, the structure of the proposed layer is described in detail and a learning algorithm is derived. Next, a linear approximation of the BoF pooling layer is derived and discussed in detail. Finally, the computational complexity of the proposed approaches is discussed and compared to other methods.

A. Convolutional BoF Pooling

Let $\mathcal{X}$ be a collection of N images. A feature map is extracted from the i-th image using the feature extraction layer block, as shown in Figure 2. Note that there is no restriction on the architecture that can be used for the feature extraction process. Let L denote the number of layers that the feature extraction block is composed of. The notation $\mathbf{x}_{ij} \in \mathbb{R}^{N_F}$ is used for the j-th feature vector extracted for the i-th image of $\mathcal{X}$, where N_F denotes the number of channels of the L-th convolutional layer (from which the features are extracted). Note that the size of the extracted feature map defines the number of the feature vectors that will be available to the BoF layer. For example, if a 20×20 feature map is extracted, then 400 feature vectors will be available for the quantization process. The notation N_i is used to refer to the number of feature vectors extracted from the i-th image.

The i-th image can be represented by a set of N_i feature vectors $\mathbf{x}_{ij} \in \mathbb{R}^{N_F}$ $(j = 1, \ldots, N_i)$, extracted using a trainable convolutional feature extractor. The BoF layer compiles a histogram vector for each image after quantizing the extracted feature vectors. Note that this in contrast with existing naive global pooling methods, that either simply fuse the feature vectors [2], [7], [9], [10], or employ pyramid-based polling to introduce spatial information into the extracted representation, like SPP [8]. It is worth noting that the proposed approach completely decouples the size of the extracted representation from both the number of the extracted feature vectors as well as from their dimensionality. This allows to independently control the size of the extracted representation allowing for reducing the number of parameters needed in the fully connected layer and allowing for handling images of any size.

Two sublayers are used to express the BoF model as a unified neural layer [44]: a layer that measures the similarity of the input feature vectors to the codewords and quantizes the input feature vectors, and an accumulation layer that compiles the final histogram representation by aggregating the quantized feature vectors. Therefore, these two layers form a unified pooling architecture that can be used to extract a compact representation, which is then fed to the employed classifier.

Any differentiable similarity function can be used to measure the similarity between the extracted feature vectors and the codewords. Following the soft BoF formulation proposed in [44], the RBF kernel is used as similarity metric. Therefore RBF neurons, one for each codeword, are used in the first sublayer. The output of the k-th RBF neuron $[\boldsymbol{\phi}(\mathbf{x})]_k$ is defined as:

$$[\boldsymbol{\phi}(\mathbf{x})]_k = \exp(-||\mathbf{x} - \mathbf{v}_k||_2/\sigma_k) \in \mathbb{R}, \tag{1}$$

where $\mathbf{x}$ is a feature vector and $\mathbf{v}_k$ is the center of the k-th RBF neuron. A scaling factor σ_k is also used to individually adjust the Gaussian function of each RBF neuron. The size of the extracted representation can be controlled by adjusting the number of RBF neurons used in the BoF layer. The notation N_K is used to refer both to the size of the extracted representation and the number of neurons used in the BoF layer.

Following the l^1 scaling, which is used in soft BoF formulations [44], [48], a normalized RBF architecture is employed. The output of the k-th RBF neuron is calculated as:

$$[\boldsymbol{\phi}(\mathbf{x})]_k = \frac{\exp(-||\mathbf{x} - \mathbf{v}_k||_2/\sigma_k)}{\sum_{m=1}^{N_K} \exp(-||\mathbf{x} - \mathbf{v}_m||_2/\sigma_m)} \in \mathbb{R}. \tag{2}$$

The normalization employed in (2) contributes to the ability of the proposed method to withstand, to some extent, mild distribution shifts, improving the scale invariance of the method. This can be better understood by observing that (2) effectively quantizes the extracted feature vectors, ensuring that a meaningful normalized membership vector, that correctly identifies the codewords with the highest similarity to the feature vectors, will be always obtained after this process. The improved scale invariance of the proposed method is experimentally demonstrated in Section IV and in the supplementary material (Fig. 3).

The final representation of each image is extracted by accumulating the responses of the RBF neurons for each feature vector that is fed to the BoF layer:

$$\mathbf{s}_i = \frac{1}{N_i} \sum_{j=1}^{N_i} \boldsymbol{\phi}(\mathbf{x}_{ij}) \in \mathbb{R}^{N_K}, \tag{3}$$

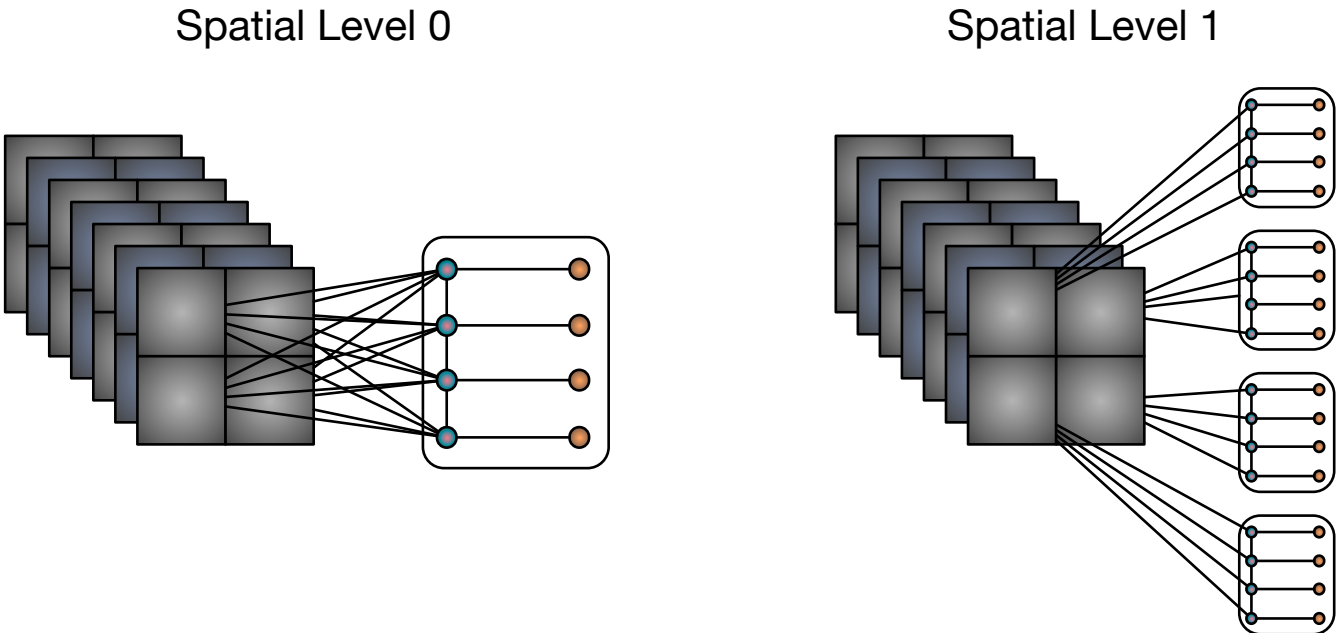

Fig. 3. A spatial pyramid scheme is used to introduce spatial information into the representation extracted using the proposed model.

where $\boldsymbol{\phi}(\mathbf{x}) = ([\boldsymbol{\phi}(\mathbf{x})]_1, \ldots, [\boldsymbol{\phi}(\mathbf{x})]_{N_K})^T \in \mathbb{R}^{N_K}$ is the output vector of the RBF layer. The histogram $\mathbf{s}_i$ defines a distribution over the RBF neurons and describes the visual content of each image. The vector $\mathbf{s}_i$, which have unit l^1 norm, is then fed to the fully connected layer.

The BoF model compiles a histogram that expresses only the distribution of the extracted feature vectors, discarding most of the spatial information (expressed by the position from which they were extracted). To this end, spatial information is re-introduced to the extracted representation using a spatial pyramid scheme, similar to the Spatial Pyramid Matching method [17]. The image is segmented into a number of regions and a separate histogram is extracted from each one. Then, the extracted histograms are fused together to form the final histogram representation, as shown in Figure 3. The size of the extracted representation is $N_K N_S$, where N_S denotes the number of spatial regions.

A classifier must be used to infer the class of an image after extracting its histogram representation $\mathbf{s}_i$. Any classifier/model with differential loss function can be used to this end. To simplify the presentation of the proposed method, it is assumed that a multilayer perceptron (MLP) with one hidden layer is used. The number of hidden neurons is denoted by N_H, while one output neuron is used for each class leading to an output layer with N_C neurons (for a classification problem with N_C different classes). When regression is performed, only one output neuron is used, i.e., $N_C = 1$. The *elu* activation function [49], with the default hyper-parameters, is used for the hidden layer. Using the *elu* activation function in the layer that receives the extracted histograms improves the convergence speed of the network over other evaluated activation functions (such as the plain *ReLU*). The softmax activation function is used for the output layer for classification, while no activation function is used for regression. For training the network, the categorical cross-entropy loss is used for classification tasks and the squared error loss function is used for regression tasks. Dropout with rate $p = 0.5$ is used for the hidden layer, except otherwise stated.

Gradient descent is employed to learn the parameters of the proposed architecture:

$$\Delta(\mathbf{W}_{MLP}, \mathbf{V}, \boldsymbol{\sigma}, \mathbf{W}_{conv}) = -(\eta_{MLP} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{MLP}}, \eta_V \frac{\partial \mathcal{L}}{\partial \mathbf{V}}, \eta_\sigma \frac{\partial \mathcal{L}}{\partial \boldsymbol{\sigma}}, \eta_{conv} \frac{\partial \mathcal{L}}{\partial \mathbf{W}_{conv}}), \tag{4}$$

where the notation $\mathcal{L}$ is used to refer to the used loss function, $\boldsymbol{\sigma}$ to the vector of scaling factors ($\boldsymbol{\sigma} = (\sigma_1, \ldots, \sigma_{N_K})$), and $\mathbf{V} = (\mathbf{v}_1, \ldots, \mathbf{v}_{N_K})$ denotes the centers of the RBF neurons. The parameters of the classification layer and the feature extraction layer are denoted by $\mathbf{W}_{MLP}$ and $\mathbf{W}_{conv}$ respectively. The learning rates for each of the parameters are denoted by η_{MLP}, η_V, η_{sigma} and η_{conv} respectively. In this work, the Adam (Adaptive Moment Estimation) method [50], is employed for performing the optimization. The derivatives of the CBoF layer used in (4) are analytically derived in the supplementary material.

The convolutional feature extractor can be either pre-trained or be randomly initialized. In the Section IV it is demonstrated that the proposed method can efficiently handle both cases. For the RBF neurons, k-means initialization is used: all the feature vectors $\mathcal{S} = \{\mathbf{x}_{ij} | i = 1, \ldots, N, j = 1, \ldots, N_i\}$ are clustered into N_K clusters and the centroids (codewords) $\mathbf{v}_k \in \mathbb{R}^{N_F} (k = 1, \ldots, N_K)$ are used to initialize the centers of the RBF neurons. It is worth noting that this process is also used by the BoF model for learning the codebook. However, the proposed method employs this step only for initializing the centers, which are

then further optimized according to the employed objective. The scaling factors for the RBF neurons are initially set to 0.1. Finally, random orthogonal initialization is used for initializing the parameters of the employed MLP [51].

B. Linear BoF Pooling Block

The key idea in the Linear BoF pooling is to replace the highly non-linear similarity calculations, that involve calculating the pairwise distances and then transforming them into similarities using an RBF kernel, as described in (1), with a faster linear operator. That is, the highly non-linear exponential operator and the scaling factor used in (1) are omitted obtaining the following (unbounded) similarity function:

$$[\boldsymbol{\phi}(\mathbf{x})]_k = -||\mathbf{x} - \mathbf{v}_k||_2 \in \mathbb{R}. \tag{5}$$

Eq. (5) can be expanded into (6) by simply using the square of the distance between the feature vectors and the codewords:

$$[\boldsymbol{\phi}(\mathbf{x})]_k = -||\mathbf{x} - \mathbf{v}_k||_2^2 = 2\mathbf{x}^T\mathbf{v}_k - ||\mathbf{x}||_2^2 - ||\mathbf{v}_k||_2^2. \tag{6}$$

Therefore, the similarity between a codeword and an input feature vector can be expressed as the inner product between these vectors after subtracting their l^2 norm. Note that (after completing the training process) the norm of each codeword is constant, i.e., $||\mathbf{v}_k||_2^2 = c_k$. Assuming that the norm of the input feature vector is also constant (this can be easily ensured using a simple normalization scheme), i.e., $||\mathbf{x}||_2^2 = c_f$, (6) is simplified into: $[\boldsymbol{\phi}(\mathbf{x})]_k = 2\mathbf{x}^T\mathbf{v}_k - c$, where $c = c_k + c_f$ is just a fixed constant for each codeword. Therefore, by omitting the additive factor c the similarity function can be expressed as:

$$[\boldsymbol{\phi}(\mathbf{x})]_k = \mathbf{x}^T\mathbf{v}_k \tag{7}$$

Eq. (7) actually expresses the cosine similarity metric ($\frac{\mathbf{x}^T\mathbf{v}_k}{||\mathbf{x}||_2||\mathbf{v}_k||_2}$) under the unit length assumption, i.e., $||\mathbf{x}||_2 = ||\mathbf{v}_k||_2 = 1$. However, when the unit length assumption does not hold, (7) ranges from $-\infty$ to ∞, while the cosine similarity always ranges from -1 to 1.

To avoid costly normalizations during the training/deployment of the network, the absolute value operator is used to ensure that (7) properly encodes a similarity metric, leading to the similarity metric used in the Linear CBoF:

$$[\tilde{\boldsymbol{\phi}}(\mathbf{x})]_k = |\mathbf{x}^T\mathbf{v}_k| \in \mathbb{R} \tag{8}$$

where $|\cdot|$ is the absolute value operator. Large similarity values indicate a high degree of correlation (either positive or negative) between the two vectors, while values close to 0 indicate orthogonality (no correlation) [22]. Note that the proposed similarity metric ignores the sign of the correlation. This is not expected to harm the accuracy of the method, since a high degree of correlation between two vectors, i.e., being collinear, conveys important information about them. Furthermore, the employed feature extractor is adapted to the used similarity metric, since the gradients that are back-propagated to the corresponding layers ignore the sign of the correlation.

Similarly to the soft quantization employed in the regular CBoF pooling (described in (2)), the membership vector for a feature vector $\mathbf{x}$ is calculated as:

$$[\tilde{\boldsymbol{\phi}}(\mathbf{x})]_k = \frac{|\mathbf{x}^T\mathbf{v}_k|}{\sum_{m=1}^{N_K}|\mathbf{x}^T\mathbf{v}_m|} \in \mathbb{R}. \tag{9}$$

Then, as in the regular CBoF pooling, the histogram representation is extracted by averaging over the extracted membership vectors (described in (3)).

Gradient descent is used for learning the parameters of the Linear CBoF layer, as before, and the corresponding derivatives are analytically derived in the supplementary material. Also, note that using k-means initialization does not improve the accuracy for the Linear CBoF pooling since it is no longer meaningful to use euclidean-based clustering when the inner product-based similarity is calculated between the feature vectors and the codewords.

The aforementioned modifications allow for readily implementing the proposed method using existing deep learning tools, since the Linear CBoF layer can be expressed as sequence of a regular convolution layer and an average pooling layer with appropriately set pooling window and stride. To understand these note the following. First, the similarity calculations described in (8) can be implemented as a regular convolutional layer with the codebook used as the convolutional filters and the absolute value operator as the activation function. Therefore, N_K filters of size $1 \times 1 \times N_F$ can be used, where N_K is the number of codewords and N_F is the number of filters in the previous convolutional layer. Furthermore, in the conducted regression experiments it was established that skipping the normalization described in (9) does not necessarily negatively impact the pooling performance. Finally, average pooling with window size and stride equal to the size of the extracted feature maps can be used to extract the histogram representation of the input image as a feature map of size $1 \times 1 \times N_K$. Implementing the proposed spatial segmentation scheme requires only to alter the used window size, e.g., reducing the window size by half is equivalent to using spatial segmentation at level 1, since four independent histograms are extracted. However, note that implementing spatial segmentation this way shares the same codebook over the spatial regions. Even though this reduces the learning capacity of the model, it can have a positive regularizing effect reducing the risk of over-fitting.

Note that the architecture used for implementing the linear variant of the method has some similarities with the Network-in-Network approach that also uses 1×1 convolutions to reduce the dimensionality of the extracted features [52]. Under this consideration, the Network-in-Network architecture (with one 1x1 layer followed by a global pooling operator) can be thought as a primitive way to extract a BoF-inspired representation, giving further insight on the success of the aforementioned technique.

C. Computational Complexity Analysis

The asymptotic cost of feed forwarding the proposed architecture, along with its storage requires are calculated in this Section. The conducted analysis focuses on the part of the network after the last convolution layer. The notation N_F is used to refer to the number of filters used in the last convolutional layer, while N_i is the number of feature vectors extracted from this layer. Also, it is assumed that an MLP with N_H hidden neurons is used. The number of used spatial regions is denoted by N_S. Finally, to simplify the conducted analysis, it assumed that $N_F < N_H$ (as in the conducted experiments) and the quantity $C_L = N_H N_C$ is defined and used to refer to the storage requirements/feed-forward cost for the layers after the first fully connected layer. The same analysis is also conducted for the Global Max Pooling (GMP) and the Spatial Pyramid Pooling (SPP) methods.

A plain CNN requires $O(N_i N_F N_H + C_L)$ parameters in the fully connected layer, the GMP pooling method requires $O(N_F N_H + C_L)$ parameters, the SPP pooling method requires $O(N_S N_F N_H + C_L)$ parameters, while the proposed CBoF and Linear CBoF methods require $O(N_S N_K N_F + N_S N_K N_H + C_L)$ parameters. If the codebook is shared between the spatial regions, then the number of parameters is reduced to $O(N_K N_F + N_S N_K N_H + C_L)$. The CBoF method is capable of decoupling the extracted representation both from the input feature dimensionality N_F and the number of extracted feature vectors N_i, allowing for using significantly smaller networks.

The cost (number of mathematical operations) of performing the feed-forward process is $O(N_i N_F N_H + C_L)$ for the CNN, $O(N_i N_F + N_F N_H + C_L)$ for the GMP method, $O(N_i N_F + N_S N_F N_H + C_L)$ for the SPP method and $O(N_i N_K N_F + N_S N_K N_H + C_L)$ for the CBoF-based methods. Note that an extra cost incurs for the feature aggregation/quantization step for both the SPP and CBoF methods ($O(N_i N_F)$ and $O(N_i N_K N_F)$ respectively). However, this cost is amortized by the smaller number of calculations required after this step (assuming that $N_K < N_H$, as in the conducted experiments). The asymptotic cost for calculating the similarity between the feature vectors and the codebook is the same for both the regular CBoF and Linear CBoF, since the distance between two vectors can be calculated using three inner product calculations, i.e., $||\mathbf{x} - \mathbf{y}||_2^2 = \mathbf{x}^T\mathbf{x} + \mathbf{y}^T\mathbf{y} - 2\mathbf{x}^T\mathbf{y}$ for any two vectors $\mathbf{x}$ and $\mathbf{y}$. Finally, note that the number of the extracted feature vectors (N_i) depends on the size of the input image. Therefore, the complexity of feed forwarding the network can be readily adjusted for the GMP/SPP and CBoF methods by using an appropriately-sized input image.

IV. Experiments

In this Section the proposed method is evaluated using seven datasets and three different evaluation setups. Due to space constraints, the datasets and the networks used for the evaluation are described in detail in the supplementary material.

A. Classification Evaluation

1) MNIST Evaluation: First, the MNIST dataset [53], was used to compare the proposed CBoF method (spatial level 1, 32 RBF neurons) to other pooling techniques. The same feature extraction block, as with the CBoF method, was used for the baseline CNN combined with a 2×2 pooling layer before the fully connected layer (1000×10). Dropout with rate $p = 0.5$ was used for both the input and the hidden classification layers of the CNN, since this further lowers the classification error. The GMP/SPP pooling layers are connected after the last convolutional layer instead of using max pooling. For the SPP technique one spatial level was used. Digit images of 20×20, 24×24, 28×28, 32×32 and 36×36 pixels were used (except of the CNN baseline model that can be trained only using one predetermined image size). Two classification error rates are reported. For the first one, the default image size is used (28×28 pixels), while the second one refers to using a reduced image size (20×20 pixels). The number of parameters in the feature extraction block is common across all the model. Therefore, the number of parameters used in the layers after the feature extraction layer are reported. The parameters of all the layers of the networks are learned during the training.

The experimental results are reported in Table I. The CBoF method performs better than both the plain CNN and the GMP/SPP methods. At the same time, the CBoF method requires about an order of magnitude less parameters (after the convolutional layers) than a CNN. Furthermore, the proposed method provides better scale-invariance than the GMP/SPP techniques, even though all the methods were trained using various image sizes. The performance of Linear CBoF was also evaluated in Table I. The Linear CBoF performs similarly to the GMP and SPP methods, while providing better scale invariance, e.g., GMP achieves 3.22% classification error for 20×20 images, while Linear CBoF (0, 32) reduces the error to just 1.43% and also requires less parameters. Note that the proposed non-linear CBoF formulation further reduces the classification error over the Linear CBoF formulation, e.g., the classification error is reduced from 0.56% to 0.51% when spatial segmentation and 64 codewords are used.

TABLE I
MNIST: COMPARING CBOF TO OTHER METHODS. THE SPATIAL LEVEL AND THE NUMBER OF RBF NEURONS ARE REPORTED IN THE PARENTHESIS FOR THE CBOF MODEL. (*TRAINING FROM SCRATCH USING 20×20 IMAGES)

Method	Cl. Error (28)	Cl. Error (20)	# Param.
CNN	0.56	(0.78)*	1,035k
GMP	0.62	3.22	75k
SPP	0.52	1.78	331k
Linear CBoF (0, 32)	0.71	1.43	44k
Linear CBoF (1, 32)	0.60	1.24	141k
Linear CBoF (1, 64)	0.56	1.52	271k
CBoF (0, 32)	0.68	1.44	44k
CBoF (1, 32)	0.55	0.94	147k
CBoF (1, 64)	**0.51**	**0.83**	284k

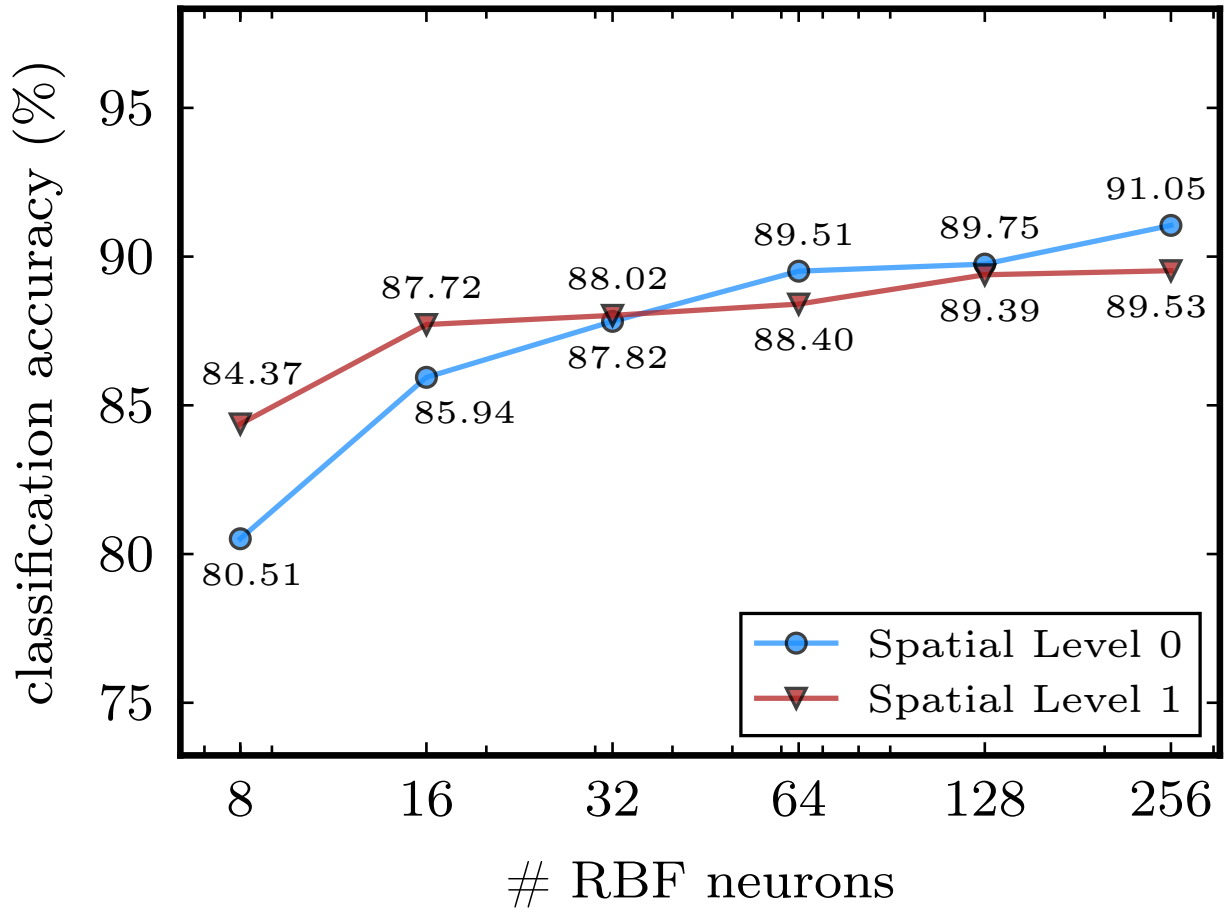

Fig. 4. 15-scene: Comparing test accuracy of CBoF for different number of codewords and spatial levels

The GMP and SPP networks were also evaluated with an added 1×1 convolutional layer with 64 filters. The proposed CBoF method still outperforms both the GMP and SPP methods, which lead to 3.52% and 1.66% classification error respectively (when 20×20 images are fed to the networks). The effect of the number of codewords, spatial levels and multi-size training are extensively evaluated and discussed in the supplementary material.

2) 15-scene Evaluation: First, the performance of the CBoF was evaluated for different codebook sizes and spatial levels using the 15-scene dataset [17]. The pretrained feature extraction layer block C, as described in the supplementary material, was used for the conducted experiments using the 15-scene dataset. The network was trained for 100 epochs. Also, the fully connected layers were pretrained for 10 epochs before training the whole network to avoid back-propagating gradients from the randomly initialized layers. The precision on the test set using different number of codewords (evaluated on one test split) are shown in Figure 4. Even when a small number of codewords is used, e.g., 32-64 codewords/RBF neurons, the network is capable of achieving remarkable high classification accuracy. Nonetheless, using more RBF neurons allows for further

TABLE II
15-SCENE: CBOF ACCURACY (%) FOR DIFFERENT IMAGE SIZES AND TRAIN SETUPS: TRAIN A (227×227 IMAGES), TRAIN B (203×203, 227×227 AND 251×251 IMAGES) AND TRAIN C (ALL THE AVAILABLE IMAGE SIZES).

Im. Size	179	203	227	251	275
Train A	88.01	90.42	90.66	90.32	88.82
Train B	89.66	91.61	92.01	91.45	90.03
Train C	**90.44**	**92.05**	**92.38**	**91.64**	**90.39**

TABLE III
15-SCENE: COMPARING CBOF TO OTHER METHODS (*TRAINING FROM SCRATCH USING 179×179 IMAGES)

Method	Accuracy (227)	Accuracy (179)	# Param.
CNN	88.79 ± 0.62	(87.90 ± 1.14)*	9,232k
GMP	86.74 ± 0.40	84.62 ± 0.32	272k
SPP	89.94 ± 0.73	87.70 ± 0.46	1,296k
Linear CBoF (0, 256)	90.10 ± 0.83	87.77 ± 0.95	337k
CBoF (0, 256)	$\mathbf{91.38 \pm 0.63}$	$\mathbf{89.86 \pm 0.80}$	337k

increasing the classification accuracy. Due to the nature of the dataset, the best classification accuracy is achieved when no spatial segmentation is used (spatial information is less important when recognizing natural scenes than when detecting the edges of aligned digits, as in the MNIST dataset). The CBoF model was also trained using images of different sizes (using the same setup as before). The experimental results, reported in Table II, indicate that multi-size training indeed improves the classification accuracy for any image size (the classification accuracy increases from 90.66% to 92.38%, when multiple image sizes are used for training).

The CBoF model using 256 RBF neurons and no spatial segmentation is also compared to other competitive techniques in Table III. All the evaluated methods share the same feature extraction layer (Feature Extraction Layer Block C). For the plain CNN, a 1000×15 fully connected layer is used after adding a max pooling layer, while the GMP/SPP layer is added after the last convolutional layer, as before. To ensure a smooth convergence during training, the learning rate for the convolutional layers was set to $\eta_{conv} = 10^{-5}$ for the CNN/GMP/SPP techniques. Note that slightly different results from [54] are reported, since the fully connected layer of the pretrained network was not used, but trained from scratch (along with the rest of the layers of the network). The proposed CBoF and Linear CBoF methods outperform both the CNN and the GMP/SPP techniques, while using significantly less parameters. The same is also true when smaller images are fed to the network. Again, the non-linear CBoF formulation outperforms the Linear CBoF method. The GMP/SPP techniques were also evaluated with one added 1×1 convolutional layer (86.32% for the GMP, 89.55% for the SPP, 256 1×1 filters were used). The proposed CBoF method still outperforms these techniques, even though one extra layer was added.

3) MIT67 Evaluation: The CBoF method was also evaluated using the MIT Indoor Scene dataset (MIT67) [55]. No spatial segmentation was used for these experiments and a BoF layer with 512 RBF neurons was used. The CNN, GMP, and SPP techniques were also evaluated using the same convolutional feature extraction layers, the feature extraction block C, as before. Three image sizes were used for the training process, i.e., 203×203, 227×227, and 251×251 pixels. The learning rate for the convolutional layers was set to $\eta_{conv} = 10^{-5}$ for the CNN/GMP/SPP techniques. Note that for the Linear CBoF the learning rate was set to $\eta_V = 0.001$. All the networks were trained for 20 epochs, while the fully connected layer was pre-trained until reaching 30% training accuracy (to avoid back-propagating gradients from a randomly initialized fully connected layer to the pretrained convolutional layers). The training set was augmented using random horizontal flips. The results for the MIT67 dataset are shown in Table IV. The Linear CBoF method performs better than the GMP and SPP methods for 227×227 images (60.42% classification accuracy), while achieving similar accuracy for 179×179 images and using overall less parameters. Again, the CBoF method outperforms all the other evaluated methods using less parameters and achieving higher classification accuracy.

TABLE IV
MIT67: COMPARING CBOF TO OTHER METHODS (*TRAINING FROM SCRATCH USING 179×179 IMAGES)

Method	Accuracy (227)	Accuracy (179)	# Param.
CNN	58.16	(54.16)*	9,284k
GMP	58.52	55.07	324k
SPP	60.42	55.55	1,348k
Linear CBoF(0, 512)	62.41	54.94	711k
CBoF(0, 512)	**64.02**	**57.58**	711k

TABLE V
AFLW: COMPARING CBOF TO OTHER METHODS

The mean absolute angular error in degrees and the amortized feed-forward time (using batch size of 32 samples) are reported. The feed-forward time was not measured for the CBoF method since the implementation was not as highly optimized as in the rest of the models.

Method	Angular Error (24)	Angular Error (32)	Angular Error (48)	**# Parameters**	**Feed-forward Time**
CNN	-	12.87	-	1,602k	- / 9.21 / - msec
GMP	12.60	12.55	14.10	131k	5.18 / 8.75 / 19.74 msec
SPP	11.92	11.78	13.09	387k	5.71 / 9.50 / 20.52 msec
CBoF (0, 64)	12.74	12.60	14.13	70k	N/A
CBoF (1, 16)	**10.14**	**9.47**	**10.34**	70k	N/A
Linear CBoF (0, 64)	12.88	12.36	13.06	70k	5.40 / 9.24 / 20.59 msec
Linear CBoF (1, 16)	**10.84**	**10.20**	**10.70**	67k	5.49 / 8.98 / 19.90 msec
Linear CBoF (1, 64)	11.16	10.31	10.79	70k	5.23 / 9.10 / 20.39 msec

B. Regression Evaluation

The Annotated Facial Landmarks in the Wild (AFLW) dataset [56] was also used to perform the regression evaluation. The proposed model was trained to regress the horizontal facial pose (yaw). The motivation behind this experiment was to evaluate the proposed method on a real problem where lightweight models, that can be deployed on embedded systems with limited memory and computing resources, are needed [22]. The Feature Extraction Layer Block B was employed as feature extractor, while an MLP with 1000 hidden units was employed to regress the yaw. Please refer to the supplementary material for more details regarding the used feature extraction architecture. Three image sizes, i.e., 32×32, 64×64, and 96×96 pixels were used during the training, while batches from each image size were fed to each network. For the evaluated models 50,000 optimization iterations were performed. The mean squared error was used as the objective function to train the networks to regress the horizontal facial pose. The Linear CBoF pooling was also evaluated to examine its behavior in this challenging real-world problem.

The results are reported in Table V. The mean absolute angular error in degrees and the amortized mean feed-forward time using a batch size of 32 samples are reported. For the regular CBoF method the feed-forward time is not reported, since the used implementation is not as highly optimized as the rest. The proposed regular CBoF method achieves the lowest angular error using spatial segmentation level 1 (9.47 angular error vs. 11.78 angular error for the next best performing competitive method), while significantly reducing the number of used parameters. Even though the CBoF (0, 64) and CBoF (1, 16) have the same number of parameters, the spatial scheme used in CBoF (1, 16) increases the accuracy, since the spatial information seems to be especially important for the task of facial pose estimation. The Linear CBoF achieves slightly higher error than the regular CBoF, while still outperforming the other methods (CNN, GMP, and SPP). Note that the codebook is shared over the spatial regions in the Linear CBoF reducing the number of used parameters even more.

Regarding the feed-forward time the differences are quite small (usually less than 5% between different models) due to the massive parallelism that modern GPUs provide (a mid range GPU capable of 2.2 TFLOPs was used). However, the results clearly demonstrate the ability of both the proposed Linear CBoF and the GMP/SPP techniques to adapt to the available computing power (by reducing the input image size) and handle images of various sizes. Nonetheless, it should be noted again that the proposed method reduces both the required memory as well as the pose estimation error over the GMP and SPP methods.

C. Retrieval Evaluation

For the retrieval evaluation the proposed method was combined with a recently proposed deep supervised hashing technique [6]. For the experiments conducted in this Subsection a fully connected layer with k linear units is used after the last pooling layer, where k is the length of the learned hash code in bits. The networks were trained using the loss function proposed in [6]. Following [6], the regularizer weight was set to $\alpha = 0.01$ and the margin to $m = 2k$. To obtain the representation (binary code) of each image in the hamming space the output of the network was passed through the $sign(\cdot)$ function. Since most of the activations are expected to lie outside the $(-1, 1)$ interval (due to the used regularizer), this is a simple and effective way to get the binary codes for the images.

First, the INRIA Holidays (Holidays) dataset [57], was used to evaluate the proposed method. During the training 1,500,000 randomly chosen (similar or dissimilar) image pairs were fed to the network. A pretrained residual network (ResNet-50, trained on the Imagenet dataset [7]) was used for the feature extraction block, and the Adam algorithm was used for training the layers that were added after the pretrained feature extraction layers (no gradients were back-propagated to the feature extraction layers for this setup). The learning rate was set to $\eta = 10^{-4}$. The results for different code lengths are shown in Table VI. The mean average precision (mAP) is reported [58]. The "DSH" method refers to directly using the hashing layer after the last global pooling layer of the ResNet50. For the "CBoF (256) + DSH" method the global pooling layer of the ResNet-50 is replaced by

TABLE VI
HOLIDAYS: RETRIEVAL EVALUATION USING DEEP SUPERVISED HASHING (THE MEAN AVERAGE PRECISION (MAP) IS REPORTED)

Method	$k = 12$	$k = 24$	$k = 32$	$k = 48$
DSH	0.313	0.539	0.629	0.720
Linear CBoF(256) + DSH	**0.428**	**0.644**	**0.681**	**0.730**

TABLE VII
CIFAR10: RETRIEVAL EVALUATION USING DEEP SUPERVISED HASHING (THE MEAN AVERAGE PRECISION (MAP) IS REPORTED)

Method	$k = 12$	$k = 24$	$k = 32$	$k = 48$
CNNH [63]	0.543	0.560	0.564	0.557
DLBHC [64]	0.550	0.580	0.578	0.589
DNNH [65]	0.571	0.588	0.590	0.590
DSH [6]	0.616	0.651	0.661	0.676
DSH	0.703	0.800	0.820	0.841
Linear CBoF(256) + DSH	**0.874**	**0.883**	**0.887**	**0.890**

the Linear CBoF method with 256 codewords. Combining the proposed pooling layer (that performs an intrinsic quantization of the extracted feature vectors) with the deep supervised hashing technique further improves the retrieval precision.

The proposed technique is also evaluated on the CIFAR10 dataset [59]. The same setup as before is used, however instead of using the ResNet-50 for the feature extraction, another pretrained residual network, the Wide ResNet (WRN-40-4) [11], is used (the network was pretrained to perform classification on the CIFAR10 dataset and its last fully connected layer is not used). The optimization ran for 10 epochs. The pairwise similarities between the samples of each batch (the batch size was set to 64) were fed to the network during the training. The experimental results are reported in Table VII. Using a Wide Residual Network instead of a more shallow network significantly boosts the mean average precision over the other techniques. Again, combining the Linear CBoF method with the DSH further increases the retrieval precision.

The proposed method was also evaluated on the NUS-WIDE dataset [60], following the same setup as in the previous experiments and using a state-of-the-art DenseNet201 as the employed feature extractor [61]. The DenseNet201 was pretrained on the Imagenet dataset and the same evaluation setup as in [62] was used for the hashing evaluation. The optimization ran for 10 epochs using batches of 128 samples, an MLP with 2048 hidden ReLU neurons was used to extract the hashing codes and the learning rate was set to 0.001. The evaluation results are reported in Table VIII. As before, combining the proposed Linear BoF method with the DSH method significantly increases the retrieval precision over directly using the global average pooling layer employed by the DenseNet201.

V. CONCLUSIONS

In this paper, the BoF model was formulated as a neural pooling layer that can be combined with convolutional layers to form a powerful convolutional architecture that is end-to-end trainable using the regular back-propagation algorithm. The ability of the proposed method to reduce the size of CNNs and increase the performance over other competitive techniques was experimentally demonstrated using seven image datasets and three different settings (classification, regression and retrieval). The proposed method provides a lightweight CNN architecture that can readily adapt to the available computational resources and it is invariant to mild distribution shifts. Furthermore, the proposed method can be combined with CNN compression techniques, such as [14]–[16], or use depth-wise separable convolutions in the feature extraction layers, e.g., [30], to further improve the speed and the size of the networks. Note that the proposed method is capable of reducing the parameters only in the first fully connected layer of a deep CNNs, which is usually the layer that uses the most parameters in a CNN. However, the proposed method can be used to extract compact layer-wise representations that can be combined with methods that employ features extracted from multiple layers, such as DenseNets [61]. This will allow for reducing the number of parameters needed in the various layers of the network, further improving the performance of these methods.

ACKNOWLEDGMENT

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 731667 (MULTIDRONE). This publication reflects the authors' views only. The European Commission is not responsible for any use that may be made of the information it contains.

REFERENCES

[1] X. Chen, J. Weng, W. Lu, J. Xu, and J. Weng, "Deep manifold learning combined with convolutional neural networks for action recognition," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–15, 2017.

[2] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

TABLE VIII
NUS-WIDE: RETRIEVAL EVALUATION USING DEEP SUPERVISED HASHING (THE MEAN AVERAGE PRECISION (MAP) IS REPORTED)

Method	$k = 12$	$k = 24$	$k = 32$	$k = 48$
DSH	0.653	0.684	0.683	0.702
Linear CBoF(256) + DSH	**0.721**	**0.726**	**0.725**	**0.734**

[3] W. Shi, Y. Gong, X. Tao, and N. Zheng, "Training DCNN by combining max-margin, max-correlation objectives, and correntropy loss for multilabel image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–13, 2017.

[4] W. Shi, Y. Gong, X. Tao, J. Wang, and N. Zheng, "Improving cnn performance accuracies with min-max objective," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–14, 2017.

[5] H.-F. Yang, K. Lin, and C.-S. Chen, "Supervised learning of semantics-preserving hash via deep convolutional neural networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[6] H. Liu, R. Wang, S. Shan, and X. Chen, "Deep supervised hashing for fast image retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 2064–2072.

[7] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arXiv preprint arXiv:1512.03385*, 2015.

[8] ——, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

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

[10] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[11] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[12] S. S. Haykin, S. S. Haykin, S. S. Haykin, and S. S. Haykin, *Neural networks and learning machines*. Pearson Upper Saddle River, NJ, USA:, 2009, vol. 3.

[13] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1717–1724.

[14] M. Denil, B. Shakibi, L. Dinh, N. de Freitas *et al.*, "Predicting parameters in deep learning," in *Proceedings of the Advances in Neural Information Processing Systems*, 2013, pp. 2148–2156.

[15] S. Han, H. Mao, and W. J. Dally, "A deep neural network compression pipeline: Pruning, quantization, huffman encoding," *arXiv preprint arXiv:1510.00149*, 2015.

[16] J. Wu, C. Leng, Y. Wang, Q. Hu, and J. Cheng, "Quantized convolutional neural networks for mobile devices," *arXiv preprint arXiv:1512.06473*, 2015.

[17] S. Lazebnik, C. Schmid, and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, vol. 2, 2006, pp. 2169–2178.

[18] J. Sivic and A. Zisserman, "Video google: A text retrieval approach to object matching in videos," in *Proceedings of the IEEE International Conference on Computer Vision*, 2003, pp. 1470–1477.

[19] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 2, 1999, pp. 1150–1157.

[20] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 4, pp. 640–651, 2017.

[21] N. Passalis and A. Tefas, "Learning bag-of-features pooling for deep convolutional neural networks," in *Proceedings of the International Conference on Computer Vision*, 2017.

[22] ——, "Concept detection and face pose estimation using lightweight convolutional neural networks for steering drone video shooting," in *Proceedings of the 25th European Signal Processing Conference*, Aug 2017, pp. 71–75.

[23] I. Mademlis, V. Mygdalis, N. Nikolaidis, and I. Pitas, "Challenges in autonomous uav cinematography: An overview," in *Proceedings of the IEEE International Conference on Multimedia and Expo*, 2018.

[24] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "Lstm: A search space odyssey," *IEEE Transactions on Neural Networks and Learning Systems*, 2016.

[25] J. Mairal, J. Ponce, G. Sapiro, A. Zisserman, and F. R. Bach, "Supervised dictionary learning," in *Proceedings of the Advances in Neural Information Processing Systems*, 2009, pp. 1033–1040.

[26] Z. Li, Z. Lai, Y. Xu, J. Yang, and D. Zhang, "A locality-constrained and label embedding dictionary learning algorithm for image classification," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 2, pp. 278–293, Feb 2017.

[27] Z. Zhang, W. Jiang, J. Qin, L. Zhang, F. Li, M. Zhang, and S. Yan, "Jointly learning structured analysis discriminative dictionary and analysis multiclass classifier," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–17, 2017.

[28] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Learning stable multilevel dictionaries for sparse representations," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 26, no. 9, pp. 1913–1926, Sept 2015.

[29] Y. Gong, L. Liu, M. Yang, and L. Bourdev, "Compressing deep convolutional networks using vector quantization," *arXiv preprint arXiv:1412.6115*, 2014.

[30] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," *arXiv preprint arXiv:1704.04861*, 2017.

[31] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized cnn: A unified approach to accelerate and compress convolutional networks," *IEEE Transactions on Neural Networks and Learning Systems*, vol. PP, no. 99, pp. 1–14, 2017.

[32] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen, "Compressing convolutional neural networks," *arXiv preprint arXiv:1506.04449*, 2015.

[33] T. Zhang, G.-J. Qi, B. Xiao, and J. Wang, "Interleaved group convolutions," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[34] T. Zhang, G.-J. Qi, J. Tang, and J. Wang, "Sparse composite quantization," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4548–4556.

[35] M. Malinowski and M. Fritz, "Learnable pooling regions for image classification," in *International Conference on Learning Representations Workshop*, 2013.

[36] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proceedings of the International Conference on Learning Representations*, 2014.

[37] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "Netvlad: Cnn architecture for weakly supervised place recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 5297–5307.

[38] T.-Y. Lin, A. RoyChowdhury, and S. Maji, "Bilinear CNN models for fine-grained visual recognition," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 1449–1457.

[39] A. Fukui, D. H. Park, D. Yang, A. Rohrbach, T. Darrell, and M. Rohrbach, "Multimodal compact bilinear pooling for visual question answering and visual grounding," in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 457–468.

[40] Y. Gao, O. Beijbom, N. Zhang, and T. Darrell, "Compact bilinear pooling," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016, pp. 317–326.

[41] S. Lazebnik and M. Raginsky, "Supervised learning of quantizer codebooks by information loss minimization," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 7, pp. 1294–1309, 2009.

[42] X.-C. Lian, Z. Li, B.-L. Lu, and L. Zhang, "Max-margin dictionary learning for multiclass image categorization," in *Proceedings of the European Conference on Computer Vision*, 2010, pp. 157–170.

[43] H. Lobel, R. Vidal, D. Mery, and A. Soto, "Joint dictionary and classifier learning for categorization of images using a max-margin framework," in *Image and Video Technology*, 2014, pp. 87–98.

[44] N. Passalis and A. Tefas, "Neural bag-of-features learning," *Pattern Recognition*, vol. 64, pp. 277 – 294, 2017.

[45] A. Richard and J. Gall, "A bag-of-words equivalent recurrent neural network for action recognition," *Computer Vision and Image Understanding*, vol. 156, pp. 79–91, 2017.

[46] N. Passalis and A. Tefas, "Learning neural bag-of-features for large-scale image retrieval," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 47, no. 10, pp. 2641–2652, 2017.

[47] E. Mohedano, K. McGuinness, N. E. O'Connor, A. Salvador, F. Marqués, and X. Giró-i Nieto, "Bags of local convolutional features for scalable instance search," in *Proceedings of the ACM on International Conference on Multimedia Retrieval*, 2016, pp. 327–331.

[48] J. C. Van Gemert, C. J. Veenman, A. W. Smeulders, and J.-M. Geusebroek, "Visual word ambiguity," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 32, no. 7, pp. 1271–1283, 2010.

[49] D.-A. Clevert, T. Unterthiner, and S. Hochreiter, "Fast and accurate deep network learning by exponential linear units (elus)," *arXiv preprint arXiv:1511.07289*, 2015.

[50] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[51] A. M. Saxe, J. L. Mcclelland, and S. Ganguli, "Exact solutions to the nonlinear dynamics of learning in deep linear neural network," in *Proceedings of the International Conference on Learning Representations*, 2014.

[52] M. Lin, Q. Chen, and S. Yan, "Network in network," in *Proceedings of the International Conference on Learning Representations*, 2014.

[53] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[54] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Proceedings of the Advances in Neural Information Processing Systems*, 2014, pp. 487–495.

[55] A. Quattoni and A. Torralba, "Recognizing indoor scenes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 413–420.

[56] M. Koestinger, P. Wohlhart, P. M. Roth, and H. Bischof, "Annotated facial landmarks in the wild: A large-scale, real-world database for facial landmark localization," in *IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[57] H. Jegou, M. Douze, and C. Schmid, "Hamming embedding and weak geometric consistency for large scale image search," *Proceedings of the European Conference on Computer Vision*, pp. 304–317, 2008.

[58] C. D. Manning, P. Raghavan, H. Schütze *et al.*, *Introduction to information retrieval*. Cambridge university press, 2008.

[59] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009.

[60] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y.-T. Zheng, "Nus-wide: A real-world web image database from national university of singapore," in *Proceedings of the ACM Conference on Image and Video Retrieval*, 2009.

[61] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.

[62] W.-J. Li, S. Wang, and W.-C. Kang, "Feature learning based deep supervised hashing with pairwise labels," in *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016, pp. 1711–1717.

[63] K. Lin, H.-F. Yang, J.-H. Hsiao, and C.-S. Chen, "Deep learning of binary hash codes for fast image retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 27–35.

[64] H. Lai, Y. Pan, Y. Liu, and S. Yan, "Simultaneous feature learning and hash coding with deep neural networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3270–3278.

[65] J. Wang, T. Zhang, J. Song, N. Sebe, and H. T. Shen, "A survey on learning to hash," *arXiv preprint arXiv:1606.00185*, 2016.

Nikolaos Passalis obtained his B.Sc. in informatics in 2013 and his M.Sc. in information systems in 2015 from Aristotle University of Thessaloniki, Greece. He is currently pursuing his Ph.D. studies in the Artificial Intelligence & Information Analysis Laboratory in the Department of Informatics at the University of Thessaloniki. His research interests include machine learning, computational intelligence and information retrieval.

Anastatios Tefas received the B.Sc. in informatics in 1997 and the Ph.D. degree in informatics in 2002, both from the Aristotle University of Thessaloniki, Greece. Since 2017 he has been an Associate Professor at the Department of Informatics, Aristotle University of Thessaloniki. From 2008 to 2017, he was a Lecturer, Assistant Professor at the same University. From 2006 to 2008, he was an Assistant Professor at the Department of Information Management, Technological Institute of Kavala. From 2003 to 2004, he was a temporary lecturer in the Department of Informatics, University of Thessaloniki. From 1997 to 2002, he was a researcher and teaching assistant in the Department of Informatics, University of Thessaloniki. Dr. Tefas participated in 12 research projects financed by national and European funds. He has co-authored 87 journal papers, 194 papers in international conferences and contributed 8 chapters to edited books in his area of expertise. Over 3730 citations have been recorded to his publications and his H-index is 32 according to Google scholar. His current research interests include computational intelligence, deep learning, pattern recognition, statistical machine learning, digital signal and image analysis and retrieval and computer vision.