Learning Bag-of-Features Pooling for Deep Convolutional Neural Networks

Nikolaos Passalis and Anastasios Tefas
Aristotle University of Thessaloniki
Thessaloniki, Greece
passalis@csd.auth.gr, tefas@aiia.csd.auth.gr

Abstract

Convolutional Neural Networks (CNNs) are well established models capable of achieving state-of-the-art classification accuracy for various computer vision tasks. However, they are becoming increasingly larger, using millions of parameters, while they are restricted to handling images of fixed size. In this paper, a quantization-based approach, inspired from the well-known Bag-of-Features model, is proposed to overcome these limitations. The proposed approach, called Convolutional BoF (CBoF), uses RBF neurons to quantize the information extracted from the convolutional layers and it is able to natively classify images of various sizes as well as to significantly reduce the number of parameters in the network. In contrast to other global pooling operators and CNN compression techniques the proposed method utilizes a trainable pooling layer that it is end-to-end differentiable, allowing the network to be trained using regular back-propagation and to achieve greater distribution shift invariance than competitive methods. The ability of the proposed method to reduce the parameters of the network and increase the classification accuracy over other state-of-the-art techniques is demonstrated using three image datasets.

1. Introduction

Deep Convolutional Neural Networks (CNNs) are powerful well-known models used for various computer vision tasks, ranging from image classification [24], to visual question answering [18]. Their great success in difficult large-scale recognition problems, such as classifying the images of the ImageNet dataset [24], established them as the model of choice for the aforementioned tasks, displacing other models that were previously used, such as the Bag-of-Features (BoF) model, also known as the Bag-of-Visual-Words (BoVW) model [12].

CNNs are composed of a feature extraction layer followed by a fully connected layer that acts as a classifier. The feature extraction layer is further composed of a series of convolutional layers and pooling layers. Many different combinations of convolution and pooling operators have been proposed [8, 7, 13, 26]. The feature maps extracted from the last convolutional/pooling layer are flatten into a vector which is subsequently fed to the fully connected layer. Regardless the details of the used architecture, all the CNNs share the same fundamental principle: the parameters of the feature extraction layer are not fixed, but they are learned using back-propagation, just as any other parameter of the network.

Despite their great success, most of the proposed CNN formulations are unable to handle arbitrary sized images, as they operate on a fixed input size, restricting the CNN into accepting images of a specific size and leading to a constant cost for feed-forwarding the network (the number of floating point operations (FLOPs) needed to feed-forward the network depends on the size of the input image). This is because the dimensionality of the (flattened) output of the feature extraction layer depends on the size of the input image. Therefore, an already trained CNN cannot be directly deployed in a lower computational resource scenario, *i.e*., an embedded real-time system on a drone or a mobile device, or adapt to the available computational resources, without completely replacing the fully connected layer and retraining it. Furthermore, flattening the extracted feature maps leads to tremendously large fully connected layers. For example, for the VGG-16 network [26], 90% of the network's parameters are needed for just three fully connected layers, while only 10% of the parameters are actually spent on its 13 convolutional layers used for the feature extraction. The reason for this is the large number of the extracted feature maps (512 feature maps of size 7x7).

There were some attempts to overcome the aforementioned limitations by using global pooling operators that ensure that the output of the feature extraction layer has fixed length [7, 21]. Even though these global pooling operators can be used to provide scale invariance and allow the network to operate with arbitrary sized images, it was experimentally established (Section 4) that the accuracy of the network is reduced when the input image size is altered.

This is because the scale invariance is not provided by learning a scale-invariant pooling layer, but by learning scale-invariant filters that are then pooled together. However, as it is demonstrated in this work, learning both the pooling layer and the convolutional layers can provide significantly better scale-invariance, while reducing the size of the resulting network. Note that compression techniques, such as [5, 6, 29], can be also used to reduce the size of CNNs, however they are not capable of dealing with arbitrary sized images.

In this work, a BoF-inspired layer, that acts as a trainable quantization-based pooling layer, is used between the feature extraction layer and the fully connected layer to resolve these problems. It worths mentioning that the BoF model was originally designed to tackle similar problems, i.e., to deal with a variable number of feature vectors and provide scale and position invariance, that arose when handcrafted feature extractors were used. To understand this, consider the pipeline of the BoF model: First, a number of feature vectors are extracted from an image using a handcrafted feature extractor, such as SIFT [17], HoG [4], or LBP [20]. The number of feature vectors might vary according to the type of feature extraction and the size of each image. Then, these feature vectors are quantized into a predefined number of bins, called codewords. Finally, a constant length histogram representation is extracted for each image by counting the number of feature vectors that were quantized into each bin.

The feature maps extracted from the last convolutional layer of a CNN can be converted to feature vectors as follows. Each convolutional filter contributes to a specific position of the feature vectors, i.e., the dimensionality of the extracted feature vectors equals to the number of the used convolutional filters, while the size of the feature maps defines the number of the extracted feature vectors. Instead of directly feeding the extracted vectors into the fully connected layer, a BoF-based pooling layer is used between the last convolutional layer and the fully connected layer. This can reduce the size of the fully connected layer and allows to operate the CNN using images of any size. However, combining the BoF model with CNNs is not a straightforward task. The representation learned by the convolutional layers of a CNN is constantly altering during the training prohibiting standard BoF quantization techniques, such as k-means quantization, to be used effectively. Therefore, the BoF model must be appropriately modified in order to efficiently handle trainable convolutional feature extractors.

The contributions of this work are briefly summarized bellow. First, a neural generalization of the BoF model, composed of RBF neurons, is proposed as a generic pooling layer that can fit between the feature extraction layer and the fully connected layer of a CNN. Note that the proposed pooling layer can be also used at various depths of the network to further increase the scale-invariace and bring more fine-grained information to the fully connected layer. To the best of our knowledge, this is the first time that the BoF model is combined with convolutional neural networks into one unified architecture, called Convolutional BoF (CBoF), that is end-to-end differentiable and allows from training the resulting network using regular back-propagation. It is shown, using extensive experiments on three datasets, that CBoF provides much better invariance to distribution shifts, caused by feeding images of different sizes to the network than competitive global pooling strategies, such as Spatial Pyramid Pooling [7]. Also it can greatly reduce the number of parameter of the network over the competitive methods. This allows the proposed technique to be easily used with embedded devices with limited processing power and memory, e.g., GPU-based processing units for drones and robots. Finally, a spatial segmentation scheme is also proposed and combined with the CBoF to retain the spatial information that is carried by the extracted feature vectors. A reference implementation of the proposed method is available at `https://github.com/passalis/cbof`.

The rest of the paper is structured as follows. In Section 2 the related work is introduced and compared to the proposed CBoF model. The proposed method is presented in detail in Section 3. Next, the CBoF is evaluated and compared to other state-of-the-art techniques using three different datasets (Section 4), including a large-scale face image dataset. Finally, conclusions are drawn in Section 5.

2. Related Work

The problem of dealing with large CNNs is well recognized in the literature, with several recent works trying to reduce the model size [5, 6, 7, 29]. Most of these works use compression and pruning techniques to reduce the size of CNN models [1, 5, 6, 29]. These approaches focus on compressing an already trained CNN, instead of training a CNN with less parameters in the first place. Also they are unable to natively handle images of various sizes, leading to a constant cost for feed-forwarding the convolutional layers (since the image must be resized to a predefined size). Note that vector quantization is used both by some of these works [5, 29], and the proposed CBoF technique. However, the CBoF method uses a differentiable quantization scheme that allows training both the quantizer and the rest of the network simultaneously, instead of using fixed quantization just to reduce the size of the model. This allows for directly training CNN models with less parameters (instead of compressing them after training) as well as natively handling differently sized images. Note that the proposed CBoF model can be readily combined with any CNN compression technique to further decrease the size of the model.

The problem of dealing with arbitrary sized images is addressed in the literature using global pooling operators [7, 21, 19]. Since the naive global max pooling [21], leads

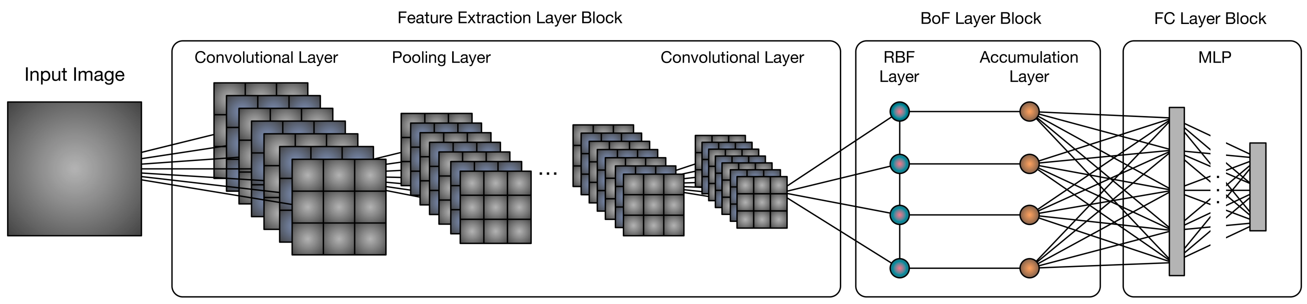

Figure 1. Convolutional BoF Model

to the loss of valuable spatial information, spatial pooling techniques [19, 7], were proposed to overcome this issue. Both the global pooling techniques and the proposed CBoF technique reduce the size of network and they can handle images of different sizes. However, as we show in Section 4, the proposed method greatly outperforms all the other global pooling methods, achieving both higher recognition accuracy and using overall less parameters. This can be attributed to the ability of the proposed method to *learn* how to perform quantization-based pooling, that allows for efficiently compressing the output of the feature extraction layer and providing much better invariance to distribution shifts caused by feeding differently sized images to the network.

Finally, the proposed method is related to supervised dictionary learning approaches for the BoF representation. A very rich literature exists for BoF supervised dictionary learning [11, 15, 16, 23]. However, all these methods are designed to work with handcrafted feature extractors instead of trainable convolutional layers. To the best of our knowledge, this is the first work that combines the BoF model with convolutional layers into one unified architecture that allows for training the resulting network from scratch by back-propagating the error from the fully connected layer to the convolutional feature extractor.

3. Proposed Method

The proposed Convolutional BoF model is composed of three layer blocks: a) a feature extraction layer block (composed of many convolutional and pooling layers), b) a BoF pooling layer block and c) a fully connected layer block. The structure of the CBoF model is illustrated in Figure 1. Each of the layers used in the CBoF is described in detail in the next subsections. Next, a learning algorithm for the CBoF model is derived and its computational complexity is discussed and compared to other methods.

3.1. Feature Extraction Layer Block

Let $\mathcal{X}$ be a set of N images to be classified using the CBoF model. The i-th image is fed to the feature extraction layer, which is composed of a sequence of convolutional layers and subsampling (pooling) layers. Any CNN architecture can be used as feature extractor, such as the LeNet [13], the VGG [26], or the ResNet [8], after removing its fully connected layers.

The last convolutional layer, denoted by L, is used to extract feature vectors that are subsequently fed to the BoF layer. The j-th feature vector of the i-th image is denoted by $\mathbf{x}_{ij} \in \mathbb{R}^D$, where D is the number of filters used in the last convolutional layer. The number of the extracted feature vectors depends on the size of the feature map and the used filter size, as described in the Introduction. For example, for an image of 28x28 pixels and two convolutional layers with filter size 5x5 a total of $20 \times 20 = 400$ feature vectors can be extracted from the final convolutional layer (assuming that the filters fully overlap with the image). To simplify the presentation of the proposed method, the number of feature vectors extracted from the last convolutional layer for the i-th image is denoted by N_i.

3.2. BoF Layer Block

After the feature extraction the i-th image is represented by a set of N_i feature vectors: $\mathbf{x}_{ij} \in \mathbb{R}^D$ $(j = 1...N_i)$. Instead of simply fusing the extracted feature vectors, like the CNNs [8, 13, 24, 26], or using a spatial pooling approach, like the SPP [7], the BoF layer compiles a fixed-length histogram for each image by quantizing its feature vectors into a predefined number of histogram bins/codewords. Note that the length of the extracted histogram vector does not depend on the number of available feature vectors, which allows the network to handle images of arbitrary size without any modification.

The BoF model is formulated as a generic neural layer that is composed of two sublayers: an RBF layer that measures the similarity of the input features to the RBF centers and an accumulation layer that builds the histogram of the quantized feature vectors. The proposed layer can be thought as a unified processing layer that feeds the extracted representation to a subsequent classifier. The output of the k-th RBF neuron $[\boldsymbol{\phi}(\mathbf{x})]_k$ is defined as:

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

where $\mathbf{x}$ is a feature vector and $\mathbf{v}_k$ is the center of the k-th RBF neuron. The RBF neurons behave somewhat like the codewords in the BoF model, *i.e.*, they are used to measure the similarity of the input vectors to a set of predefined vectors and quantize the feature space. Each RBF neuron is also equipped with a scaling factor σ_k that adjusts the width of its Gaussian function. That allows for better modeling of the input distribution, since the distribution modeled by each RBF can be independently learned. The number of RBF neurons used is denoted by N_K. The size of the extracted representation can be adjusted by using a different number of RBF neurons (N_K) in the BoF layer.

To ensure that the output of each RBF neuron is bounded, a normalized RBF architecture is used. This normalization is equivalent to the l^1 scaling that is utilized in the BoF model that uses soft-assignments [28]. Thus, the output of the RBF neurons is re-defined as:

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

The output of the RBF neurons is accumulated in the next layer, compiling the final representation of each image:

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

where $\boldsymbol{\phi}(\mathbf{x}, t) = ([\boldsymbol{\phi}(\mathbf{x}, t)]_1, ..., [\boldsymbol{\phi}(\mathbf{x}, t)]_{N_K})^T \in \mathbb{R}^{N_K}$ is the output vector of the RBF layer. Note that each $\mathbf{s}_i$ has unit l^1 norm and defines a histogram distribution over the RBF neurons that describes the visual content of each image. The vector $\mathbf{s}_i$ can be then used for the subsequent classification or retrieval tasks [22]. Hard quantization is also supported by the proposed formulation. In this case, only the RBF neuron with the maximum response is activated for each feature vector and the rest of the architecture remains unchanged. However, this modification introduces non-continuities that make the optimization of the input layer intractable and it is not used in this work.

Note that the previous process discards most of the spatial information carried by the feature vectors, since all the feature vectors are described by the same histogram regardless the part of the image from which they were extracted. To overcome this, a spatial segmentation scheme, similar to the Spatial Pyramid Matching scheme [12], is proposed. Before quantizing the feature vectors using the BoF layer, each feature vector is assigned into one spatial region and a separate BoF layer is used to quantize the feature vectors that belong to each region. That way, a different histogram is extracted from each region, allowing to introduce more spatial information into the CBoF model. This process is illustrated in Figure 2. The number of used spatial regions is denoted by N_S.

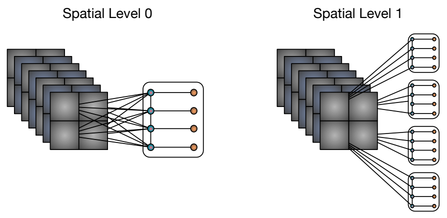

Figure 2. Using a spatial segmentation scheme to introduce spatial information into the extracted CBoF representation.

3.3. Fully Connected Layer

The previous layer receives the feature vectors of an image and compiles its histogram representation. Then, this histogram must be fed to a classifier that decides the class of the image. In this work a multilayer perceptron (MLP) with one hidden layer is used for this purpose, although any other classifier with differentiable loss function can be used.

Only the single-label classification problem is considered: the i-th training image is annotated by a label t_i and there are N_C different labels. This is without loss of generality since the proposed method can be readily applied to multi-label classification or regression problems as well.

Let $\mathbf{W}_H \in \mathbb{R}^{N_H \times (N_K N_S)}$ be the hidden layer weights and $\mathbf{W}_O \in \mathbb{R}^{N_C \times N_H}$ be the output layer weights, where N_H is the number of hidden neurons. Then, the hidden layer activations for the input histogram $\mathbf{s}_i$ of the i-th image are computed as:

$$\mathbf{h}_i = \phi^{(elu)}(\mathbf{W}_H \mathbf{s}_i + \mathbf{b}_H) \in \mathbb{R}^{N_H} \tag{4}$$

where $\phi^{(elu)}(x)$ is the *elu* activation function [3]: $\phi^{(elu)}(x) = x$ if $x > 0$, or $\phi^{(elu)}(x) = \alpha_{elu}(exp(x) - 1)$ otherwise. The parameter α_{elu} is typically set to 1. The activation function is applied element-wise and $\mathbf{b}_H \in \mathbb{R}^{N_H}$ is the hidden layer bias vector.

Similarly, the output of the MLP is calculated as:

$$\mathbf{y}_i = \phi^{(softmax)}(\mathbf{W}_O \mathbf{h}_i + \mathbf{b}_O) \in \mathbb{R}^{N_C} \tag{5}$$

where each output neuron corresponds to a label (the one-vs-all strategy is used), $\mathbf{b}_O \in \mathbb{R}^{N_C}$ is the output layer bias vector and $\phi^{(softmax)}$ is the softmax activation function.

In this work, the categorical cross entropy loss is used for training the network:

$$L = -\sum_{i=1}^{N} \sum_{j=1}^{N_C} [\mathbf{t}_i]_j \log([\mathbf{y}_i]_j) \tag{6}$$

where $\mathbf{t}_i \in \mathbb{R}^{N_C}$ is the target output vector, which depends on the label (t_i) of the input image and it is defined as: $[\mathbf{t}_i]_j = 1$, if $j = t_i$, or $[\mathbf{t}_i]_j = 0$, otherwise. For both the input layer and the hidden layer the dropout technique can be used [27]. Dropout with rate $p = 0.5$ is used, except otherwise stated.

3.4. Learning with the CBoF

All the layers of the CBoF network can be trained using back-propagation and gradient descent: $\Delta(\mathbf{W}_{MLP}, \mathbf{V}, \boldsymbol{\sigma}, \mathbf{W}_{conv}) = -(\eta_{MLP}\frac{\partial L}{\partial \mathbf{W}_{MLP}}, \eta_V \frac{\partial L}{\partial \mathbf{V}}, \eta_\sigma \frac{\partial L}{\partial \boldsymbol{\sigma}}, \eta_{conv}\frac{\partial L}{\partial \mathbf{W}_{conv}})$, where the notation $\mathbf{W}_{MLP}$ and $\mathbf{W}_{Conv}$ is used to refer to the parameters of the classification layer and the feature extraction layer respectively. Instead of using simple gradient descent, a recently proposed method for stochastic optimization, the Adam (Adaptive Moment Estimation) algorithm [9], is utilized for learning the parameters of the network.

The convolutional layers can be either learned from scratch or finetuned, when a pretrained feature extractor is used. The hidden weights of the MLP are initialized using random orthogonal initialization [25]. The centers of the RBF neurons can be either randomly chosen or initialized using the k-means algorithm over the set of all feature vectors $\mathcal{S} = \{\mathbf{x}_{ij} | i = 1...N, j = 1...N_i\}$. In this work, the set $\mathcal{S}$ is clustered into N_K clusters and the corresponding centroids (codewords) $\mathbf{v}_k \in \mathbb{R}^D (k = 1...N_K)$ are used to initialize the centers of the RBF neurons. The same approach is used in the BoF model to learn the codebook that is used to quantize the feature vectors. However, in contrast to the BoF model, the CBoF model uses this process only to initialize the centers. Both the RBF centers and the scaling factors (initially set to 0.1) are learned.

3.5. Computational Complexity Analysis

The asymptotic storage requirements for the network and the cost for the feed-forward process are derived in this Section. Note that these quantities are calculated for the part of the network after the last convolutional layer (assuming that N_F convolutional filters are used in the last convolutional layer, the size of each feature map/number of feature vectors is N_i and two fully connected layers are used). The proposed method is also compared to a regular CNN as well as to the SPP technique. For both the CBoF and the SPP techniques the number of used spatial regions is denoted by N_S.

The CNN method requires $O(N_i N_F N_H + N_H N_C)$ weights after the last convolutional layer, the SPP method requires $O(N_S N_F N_H + N_H N_C)$ weights, while the proposed CBoF method requires $O(N_S N_K N_F + N_S N_K N_H + N_H N_C)$ weights. Note that the CBoF method is capable of decoupling the input feature dimensionality N_F from the network architecture, allowing to use significantly smaller networks The cost of feed forwarding the network is $O(N_i N_F N_H + N_H N_C)$ for the CNN, $O(N_i N_F + N_S N_F N_H + N_H N_C)$ for the SPP method and $O(N_i N_K N_F + N_S N_K N_H + N_H N_C)$ for the CBoF method. Both the SPP and the CBoF methods carry an extra cost for the feature aggregation/quantization step ($O(N_i N_F)$ and $O(N_i N_K N_F)$ respectively), but in practice this cost is amortized by the smaller number of operations that are needed after this step (it is reasonable to assume that $N_K < N_H$, as in the conducted experiments). Also, note that since N_i is dependent on the input image size, the cost of the feed forward process can be adjusted for the SPP and the CBoF methods by simply resizing the input image.

4. Experiments

Three different datasets were used to evaluate the proposed method: the MNIST database of handwritten digits (MNIST) [14], the fifteen natural scene (15-scene) dataset [12], and the large-scale Annotated Facial Landmarks in the Wild (AFLW) dataset [10].

The MNIST database [14], is a well-known dataset that contains 60,000 training images and 10,000 testing images of handwritten digits. The training set was split into 50,000 training images and 10,000 validation images that were used to evaluate the learned models and prevent overfitting. There are 10 different classes, one for each digit (0 to 9), and the size of each image is 28×28.

The 15-scene dataset [12], consists of images belonging to fifteen different natural scene categories, such as industrial, forest, city, bedroom and living room scenes. The dataset has a total of 4,485 images, with an average size of 300×250 pixels, and the number of images in each category ranges from 200 to 400 images. Since there is no predefined train/test split, the standard evaluation protocol is used [12]: the train split is composed of 1,500 randomly chosen images (100 from each category), the test split is composed of the remaining 2985 images and the evaluation process is repeated 5 times.

The Annotated Facial Landmarks in the Wild (AFLW) dataset [10], is a large-scale dataset for facial landmark localization. The AFLW dataset was used to evaluate the performance of the proposed method for the problem of facial pose estimation using a light-weight model that can be deployed on embedded devices, such as drones that will assist the video shooting of sport events. Estimating the facial pose of the actors allows for calculating the appropriate shooting angle according to the specifications of each shot. The 75% of the images were used to train the models, while the rest 25% for evaluating the accuracy of the models. The face images were cropped according to the given annotation and face images smaller than 16x16 pixels were not used for training or evaluating the model.

In this work, three different feature extraction blocks are utilized to demonstrate the flexibility of the CBoF. For the first one, the *Feature Extraction Layer Block A*, 32 5×5 convolutional filters are used in the first layer, followed by a 2×2 max layer and 64 5×5 convolutional filters in the last layer. The *Feature Extraction Layer Block B* is deeper and it is composed of two layers with 64 3×3 convolutional filters, a layer with 128 3x3 convolutional filters, a 2×2 max-pooling layer followed by another two 256 3x3 convolutional layers, a 512 3x3 convolutional layer and a 2×2 max pooling layer. These blocks are learned from scratch. For the *Feature Extraction Layer Block C* a pretrained convolutional network with 5 convolutional layers is used [31]. This block was pretrained on the Places205 dataset and the feature vectors are extracted from its last convolutional layer (256 feature maps). Different learning rates must be used for each layer, depending on whether the network is learned from scratch or being finetuned. For the first two setups (learning from scratch) the same learning rate is used for all the layers $\eta_{MLP} = \eta_V = \eta_\sigma = \eta_{conv} = 10^{-4}$. For the third setup different learning rates are used, since the feature extraction layer is already trained,: $\eta_{MLP} = \eta_\sigma = 10^{-4}$, $\eta_{conv} = 10^{-5}$ and $\eta_V = 10^{-2}$.

4.1. MNIST Evaluation

The well known MNIST dataset is used to study the behavior of the proposed method under different settings. First, the effect of the number of RBF neurons (codewords) is examined in Figure 3 using two different spatial levels, spatial level 0 ($N_S = 1$) and spatial level 1 ($N_S = 4$) (as shown in Figure 2). The network is learned from scratch (feature extraction layer block A). All the layers of the network are trained using back-propagation for 50 epochs. When no spatial segmentation is used (spatial level 0), using more RBF neurons reduces the classification error leading to 0.74% error when 128 neurons are used. Using spatial segmentation at level 1 further reduces the error to 0.61 when $4 \times 32 = 128$ codewords are used. It worths noticing that "bottlenecking" the network into just 8 RBF neurons after the convolutional layer, which is less than the classes of the problem, achieves a remarkable classification error of just 1.13% (similar to a LeNet-4 [13], which fully flattens the feature maps).

Next, the ability of CBoF to learn and classify images of various sizes is examined in Table 1 (32 RBF neurons and spatial level 0 are used). The CBoF is evaluated using images of 20×20, 24×24, 28×28, 32×32 and 36×36 pixels. These images were generated by resizing the original MNIST images (using spline interpolation). Three different training setups are considered: Train A, where only 28x28 images are used for training, Train B, where 24×24, 28×28 and 32×32 images are used for training, and Train C, where all the available image sizes are used for training. When the size of an image to be classified is close to the image size

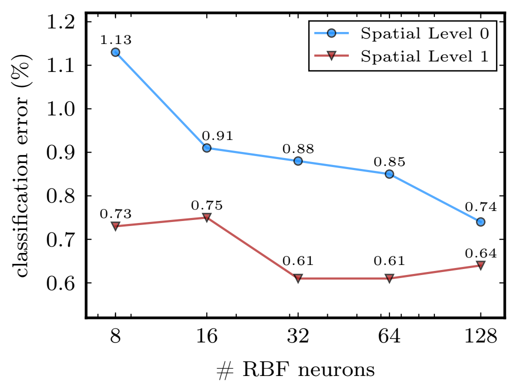

Figure 3. MNIST: Comparing test error of CBoF for different number of codewords and spatial levels.

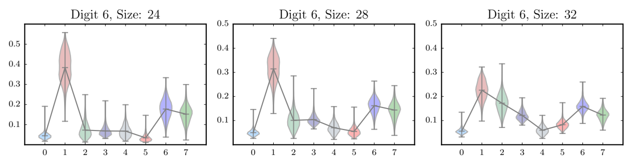

Figure 4. MNIST: Visualizing the RBF activations for different image sizes (without training for different image sizes).

Image Size	20	24	28	32	36
Train A	19.69	1.92	0.88	4.10	25.39
Train B	3.68	0.91	0.73	1.11	2.49
Train C	**1.44**	**0.82**	**0.68**	**0.95**	**1.60**

Table 1. MNIST: CBoF error (%) for different image sizes and train setups: Train A (28x28 images), Train B (24×24, 28×28 and 32×32 images) and Train C (all the available image sizes).

used for training, *i.e.*, ±4 pixels or $\pm14\%$ of training image size, the impact on classification accuracy is relatively small. The scale-invariance of the model is significantly improved when it is trained using images of different sizes (Train B and Train C setups), as shown in Table 1. Also, the multiple size training process leads to better overall accuracy, since the classification error is reduced to 0.68% from 0.88% (for the given setup).

The RBF neurons provide a Voronoi-like segmentation of the space where they operate, *i.e.*, the space of the convolutional feature vectors. This makes the proposed technique relatively invariant to mild distribution shifts. This phenomenon is illustrated in Figure 4, where the activations of the RBF neurons are shown when images of different sizes are fed to the network (which is trained only for 28x28 images). Even though the actual input distribution is shifted, relatively mild shifts are observed in the activation of the RBF neurons (especially for the smaller image).

Finally, in Table 2 the CBoF method (spatial level 1 ($N_S = 1$), 32 RBF neurons) is compared to other state-of-the art techniques for image classification. For the CNN method the same convolutional layers as in the CBoF method are used, combined with a 2×2 pooling layer (dropout with $p = 0.5$ is also used) before the fully connected layer (1000×10). For the Global Max Polling (GMP)/SPP the global pooling/spatial pooling layer is connected after the last convolutional layer. One spatial level is used for the SPP technique. For the GMP, SPP and the CBoF techniques the models were trained using digit images of 20×20, 24×24, 28×28, 32×32 and 36×36 pixels. The classification error for images of two sizes are reported: for the default image size (28×28 pixels) and for a reduced image size (20×20 pixels). The number of parameters used in the layers after the feature extraction layer, which is common among all the evaluated networks, is also reported. All the layers of the networks are learned during the training.

The proposed CBoF outperforms both the CNN and the GMP/SPP techniques, while using significantly less parameters after the convolutional layers (about one order of magnitude less parameters than a similarly performing CNN). Also, it provides much better scale-invariance than the competitive GMP/SPP techniques, even though both techniques were trained using 5 different image sizes. Note that the purpose of the conducted experiments was not to achieve state-of-the-art performance on the MNIST dataset, but to examine the behavior of the proposed method under different training scenarios. Nonetheless, the proposed method achieves performance close to other much more complicated techniques, such as ensembles of neural networks trained using various distortions of the images [2]. These methods can be combined with the proposed approach and possibly increase the classification accuracy even more, while reducing the overall network size. Also, to demonstrate that the actual architecture of the BoF layer contributes to the improved results, the GMP and SPP networks were also evaluated with an added 1×1 convolutional layer with 64 filters. Again, the proposed CBoF method outperforms both the GMP and SPP methods (3.52% classification error for the GMP and 1.66% for the SPP method (for 20x20 images)).

4.2. 15-scene Evaluation

First, the performance of the CBoF is evaluated for different codebook sizes and spatial levels. Instead of using randomly initialized convolutional layers, a pretrained CNN is used (feature extraction layer block C) [31]. The network was trained for 100 epochs. Also, the fully connected layers were pre-trained for 10 epochs before training the whole network to avoid back-propagating gradients from randomly initialized layers. Figure 5 illustrates the precision on the test set for different number of codewords (evaluated on one test split). As before, using more codewords

Method	Cl.Err. (28)	Cl.Err. (20)	# Param.
CNN	0.56	(0.78)*	1,035k
GMP	0.62	3.22	75k
SPP	0.52	1.78	331k
CBoF(0, 32)	0.68	1.44	45k
CBoF(1, 32)	0.55	0.94	147k
CBoF(1, 64)	**0.51**	**0.83**	284k

Table 2. MNIST: Comparing CBoF to other state-of-the-art techniques. The spatial level and the number of RBF neurons are reported in the parenthesis for the CBoF model. (*Training from scratch for 20×20 images)

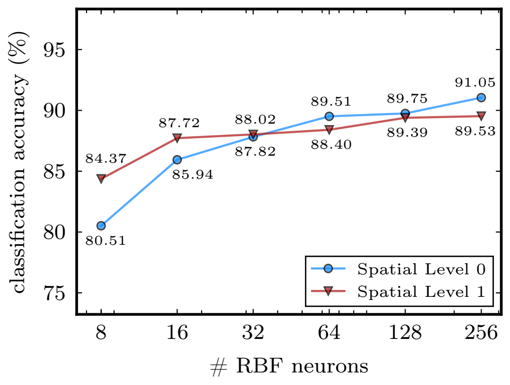

Figure 5. 15-scene: Comparing test accuracy of CBoF for different number of codewords and spatial levels

leads to better classification accuracy when no spatial segmentation is used. Unlike the MNIST dataset, using spatial segmentation does not improve the overall classification accuracy for the 15-scene dataset. This is happening possibly due to the nature of the dataset (the spatial information is less important when recognizing natural scenes than when detecting the edges of aligned digits) and the larger receptive field of the final convolutional filters.

Then, the CBoF model is trained using images of different sizes (one dataset split is used for the evaluation). The results are shown in Table 3. Again, training with different image sizes helps to improve the classification accuracy for any image size. Note that the classification accuracy increases from 90.66% to 92.38%, when multiple image sizes are used for training, highlighting the importance of training the network using different image sizes.

Finally, the CBoF (256 RBF neurons, no spatial segmentation) is compared to other state-of-the-art techniques in Table 4. All the evaluated method share the same feature extraction layer. For the CNN, a 1000×15 fully connected layer is added after the last pooling layer, while the GMP/SPP layer is added between the last convolutional and the fully connected layers Note that the fully connected layer of the pretrained network is not used [31], leading to slightly different reported results for the CNN method.

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 3. 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).

Method	Acc. (227)	Acc. (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
CBoF (0, 256)	$\mathbf{91.38 \pm 0.63}$	$\mathbf{89.86 \pm 0.80}$	337k

Table 4. 15-scene: Comparing CBoF to other state-of-the-art techniques (*Training from scratch for 179×179 images)

Instead, all the layers of the network are learned during the training procedure using back-propagation. The following learning rares were used for the CNN, GMP, and SPP methods: $\eta_{conv} = 10^{-3}$ and $\eta_{conv} = 10^{-5}$. Again, the proposed CBoF method outperforms both the CNN and the GMP/SPP techniques while using significantly less parameters. The same is also true when smaller images are fed to network. 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 there techniques, even though one extra layer was added.

4.3. AFLW Evaluation

Finally, the proposed method is evaluated on AFLW dataset. The Feature Extraction Layer Block B was used for these experiments and a hidden layer with 1000×1 neurons was utilized. The models were trained for 5000 iterations, while the CBoF and the GMP/SPP methods were trained using three image sizes, *i.e.*, 32×32, 64×64, and 96×96 pixels (for each iteration batches from each size were fed to each network). No dropout was used after the feature extraction layer block for the GMP, SPP and CBoF methods, since it was established that slows down the training process significantly. All the networks were trained to perform regression on the horizontal facial pose (yaw) using the mean squared error as the objective function for the training procedure. The experimental results are shown in Table 5. The mean absolute angular error (Ang. Err.) in degrees is reported. As in the previous experiments, the proposed method significantly reduces the size of the models, while still increasing the accuracy of the yaw angle estimation. Note that even though the CBoF (0, 128) and the CBoF (1, 32) use the same number of parameters, the spatial segmentation scheme used in CBoF (1, 32) significantly increases the accuracy since the spatial information is especially important for the task of facial pose estimation.

Method	Ang. Err. (64)	Ang. Err. (32)	# Param.
CNN	14.50	(13.91*)	61,954k
GMP	13.70	12.83	514k
SPP	12.37	11.96	2,562k
CBoF (0, 128)	16.23	13.84	196k
CBoF (1, 32)	**11.92**	**11.25**	196k

Table 5. AFLW: Comparing CBoF to other state-of-the-art techniques (the mean absolute angular error in degrees is reported) (*Training from scratch for 32×32 images)

5. Conclusions

In this paper, a neural extension of the well-known BoF model was combined with convolutional neural networks to form powerful image recognition machines. It was demonstrated, using three image datasets, that the proposed CBoF model is able to natively classify images of various sizes as well as to significantly reduce the number of parameters in the network, while achieving remarkable classification accuracy. In contrast to CNN compression techniques, that simply compress an already trained CNN, the proposed method provides an improved CNN architecture that is end-to-end differentiable and it is invariant to mild distribution shifts. Nonetheless, the proposed technique can be combined with CNN compression techniques, such as [5, 6, 29], to further reduce the size of the resulting network. Furthermore, the proposed RBF-based quantization scheme can be approximated using only linear layers instead of RBF-layers increasing the classification speed even more. Preliminary experiments show that this approach can indeed increase both the training and the testing speed, without significantly harming the accuracy of the models. The proposed pooling layer can be further enhanced by also learning the spatial regions over which the pooling is performed, similarly to [19], or using visual attention mechanisms [30].

Acknowledgment

This project has received funding from the European Unions 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] W. Chen, J. T. Wilson, S. Tyree, K. Q. Weinberger, and Y. Chen. Compressing convolutional neural networks. *arXiv preprint arXiv:1506.04449*, 2015.

[2] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3642–3649, 2012.

[3] 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.

[4] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 886–893, 2005.

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

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

[7] K. He, X. Zhang, S. Ren, and J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. In *Proceedings of the European Conference on Computer Vision*, pages 346–361, 2014.

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

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

[10] 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 *First IEEE International Workshop on Benchmarking Facial Image Analysis Technologies*, 2011.

[11] S. Lazebnik and M. Raginsky. Supervised learning of quantizer codebooks by information loss minimization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(7):1294–1309, 2009.

[12] 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*, volume 2, pages 2169–2178, 2006.

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

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

[15] 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*, pages 157–170. 2010.

[16] 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*, pages 87–98. 2014.

[17] D. G. Lowe. Object recognition from local scale-invariant features. In *Proceedings of the IEEE International Conference on Computer vision*, volume 2, pages 1150–1157, 1999.

[18] L. Ma, Z. Lu, and H. Li. Learning to answer questions from image using convolutional neural network. *arXiv preprint arXiv:1506.00333*, 2015.

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

[20] T. Ojala, M. Pietikainen, and T. Maenpaa. Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(7):971–987, 2002.

[21] 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*, pages 1717–1724, 2014.

[22] N. Passalis and A. Tefas. Entropy optimized feature-based bag-of-words representation for information retrieval. *IEEE Transactions on Knowledge and Data Engineering*, 28(7):1664–1677, 2016.

[23] N. Passalis and A. Tefas. Neural bag-of-features learning. *Pattern Recognition*, 64:277–294, 2017.

[24] 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*, 115(3):211–252, 2015.

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

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

[27] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[28] 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*, 32(7):1271–1283, 2010.

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

[30] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning*, pages 2048–2057, 2015.

[31] 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*, pages 487–495, 2014.