String representations and distances in deep Convolutional Neural Networks for image classification

Cécile Barat, Christophe Ducottet

String representations and distances in deep convolutional neural networks for image classification

Cécile Barat[a], Christophe Ducottet[a,*]

[a] *Université de Lyon, CNRS, UMR 5516, Laboratoire Hubert Curien, Université de Saint-Étienne, Jean-Monnet, F-42023, Saint-Étienne, France.*

Abstract

Recent advances in image classification mostly rely on the use of powerful local features combined with an adapted image representation. Although Convolutional Neural Network (CNN) features learned from ImageNet were shown to be generic and very efficient, they still lack of flexibility to take into account variations in the spatial layout of visual elements. In this paper, we investigate the use of structural representations on top of pre-trained CNN features to improve image classification. Images are represented as strings of CNN features. Similarities between such representations are computed using two new edit distance variants adapted to the image classification domain. Our algorithms have been implemented and tested on several challenging datasets, 15Scenes, Caltech101, Pascal VOC 2007 and MIT indoor. The results show that our idea of using structural string representations and distances clearly improves the classification performance over standard approaches based on CNN and SVM with linear kernel, as well as other recognized methods of the literature.

Keywords: Convolutional Neural Network, string representation, edit distance, image classification
2010 MSC: 00-01, 99-00

1. Introduction

The dominant approach to image classification has long been to extract image representation vectors from local handcrafted features, e.g. SIFT or SURF. Traditional approaches such as bag of visual words (BoVW) with spatial pyramid representation (SPR) or Fisher vectors (FV) first extract the local features, then encode them using a visual vocabulary and finally pool them into a single image representation vector. In a second step, a classifier such as Support Vector Machine (SVM), is trained on all resulting vectors to later recognize the class of unknown images.

Recently, Convolutional Neural Networks (CNNs) have provided outstanding performance in the field [1, 2, 3]. It is attributed to their ability to simultaneously learn

[*] Corresponding author, Tel: +33 477915787, Fax: +33 477915781
Email address: ducottet@univ-st-etienne.fr (Christophe Ducottet)

discriminative local features and a powerful image representation vector from an image without any preprocessing requirement. A CNN structure is generally composed of alternating convolutional layers and sub-sampling layers, followed by fully connected layers and the output layer. The latter is often a softmax classifier designed to generate probabilistic labels for each class. It is common to find hybrid approaches that combine a SVM classifier with a CNN feature extractor pretrained on a large dataset such as ImageNet. The overall idea is to cut off the output of CNN one or two layers before the output layer and thereby use the image representation vector obtained as input for SVM. Doing so, it is expected to benefit from generic pretrained CNN features and learn specific models on smaller datasets using SVM. Instead of using an SVM to learn the specific model, another alternative is to refine the pretraining of the CNN by adding a fine-tunning step with the target dataset [2, 3].

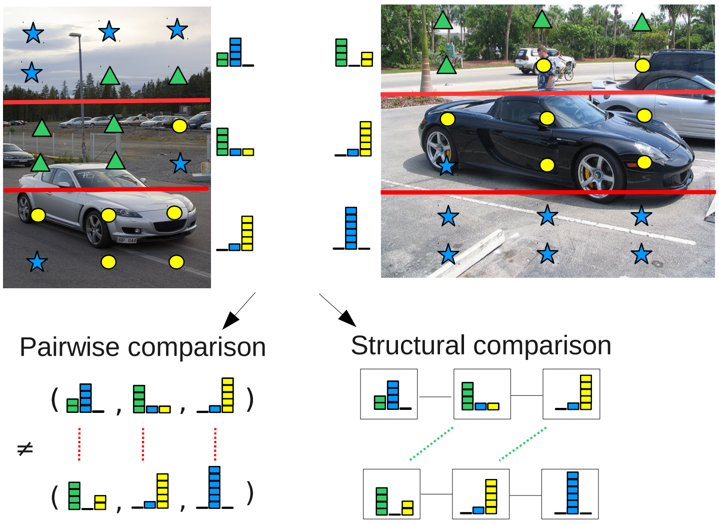

Figure 1: Pairwise and structural comparison of two images. Images are represented using spatial sum pooling of features. For the purpose of simple illustration, three types of local features are used to denote car features (yellow circles), background features (green triangles) and uniform features (blue stars). In classical approaches, similarity is computed using pairwise region-to-region comparisons. Here, the matching fails because the object of interest, i.e. the car, appears at different locations within images. In structural approaches, images are considered in terms of their parts. Parts at different positions can be matched together in order to compensate some geometric transformation effects, as translation here.

In all previous approaches, an image is represented as a fixed length vector. The importance of capturing the spatial layout of the image through this global vector has been recognized for object and scene classification tasks. In contrast to BoVW which ignores the spatial layout of the features, CNN tends to preserve it too much, reducing in all cases the invariance to geometric transformations. Invariance of the BoVW model is coupled with the invariance properties of the local features. Spatial pyramid pooling was definitely the most noticeable contribution to integrate their spatial layout in the model. In SPR like methods, the final representation vector captures the local invariance but assumes that similar parts of a scene or an object generally lay in similar regions of the image, limiting the robustness to translation, rotation, scaling or other viewpoint change. In the same manner, CNN pretrained on very large datasets such as ImageNet has the ability to learn complex transformations and the local invariance of features through max-pooling within each feature map. The fully connected layers encode their spatial layout, independently from the image content, limiting the robustness to geometric transformations. Recent research works aim to propose more orderless representations on top of CNN to improve classification results, for examples, for texture [4] or for scene [5, 6, 7, 8]. To sum up, a first limitation of these models, while very efficient, is that they resolve local visual ambiguities, but still, the final global representation vector is not invariant to geometric transformations.

A second limitation of these representations is the lacking ability to represent relationships that might exist between different regions of an image. As a result, to compute a measure of similarity between each pair of images as required by SVM, the only possibility is to make pairwise region-to-region comparisons, limiting the ability of the system to recognize images that are visually similar but not identical because of some geometric transformation of the total image or one of its part. It would seem interesting to introduce flexibility during the matching process and find the best correspondences between the different regions taking into account their visual content (Figure 1).

So far, in the mentioned methods, image representation vectors capture the local properties of images, and an approximate global matching of these local features among images is actually computed for classification. Our idea in this paper is to increase the representation power of these models by using a structural approach. We propose to model images as strings of visual features. Combining SPR and CNN models with strings allows us to get the best of the two worlds and offers several advantages. First, it allows keeping the discriminative information about the local regions, but also to describe their adjacency dependencies, and implicitly the image structure. An image is now considered in terms of its parts. Second, the concept of approximate matching can be applied to make the comparison of two images more robust to geometric transformations. One popular category of methods for approximate matching uses the edit distance. It can be seen as an active process that finds the best alignment between two strings owing to three main edit operation, i.e. insertion, deletion and substitution. Importantly, parts at different positions in images can now be compared and the number of parts does not have to be fixed. It may be adapted to the visual content. Third, an edit distance enables to define a string kernel encoding similarity between each pair of images represented as strings, and therefore, it offers the possibility to benefit from all SVM classification tools on structured data.

In this paper, we investigate the use of structural representations of images to improve image classification. Our contributions are multiple. First, we propose solutions to generate strings with local generic features extracted from a pretrained CNN. A great advantage of using available pretrained CNN tools is to avoid the time consuming CNN training or fine-tuning step. Second, we introduce two edit distance variants dedicated to approximate matching between images represented as strings of visual feature vectors. Third, through a thorough evaluation of our method on different datasets, we show the benefit of taking into account the structural relationships between local CNN features compared to a standard pairwise matching. Also, we show it is more efficient than learning weights of a fully connected layer. Fourth, in our approach with CNN, any size of image can be processed by the convolutional and pooling layers without any resizing.

Moreover, a single pass through the network is required.

The paper is organized as follows. Section 2 presents the related work and section 3 introduces our approach to generate strings from local features. Section 4 explains how to use an edit distance to compare such strings and proposes two new edit distance variants adapted to the image classification domain. Section5 describes experiments and results on image classification tasks. Section 6 concludes the paper and indicates future work.

2. Related work

In the image domain, several problems have successfully been modelled and solved using strings rather than local feature vectors, e.g. text recognition [9, 10, 11], shape matching [12, 13], image classification [14, 15, 16, 17] and video classification [18]. In this section, we focus on the related work that addresses the question of spatial information and topological relationships within SPR and CNN frameworks for image classification.

2.1. SPR based models

Inspired by SPR, many methods rely on spatial pooling to form an image signature vector that encodes spatial information. Different strategies can be adopted to partition an image into spatial bins. The standard SPR model uses three layers of grids $1 \times 1, 2 \times 2, 4 \times 4$. Other grid configurations have been proposed combining for instance $1 \times 1, 3 \times 1$ (three horizontal lines), 2×2 divisions [19, 20, 21]. [22] suggested to learn the grid divisions. Other configurations were proposed such as pyramidal rings [23], oriented partitions [24] or randomized partitions [25] to make the image representation more robust to geometric transformations. Once the partitioning scheme is defined, most methods consist in computing a local BoVW within each of the spatial bins and concatenate the obtained vectors. The classification step is performed using a distance between those image vectors, ending up to measure the similarity between pairwise spatial regions. These methods offer the advantage to decompose an image into its parts, but loose their topological relationships.

Some attempts have been made to introduce order and topological information into this type of representations using strings [16, 17]. In [16], the authors use a 4×4 partitioning (SPR level 2). Then, an image is represented as a string of 16 local SIFT BoVW obtained following the raster-scan ordering. Images are compared using an edit distance metric as discussed in section 4. The drawback of this approach is that two successive symbols in the string may not correspond to neighboring regions of the image. To overcome this drawback, authors in [17] suggest to represent an image as a set of multiscale strings. First, they use a multiscale grid partitioning. Second, strings are formed by scanning vertically each column of the grids. Doing so, in each string, neighboring symbols correspond to adjacent regions in the image. The vertical ordering is explained by the natural sequencing of objects in a scene along the vertical direction. In our work, as explained in the following section 3, we retain this approach to build strings from images decomposed into grids.

2.2. CNN based models

CNN have emerged as the new state-of-the art approach for image classification. The standard approach consists in pre-training the CNN architecture with a large dataset (ie ImageNet) and to learn a specific classification model either by training a SVM or by fine tunning the CNN on the target dataset [1, 2, 3]. Interestingly, as said in introduction, a CNN model pre-trained on a large dataset can be used as a generic feature extractor. Several available tools exist for this purpose such as *OverFeat* [26] or Caffe [27].

While the output feature vector is robust to local changes, its robustness to global deformations needs improvements since it preserves too much spatial information. Several very recent works address this question [5, 6, 7, 8]. In [5, 7, 8], authors propose a similar coarse-to-fine analysis of the image within a pyramid structure at N levels. At each level, all patches of a given size are extracted from an image and fed to a CNN network. The 7th layer output which is 4096 dimension is taken as the representation vector of the patch. At level 1, the global image is processed. The three methods differ in the way the obtained CNN features are encoded. In [5], all patch responses are aggregated using a VLAD approach. In [7], a codebook is first learnt from these CNN features and second, the final representation vector is obtained using Locality-constrained Linear Coding and max pooling. In [8], the final representation vector is obtained by average pooling of patch responses of a given scale and concatenation of the different scale resulting feature vectors. These three interesting approaches unfortunately require a lot of computation time to extract the feature vectors from images because each patch must be processed by the CNN. Moreover, the spatial layout of patches extracted at a given scale is ignored.

Another closely related work is the work of He et al. [6], where it is suggested to replace the last pooling layer with a spatial pyramid pooling layer. The obtained vector is then fed to the fully-connected layer. To handle variable image sizes, the authors propose to train several versions of the CNN at various sizes which increase the time required to train the CNN. Moreover, as in BoVW models, this approach increases the local invariance properties, but still preserves the global pairwise matching of the local features that limit the image comparison during the classification step.

To the best of our knowledge, no work in the field so far attempted to unify the CNN based models and structural approaches to improve classification performance.

3. Image representation

Our proposition is to combine the great potential of different ideas seen previously to capture information about local regions together with their spatial adjacency properties. We propose to model images using 1) pretrained CNN features, 2) spatial pyramid pooling applied at the output of the last convolution layer (layer 6) 3) a set of multiscale strings. The pipeline is illustrated in Figure 3.

Note that the CNN is just used as a feature generator. In our work, we never train the CNN with any test dataset, but we use the weights trained from ImageNet by Sermanet et al. [26] team.

3.1. CNN feature extractor

The architecture of the CNN we consider for feature extraction is based on the model of Krizhevsky et al. [1]. We use the *OverFeat* implementation ([26]) available with two network architectures: a fast model and an accurate one. We chose the accurate model pre-trained on the ImageNet ILSVRC 2013 dataset ([28]). It is composed of nine layers including 6 convolutional ones and 3 fully connected ones, the last one being a softmax classifier (Figure 2).

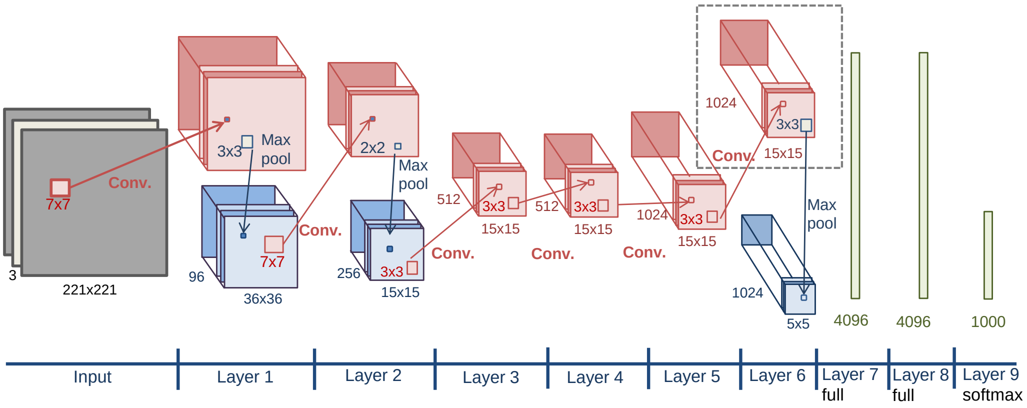

Figure 2: Architecture of the fast $Overfeat$ model. An input image of size 221*221 is fed to the network. This image is then passed through several convolutions (red) and max pooling (blue) operations at different layers. Layers 7 and 8 are fully connected layers. The final layer is a softmax classifier. Note that the last convolutional feature maps at layer 6 (dashed grey box) are used in our pipeline.

This architecture operates on images of spatial input size of 221×221 pixels. This constraint, imposed by the fully connected layers, often appears as a strong limitation in the use of CNN for image classification. However, in $OverFeat$, when the feature extractor mode is used, a sliding input window principle is implemented. The input images can be of any size greater than the spatial input size. It is then decomposed into overlapping square 221 pixels windows with an offset of 36 pixels in each dimension. The resulting feature matrix available at the output of a CNN layer has a spatial size augmented by one unit each 36 additional pixels in each spatial dimension.

In our work, to deal with arbitrary input image sizes without cropping, we first resize the image so that the smallest dimension becomes 221 pixels and the other dimension preserves the aspect ratio. Indeed, although the feature extractor may not require any resizing, a standardization of the input size is a common practice to provide efficient image classification for datasets of variable image sizes. Then, in our method, we extract features from the last convolution layer (layer 6, before max pooling) to build the strings. In addition, to compare our results with previous works, we also extract features from the first fully connected layer (layer 7) as a baseline. At the convolutional output of layer 6, feature vectors have 1024 dimensions and the feature matrix has a minimum spatial size of 15×15. If X is the largest spatial size of the image after resizing, the feature matrix is expended by $(X - 221)/12$ units in this dimension. The 12 factor is the sub-sampling ratio of the network at this layer. At the output of layer 7, there is only one feature vector of 4096 dimensions per input window. The same way, $(X - 221)/36$ additional feature vectors are computed in the largest dimension.

It is common to observe that image classification results increase with the number of features. When using a CNN feature extractor, it is possible to generate higher number of features by reducing the stride between them. Two approaches are possible: reducing the stride only or reducing the stride and the scale of features simultaneously. The stride reduction alone relies on the computing of several sets of features using translated versions of the input image [26]. The translation factor is chosen to be a fraction of the sub-sampling ratio of the features at the given layer. In our case, to get four times more features at the output of layer 6, we use four images translated by 6 pixels in each direction (the sub-sampling ratio is 12). To reduce the size and the scale of features simultaneously, we only need to start from a larger input image. Instead of using an image of 221 pixels in the smaller dimension, we resize it so that it is becomes 442 pixels. This produces also four times more features together with decreasing their scale in the original image.

3.2. Spatial pyramid pooling and string representation

As just explained, our method extracts CNN features at layer 6. Then, our proposition is to pool the features across a spatial pyramid and to structure the spatial bins of the pyramid as a set of strings.

Because of the 1D structure of strings, we must first choose a scanning direction. Although this choice may appear arbitrary, several works argue the use of a particular direction. For example, [24] suggests that vertical or horizontal directions can plausibly better describe relationships among local features. For instance, the sky is above trees, and trees are above grass. Similarly, for urban scenes, in [20], the authors propose to replace the SPM grid division with divisions along the vertical axis to better take into account the composition of this kind of images. Here, following the string definition of [17], we choose to define a set of P vertical strings by scanning each pyramid level column by column. For example, with the standard 3 levels 4×4, 2×2, 1×1 pyramid, we can build 4 vertical strings of 4 symbols at level 2, 2 vertical strings of 2 symbols at level 1 and 1 string of 1 symbol at level 0.

In standard pairwise matching methods, the number of bins in the spatial pyramid must be kept small to avoid matching errors. In our flexible matching approach, higher bin numbers can be used in the direction of strings, leading to a non symmetric partitioning. Formally, we introduce two parameters H and W to define our spatial pyramid. Parameter H (respectively W) represents the number of divisions along vertical (respectively horizontal) axis for the highest level of the pyramid. The number of division for lower levels are obtained after division by successive powers of two until having only one horizontal division. In the standard configuration of our feature extractor, a maximum of $H = 15$ divisions along the vertical axis are possible. In addition, to the previous levels we systematically add a global representation of the image by pulling all feature vectors in a single spatial bin. Figure 3 illustrates the string construction with $H = 8$ and $W = 3$.

After pooling, attention must be paid to normalization of resulting feature vectors. Following the classical spatial pyramid representation, we first normalize feature vectors from each spatial bin independently and we further normalize each pyramid level according to its total number of bins. Vector normalization depends on the classification kernel used for classification. In the linear case a ℓ^2 normalization is applied whereas in χ^2 based kernels a ℓ^1 normalization is required.

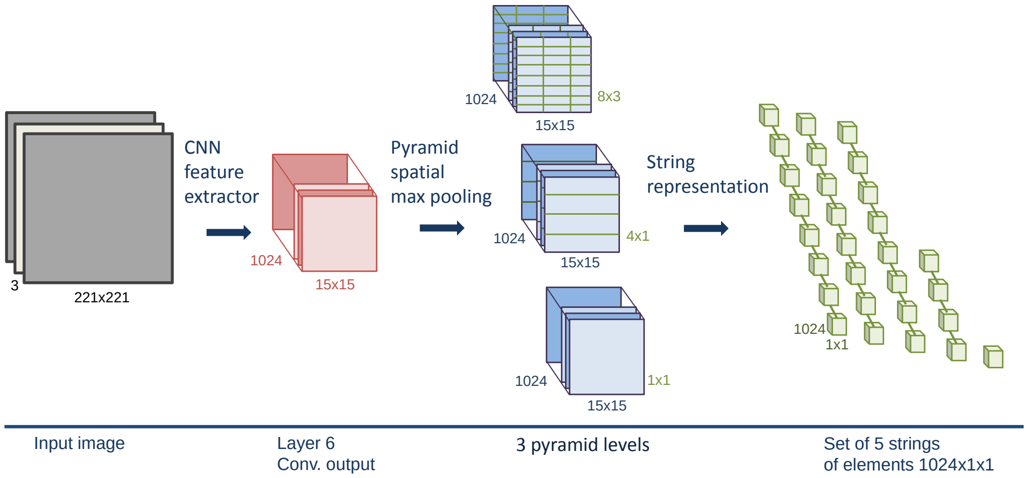

Figure 3: String construction pipeline. The input image is fed to the first 6 layers of the CNN architecture. The 1024 convolutional maps are spatially max pooled according to a given decomposition scheme. Here, a vertical decomposition scheme 8×3, 4×1, 1×1 is illustrated. Each vertical band gives rise to a string. The number of symbols in the string depends on the previous decomposition scheme and is equal to the number of vertical adjacent regions (here, 3, 1 and 1). The dimension of each symbol is equal to 1024, resulting from the concatenation of the 1024 feature values obtained in each pooled region.

Typical parameters considered in experiments are $H = 15$ and $W = 3$ which corresponds to the three pyramid levels 15×3, 7×1 and 1×1. This spatial pooling generates a set of $P = 5$ strings by scanning each pyramid level column by column: three strings of 15 feature vectors each, one string of 7 feature vectors and a final 1 feature vector string representing the whole image. In section 5, we will investigate the influence of parameter H by taking $H = 15$, $H = 8$ or $H = 4$.

4. An edit distance for strings of visual features

Comparing strings of visual features is interesting in image classification. Instead of computing an approximate global matching of the visual features, as in SPR-based methods, it enables to find the best alignment of the visual features in the two images that makes it more robust to geometric transformations. Parts at different positions in images can now be compared. The best alignment results from a sequence of edit operations and provides a measure of similarity of the two images. It is computed using the standard edit distance, which will be first recalled in this section. Then, we address the question of finding edit operations and costs adapted to the context of strings of visual feature vectors describing image regions and their dependencies.

4.1. The standard edit distance

The standard edit distance allows computing the optimal alignment of two strings of symbols. A set of edit operations is defined to modify the input string $X = x_1 x_2 \ldots x_N$ into the output string $Y = y_1 y_2 \ldots y_M$. Several sequences of edit operations exist to transform X into Y. A cost function specifies the cost of each edit operation. In this way, the global cost of an edit sequence can be computed as the sum of all individual operations. The idea of the distance is the lower the cost of the necessary operations the more similar the strings. The edit distance between two strings is indeed defined as the minimum cost of all possible sequences of edit operations that transform X into Y. The supported edit operations with their associated cost functions are as follows:

- insertion: a symbol y_j can be inserted into X with cost $c_{ins}(y_j)$,
- deletion: a symbol x_i can be deleted from X with cost $c_{del}(x_i)$,
- substitution: a symbol x_i can be replaced by a symbol y_j with cost $c_{sub}(x_i, y_j)$.

A simple example of alignment of two strings using the edit operations is shown in Figure 4. It can be retained that the edit distance is able to measure the similarity of corresponding symbols (with the substitution operation), even if they are not at the same spatial corresponding position. Also, it integrates information about the dissimilarity of unmatched symbols (with the insertion and deletion operations).

Computing the standard edit distance can be formulated as an optimization problem and can be carried out with a dynamic programming algorithm. The algorithm consists in computing a $D(N, M)$ matrix, where $D(i, j)$ represents the minimum cost of transforming $X = x_1x_2 \dots x_i$ into $Y = y_1y_2 \dots y_j$, with allowable edit operations mentioned above. The computational complexity is proportional to the product of the length of the two strings, *i.e.* in $\mathcal{O}(N \times M)$. The computation is carried out using the following recurrence relation:

$$\begin{cases} D_{0,0} = & 0 \\ D_{0,j} = & D_{0,j-1} + c_{ins}(y_j), \quad {\scriptstyle j=1\dots N} \\ D_{i,0} = & D_{i-1,0} + c_{del}(x_i), \quad {\scriptstyle i=1\dots M} \\ D_{i,j} = \min\big(& D_{i-1,j} + c_{del}(x_i), \\ & D_{i,j-1} + c_{ins}(y_j), \\ & D_{i-1,j-1} + c_{sub}(x_i, y_j)\big), \\ & \quad {\scriptstyle i=1\dots M, j=1\dots M} \end{cases} \tag{1}$$

Note that a variant where only the substitution operation is allowed corresponds to a classical pairwise matching approach.

The three classical edit operations are known to be powerful to transform one string into another one and compute their similarity. However, these edit operations can be defined in many ways, and have to be adapted to the application domain. In the present paper, we are concerned with image classification and strings of feature vectors describing image regions. Two questions arise: what does it means to insert/remove feature vectors when comparing two images? Also, the costs assigned to the three edit operations affect the resulting optimal alignment between two strings. If the costs are changed, then the optimal alignment may also be changed. A second important question is how to choose the costs for strings of visual features. We bring answers in the next sections with two edit distance variants adapted to image classification.

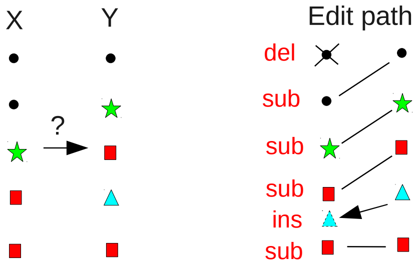

Figure 4: A toy example to illustrate approximate string matching with the standard edit distance. Strings are composed of four types of symbols (dots, triangles, circles and squares). String X has to be modified to match string Y using the classical edit operations. Substitutions occur when symbols are similar. Deletions and insertions allow mismatches to be corrected. The sequence of edit operations to align the two strings is : del - sub - sub -sub - ins - sub.

4.2. Edit distance for image classification

Some edit distance variants have been proposed in many pattern matching applications where strings are composed of a finite set of symbols. For example, several approaches have been experimented for Optical Character Recognition and word spotting in ancient document images [29, 11]. Few methods have been presented to consider images represented as strings of feature vectors in image classification [14, 16, 17]. In [14], the authors propose different solutions to generate strings from the the local signatures of interest points. The feature vectors do not represent region information as in our approach, and their proposed distance does not apply. In [16, 17], images are represented as strings of local signatures of regions, i.e. local bag of words, as discussed in 2.1. To compare such representations, in [16], the authors employ a variant of the edit distance called the Smith-Waterman distance. Its purpose is to find local alignments between two feature sequences and thus to search for a short sequence X into a long string Y, which is not in our focus. In [17], the authors modify the deletion / insertion operations to virtually adapt the fixed grid initial partitioning. Their new edit operations are indeed used to remove repetitions between consecutive similar symbols in a string when it allows the second string to be better matched. Despite its good results, the main drawback of this approach is to loose the information carried by the deleted histograms, as shown in Figure 5(a).

Here, we propose two versions of the edit distance to compare strings of visual features. A symbol in the string corresponds to a CNN feature vector describing an image region. The first version, denoted cED, could be qualified of traditional. It uses the three classical edit operations. It counts the overall cost of single symbols manipulations. The second version, denoted mED, is augmented with a new merge operation, which allows a combination of symbols to be matched with either one or many combined symbols. It further extends the notion of inserting/ deleting symbols. The experimental section will study their performance over rigid matching.

4.3. cED and mED definitions

cED definition. The cED algorithm works with the three basic edit operations. As explained previously, the background idea of substitution is to measure the similarity between corresponding symbols, i.e. CNN vectors. An immediate solution is to apply a classic distance d between feature vectors, such as ℓ_1, ℓ_2 or χ^2. The result defines the cost of the operation. The lower the cost, the more similar the feature vectors and the underlying image information. Insertion and deletion model the notion of dissimilarity between unmatched regions. We propose to associate a fixed cost c to these operations that acts as a penalty factor. Practically, when feature vectors are normalized, this fixed cost could be taken as the distance to the null vector which is equal to the norm of the feature vector. Finally, our proposition comes to consider that two CNN feature vectors are similar if their distance is inferior to the fixed cost, i.e. two regions are similar if they have similar CNN features, otherwise a mismatch occurs which is penalized by the fixed cost in the computation of the distance. It is a natural adaptation of the edit operations to evaluate image content similarity and dissimilarity. Formally, these rules lead to the following costs functions:

$$c_{sub}(x_i, y_j) = d(x_i, y_j) \tag{2}$$

$$c_{del}(x_i) = c \tag{3}$$

$$c_{ins}(y_j) = c \tag{4}$$

The algorithm requires to compute a dynamic programming table exactly as for the standard edit distance, with the same complexity.

A merge operation between symbols. In images, due to geometric transformations such as scale or translation variations, similar content may occupy a varying number of regions of the grid partition. These changes cause mismatches that are penalized in the alignment process with cED, while content are quite similar. Authors in [17] suggested such changes can be seen as adding (or removing) regions similar to their neighbourhood. As a result, they proposed to define insertion / deletion costs as a function of the distance between two neighbouring regions. If one symbol is more similar to its following than to the corresponding one in the other string, the symbol is removed, loosing some information, as shown in Figure 5(a). We follow the background idea of this approach, denoted sED. However, while in sED the authors completely remove some regions information and manipulate individual symbols, we propose to introduce a merge edit operation that enables to model merge and split operations of regions in an image in order to better match with another image without any loss of information.

The merge operation works on two successive symbols in a string. It can be used to associate a pair of symbols of the input string $\{x_i, x_{i+1}\}$ with a symbol of the output string y_j, or inversely, it can be used to match a symbol x_i with a combination $\{y_j, y_{j+1}\}$ of the output string. Since successive merge operations can be applied either in the input string or the output string, a sequence of symbols $\{x_i, x_{i+1}, \ldots, x_{i+k}\}$ can be matched with a sequence of symbols $\{y_j, y_{j+1}, \ldots, y_{j+l}\}$. The merge operation generalizes the insertion and deletion operations. From an image point of view, the merge operation allows a group of consecutive regions of an image to be matched against a group of regions in the second image. It means that if an object is broken into several regions, the corresponding local features may be grouped together to match with the same set of features of the object distributed in one or more regions in the second image. An example is given in Figure 5(b).

More formally, given a sequence of feature vectors $\{x_i, x_{i+1}, \dots, x_{i+k}\}$, the merge operation consists in creating a new feature vector denoted $\bar{x}_{i\to i+k}$ obtained after combination of the corresponding spatial regions. Following the classical pooling operations used in BoVW or CNN, the combination can be a pooling corresponding to a component-wise max or sum:

$$\bar{x}_{i\to i+k} = \text{pool}(x_i, x_{i+1}, \dots, x_{i+k}) \tag{5}$$

The cost function of the merge operation between two symbols is defined as the ground distance d between the two symbols:

$$c_{merge}(x_i \to x_{i+1}) = d(x_i, x_{i+1}) \tag{6}$$

If the first symbols results from the merge of previous symbols, the corresponding cost is defined recursively by

$$c_{merge}(x_i \to x_{i+k}) = c_{merge}(x_i \to x_{i+k-1}) + d(\bar{x}_{i\to i+k-1}, x_{i+k}) \tag{7}$$

As in *cED* or *sED*, the distance d can be any distance between feature vectors. To properly compare symbols resulting from various numbers of merge operations, the symbols must be renormalized before distance computation. In experiments, we choose the χ^2 distance between symbols and we normalized them in ℓ_1 before distance computation.

Definition of mED - a merge-based edit distance:. The same way as the standard edit distance, a merge-based distance is computed by finding the optimal alignment of two strings considering merge and substitution operations. Unfortunately, the dynamic programming algorithm can not be used because symbols are modified each time a merge operation is applied. However, a recursive definition can be derived noticing:

- successive merge operations in the first or the second string can be done in any order leading to identical merged symbols and costs,
- after a substitution, the two remaining substrings are identical to the original substrings.

Let us denote $\text{mED}(X, Y)$ the merge-based edit distance between two strings X and Y. Then, the minimum cost $c_{script}(i, j)$ of all scripts starting with i merging operations in string X and j merging operations in string Y is given by:

$$\begin{aligned} c_{script}(i,j) =& c_{merge}(x_1 \to x_i) + c_{merge}(y_1 \to y_j) \\ &+ c_{sub}(\bar{x}_{1\to i}, \bar{y}_{1\to j}) \\ &+ \text{mED}(\{x_{i+1}, \dots x_M\}, \{y_{j+1}, \dots y_N\}) \end{aligned} \tag{8}$$

Thus, the final distance is obtained by minimizing the cost over all scripts starting with merge operations in one of the two strings:

$$\begin{aligned} \text{mED}(X,Y) = & \min_{i=1\to M, j=1\to N} (c_{merge}(x_1 \to x_i) \\ &+ c_{merge}(y_1 \to y_j) \\ &+ c_{sub}(\bar{x}_{1\to i}, \bar{y}_{1\to j}) \\ &+ \text{mED}(\{x_{i+1}, \dots x_M\}, \{y_{j+1}, \dots y_N\})) \end{aligned} \tag{9}$$

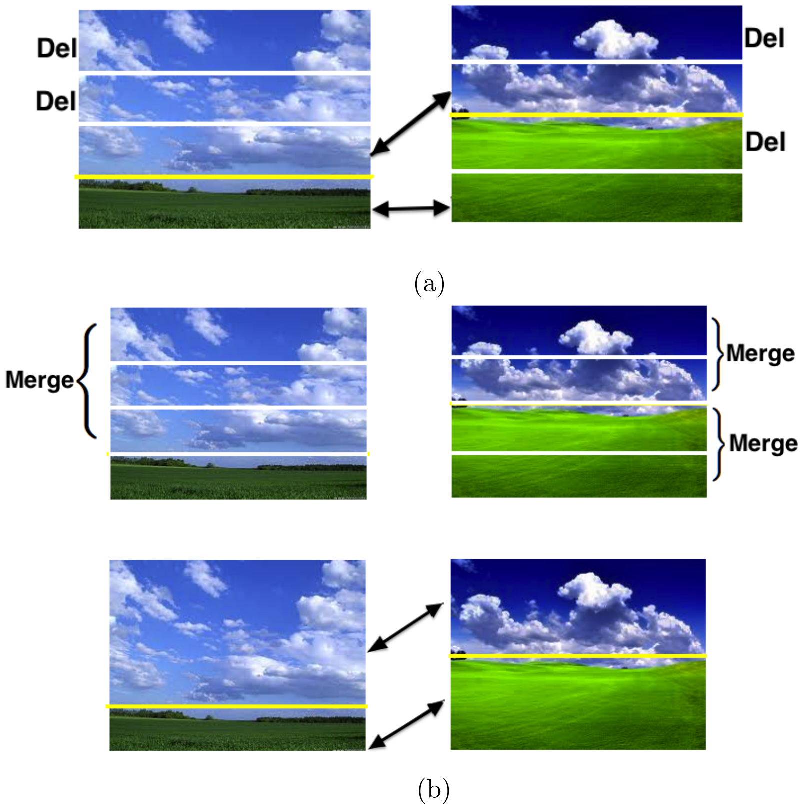

Figure 5: Comparison of sED and mED variants of edit distances. (a) Method sED proposed in [17] (b) the proposed merge-based edit distance, mED. Both methods find an alignment to better compare the visual content of the two images than direct rigid matching. However, method (a) removes some regions that are similar to the neighbouring ones, while the second one keeps all the information by merging their feature vectors.

This definition can be directly converted into a recursive algorithm. The computational complexity of this algorithm in terms of number of inter-symbol distance computations is $\mathcal{O}(N^2M^2)$. Indeed, for each evaluation of equation 9, NM distances between merged symbols must be computed. Although these distances can be precomputed to prevent re-computation during nested recursions, the total number of distances to evaluate is proportional to N^2M^2. The complexity can be reduced to $\mathcal{O}(NM)$, if merged symbols $\bar{x}_{1\to k}$ are approximated by pooling at most the last $k_{max}+1$ symbols such that:

$$\bar{x}_{i\to i+k} = \text{pool}(x_{i+k-k_{max}}, x_{i+k-k_{max}+1}, \ldots, x_{i+k}) \tag{10}$$

We experimentally verified that a value $k_{max} = 4$ gives a good approximation of the final distance.

4.4. String edit kernel

For classification purpose, the use of a SVM kernel is required. In our case, an image is represented as a set of P strings corresponding to vertical scans of the image. Thus, to compare two images $\mathcal{I}$ and $\mathcal{J}$, we propose to perform first a pairwise edit distance between string-pairs (X_i, Y_i) from the two images. Then, after summation, the results is plugged into a radial basis kernel such that:

$$K_{\text{ED}}(\mathcal{I}, \mathcal{J}) = e^{-\gamma \sum_{i=1}^{P} \text{ED}(X_i, Y_i)} \tag{11}$$

where ED stands for any edit distance, cED, sED or mED. Although this kernel is not positive definite, as shown in [30], it can still be used for SVM classification provided the parameter γ is adjusted so as the kernel matrix is positive for the current training data. Practically, this is achieved by optimizing the classification results by cross validation [30].

5. Experiments

In this section, the goal is to study the performance of structured matching on CNN features for various datasets. We aim at confirming that structured matching surpasses the performance of traditional rigid matching approaches. For this purpose, we consider two different baselines: 1) the rigid matching automatically learned by CNN using features from the first fully connected layer and 2) the rigid matching provided with a spatial pyramid pooling using features from the last convolutional layer. The results of our proposed structured matching are analyzed and compared to these baselines. In most state-of-the-art methods, several image crops sometimes at different resolutions are used to get improved results. To provide fair comparison, we also give refined results for our method by using either stride or scale reduction in the feature extraction process (see section 3.1).

In all experiments, we used Overfeat, the CNN features' extractor pretrained on ImageNet proposed by Sermanet et al. [26] team. Only the SVM classifier was learned from the target dataset.

5.1. Compared methods

Four different configurations are compared.

Baseline 1. OverFeat: We evaluate the performance of the OverFeat features generated after the first fully connected layer (layer 7), before the ReLU normalization. These features of size 4096 result from the combination of neurons from the last feature map. They are used to train SVM with a linear kernel, as often done in the literature. This baseline method is examined to compare such global image representation after a fully connected layer with our representations built on spatial pyramid pooling of the last convolutional feature map, as explained in section 3.2.

Baseline 2. SPP-Lin: We extract features from the last convolutional layer (layer 6). This produces a spatial 15×15 matrix of 1024 dimensional local feature vectors per input window position. These vectors are pooled (with max pooling) in a spatial pyramid and concatenated to get the final image representation vector, which is used to evaluate the classification through a linear SVM classifier. This baseline approach is examined to investigate the performance of spatial pyramid pooling with rigid matching in comparison with our string spatial pyramid pooling with approximate matching on the same feature maps.

SPP-ED: As in the previous approach, we extract features from the last convolutional layer (layer 6) and we pool them in the same spatial pyramid. Then we build the vertical strings and we use an edit kernel based SVM classifier as proposed in this article. We denote SPP-sED, SPP-cED and SPP-mED the methods associated to sED, cED or mED edit distances respectively. These distances rely all on the χ^2 distance for cost computation. Feature vectors are normalized in ℓ_1. In mED, the merge operation corresponds to a sum of successive symbols. In cED, ins/del costs are fixed to 0.5 which corresponds to the χ^2 distance between ℓ_1 normalized symbols and the nul symbol.

Refined SPP-ED: In this configuration, the purpose is to evaluate the full potential of our edit based distances. The best performing version according to the dataset is refined either by using resolution, scale augmentation or specific improvements which were already used in concurrent state-of-the-art methods.

5.2. Evaluation

The evaluation was conducted using four datasets: two scene datasets, 15Scenes and MIT indoor, and two object datasets, Pascal VOC 2007 and Caltech 101.

The 15Scenes dataset, introduced by [31] contains 4485 images in 15 classes from both outdoor and indoor scenes. Following the setting of [31], a 10-fold cross validation is used on random train/test subsets. 100 per class images are used for training and the rest for testing. The mean accuracy and the standard deviation over the 10-folds are reported for evaluation.

The MIT indoor dataset is a larger scene dataset, which contains $15,620$ images in 67 classes. The images represent cluttered indoor scenes with complex spatial layout. For experiments, we follow the original protocol of [32], which consists of 80 training images and 20 testing images per class. Performance is reported in terms of average classification accuracy across all categories.

The Pascal VOC 2007 dataset consists of 9963 images from 20 different object classes. Objects can have different scales, viewpoints and illuminations and several objects from different classes may appear in a single image. The evaluation follows the setup of VOC challenge by computing the mean average precision (MAP) over the given test set. The training is made using the concatenated training and validation sets.

The Caltech101 dataset [33] contains 9144 images in 102 object classes. In this dataset, a 10-fold cross validation is used on random train/test subsets. We randomly choose 15 or 30 images for training and up to 50 images per category for testing. The mean accuracy and the standard deviation over the 10-folds are reported for evaluation.

5.3. Results

5.3.1. Structural vs classical rigid representations

In this section, we first evaluate the potential of our structural representation over a classical static representation on the four datasets. For this purpose, we propose to compare the results obtained with the $Overfeat$ and SPP-Lin baselines and our SPP-xED variants. First sections of Tables 1, 2, 3 and 4 report the accuracies and MAP values of the different methods. For SPP-based methods, $W = 3$ and $H = 15$ were used for for 15Scenes, MIT indoor and Pascal VOC 2007 while $W = 4$ and $H = 15$ were applied for Caltech101.

Method	CNN Layer	Details	Accuracy
Overfeat	7/9	1 resize	86.99 ± 0.57
SPP-Lin	6/9	SPP pooling, 15×3, 7×1, 1×1	89.34 ± 0.38
SPP-sED	6/9		90.82 ± 0.39
SPP-cED	6/9		90.77 ± 0.25
SPP-mED	6/9		$\mathbf{91.06 \pm 0.43}$
Refined SPP-mED	6/9	SPP pooling micro features	$\mathbf{91.57 \pm 0.37}$
Koskela and Laaksonen [8]	7/8	1 crop	88.70 ± 0.30
Koskela and Laaksonen [8]	7/8	Multiscale (10)	$\mathbf{91.50 \pm 0.30}$
Koskela and Laaksonen [8]	7/8	Multiscale (10) fusion of 4 CNN	$92.01^{*} \pm 0.40$

Table 1: Classification accuracy for 15Scenes dataset. For multiscale methods, the number of patches used is indicated in parentheses. The last value indicated with * is not directly comparable because it combines four different CNN trained with two different versions of ImageNet.

The comparison of the two baselines $Overfeat$ and SPP-Lin already leads to a first interesting result. We indeed remark that $Overfeat$ baseline is systematically below SPP-Lin. The improvement provided by SPP-Lin is greater than 3% for the first three datasets and reaches 5.5% for Caltech101. This demonstrates that using the spatial pyramid pooling from the last convolutional layer is a better strategy than using combined features from the fully connected layers. This first intermediate result is of interest and has not been reported in the literature before.

Moreover, from the four tables, it is clear that the three structural SPP-ED methods clearly outperform the SPP-Lin baseline, and consequently, the $Overfeat$ one. The im-

Method	CNN Layer	Details	Accuracy
Overfeat	7/9	1 resize	60.41
SPP-Lin	6/9	SPP pooling 15×3, 7×1, 1×1	64.24
SPP-sED	6/9		**68.93**
SPP-cED	6/9		68.58
SPP-mED	6/9		68.61
Refined SPP-sED	6/9	SPP pooling dense features	**69.44**
Razavian et al. [34]	7/9	1 resize	58.4
Razavian et al. [34]	7/9	C+R Augm	**69.0**
Gong et al. [5]	7/8	1 resize	53.73
Gong et al. [5]	7/8	Multiscale (21)	68.88
Jie and Yan [7]	8/8	fine tune 1 resize	60.23
Jie and Yan [7]	8/8	fine tune Multiscale (55)	68.96

Table 2: Classification accuracy for MIT indoor dataset. For multiscale methods, the number of patches used is indicated in parentheses. C+R Augm. means that cropped and rotated samples are added for SVM training.

Method	CNN Layer	Details	MAP
Overfeat	7/9	1 resize	75.66
SPP-Lin	6/9	SPP pooling 15×3, 7×1, 1×1	78.87
SPP-sED	6/9		**80.36**
SPP-cED	6/9		79.95
SPP-mED	6/9		80.35
Refined SPP-sED	6/9	SPP pooling micro features F Augm	**82.47**
Razavian et al. [34]	7/9	1 resize	73.9
Razavian et al. [34]	7/9	C+R Augm	77.2
Chatfield et al. [3]	7/8	C+F Augm 10 crops	79.74
Chatfield et al. [3]	8/8	fine tune C+F Augm 10 crops	**82.42**
He et al. [6]	7/8	SPP net	80.10*

Table 3: Classification accuracy for Pascal VOC 2007 dataset. C+R Augm. means that cropped and rotated samples are added for training, C+F Augm and F Augm means that either cropped and flipped or flipped samples are added for fine tunning or SVM training. The last value indicated with * is not directly comparable because it uses a specific SPP pooling in the CNN.

Method	CNN Layer	Details	Accuracy 15 tr	Accuracy 30 tr
Overfeat	7/9	1 resize	80.03 ± 0.62	83.64 ± 0.58
SPP-Lin	6/9	SPP pooling 15×4, 7×2, 3×1, 1×1	85.26 ± 0.51	89.10 ± 0.44
SPP-sED	6/9		85.58 ± 0.68	89.48 ± 0.33
SPP-cED	6/9		$\mathbf{86.47} \pm \mathbf{0.57}$	$\mathbf{90.25} \pm \mathbf{0.36}$
SPP-mED	6/9		86.14 ± 0.58	90.07 ± 0.31
Refined SPP-cED	6/9	SPP pool. W=6 dense feat.	$\mathbf{87.49} \pm \mathbf{0.43}$	$\mathbf{91.07} \pm \mathbf{0.42}$
Zeiler and Fergus [2]	7/8	1 crop	$\mathbf{83.8} \pm \mathbf{0.5}$	86.50 ± 0.5
Chatfield et al. [3]	7/8	C+F Augm 10 crops	-	87.76 ± 0.66
Chatfield et al. [3]	8/8	fine tune C+F Augm 10 crops	-	$\mathbf{88.35} \pm \mathbf{0.66}$
He et al. [6]	6/8	SPP net $6 \times 3 \times 2 \times 1$	-	89.50^*
He et al. [6]	5/8		-	$91.44^* \pm 0.7$

Table 4: Classification accuracy for Caltech101 dataset. C+F Augm means that either cropped and flipped samples are added for SVM training. The two last results indicated with * are not directly comparable because they use a specific SPP pooling in the CNN.

provement is always greater than 1% and reaches about 5% for the MIT indoor dataset. This highlights the key result of this paper: using string representations and edit distances for image comparison significantly improves classification results in comparison with classical static representations and rigid matching approaches.

5.3.2. Comparison of SPP-based approaches

In this section, we propose to study more in details the behavior of the SPP-based methods varying the representation parameters and the type of dataset. Figure 6 reports the accuracies or MAP values in function of H for the different SPP based methods on each dataset. We recall that H is the maximum number of divisions along the vertical axis. Three values are studied: $H = 4$, $H = 8$ and $H = 15$. The number of divisions along the horizontal axis is fixed to $W = 3$ for 15Scenes, MIT indoor and Pascal VOC 2007 and to $W = 4$ for Caltech101 to better take into account the particular symmetry of images of this dataset.

Comparing SPP-based methods in figure 6, we again observe that structural representations and edit distances clearly outperform the $SPP - Lin$ baseline whatever the parameter H is. The improvement is always greater than 1% and reaches about 5% for MIT indoor dataset. Moreover, for all datasets, the rigid matching result is maximal for $H = 8$ and then decreases or remains constant for $H = 15$. Conversely, edit distances are less penalized by bad matches and the best results are obtained for $H = 15$. More precisely, we can notice that for 15Scenes and Caltech101 datasets where subjects are scaled and centred, the improvement is lower. Conversely, for MIT indoor or Pascal VOC datasets, edit distances are able to compensate objects variations in position or

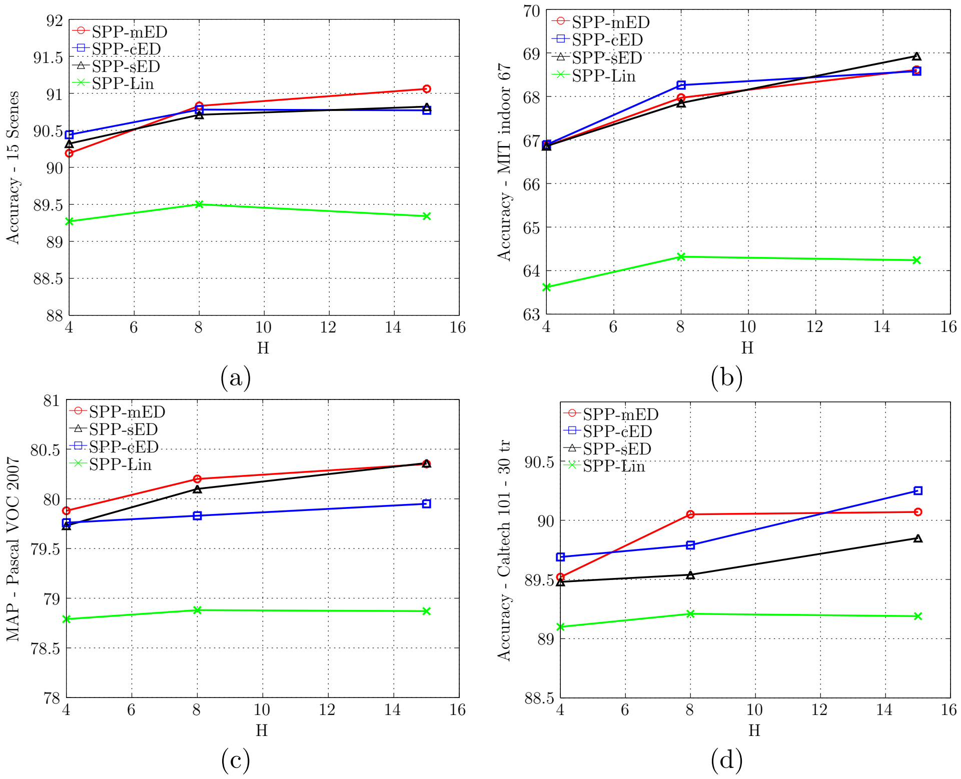

Figure 6: Comparison of structured and non-structured representations for different numbers of divisions H of the spatial pyramid pooling. The maximum number of divisions along horizontal axis is equal to $W = 3$ for 15Scenes (a), MIT indoor (b) and Pascal VOC 2007 (c) and $W = 4$ for Caltech101 (d).

scale thus provide higher improvement. Another important parameter for edit distances is the vertical organization of the scene. As shown by table 5, the best performing object classes of Pascal VOC 2007 have a significant vertical organization as ”sofa”, ”chair” and ”bus”.

Class	SPP-Lin	SPP-sED	SPP-cED	SPP-mED
aeroplane	92.4	**94.3**	93.5	94.0
bicycle	87.8	**89.2**	88.7	89.0
bird	88.4	87.9	**88.5**	88.1
boat	84.4	85.5	85.4	**85.6**
bottle	48.1	48.7	**49.2**	**49.2**
bus	75.8	**78.6**	77.6	78.5
car	89.0	90.6	90.3	**90.7**
cat	87.4	**88.2**	87.3	88.0
chair	62.8	**66.0**	65.1	65.5
cow	67.7	70.2	**70.8**	**70.8**
diningtable	73.0	74.1	**74.3**	**74.3**
dog	82.7	83.7	**83.9**	83.8
horse	89.8	90.8	90.6	**91.1**
motorbike	82.1	**83.9**	83.1	83.8
person	94.6	95.0	95.0	**95.1**
pottedplant	59.8	**61.4**	60.5	60.6
sheep	77.0	**78.6**	77.5	77.9
sofa	65.5	70.5	69.6	**71.0**
train	92.3	92.9	92.5	**93.1**
tvmonitor	76.7	**77.2**	75.5	77.0
Total	78.87	80.36	79.95	80.35

Table 5: By-class results for Pascal VOC 2007 dataset.

Finally, if we focus on the three versions of edit distance, we note different behaviors depending on both the edit distance version and the dataset. Considering all datasets, we can notice that mED is always the best or the second best performing method. Thus, the merge operation introduced in this distance is better than del/ins operations used either in cED or in the original sED proposed by Nguyen et al. [17]. We can also notice that cED performs better for highly structured images with a fixed spatial layout as in Caltech 101 dataset. Conversely, sED and mED better handle a variable spatial layout as in 15Scenes, MIT Indoor or Pascal VOC 2007 datasets. Considering by class results from Pascal VOC 2007 (table 5), we can confirm and refine these comments. Globally, cED performs poorly because of the variable spatial layout. However, it performs slightly better for ”bird” and ”dog” classes for which the subject is more often centered in the image. In average mED and sED perform almost equally but we notice that again, mED is always the best or second best performing method. Lastly, we notice that mED seems to perform better for natural low structured scenes as ”boat”, ”horse” or ”cow” whereas sED seems to perform better for more structured scenes as ”aeroplane”, ”bicycle”, ”chair” or ”pottedplant”.

5.3.3. Comparison to the state-of-the-art

Most state-of-the-art results are obtained using specific refinements either in the CNN pre-training or in the classification step. To illustrate the full potential of SPP-ED and provide a fair comparison, we have also computed the best SP-ED result using denser features obtained either by reducing the stride alone (indicated by ”dense features” in the tables) or by reducing the stride and the scale of features (indicated by ”micro features”).

The choice of using dense or micro features depends on the dataset. Indeed, CNN features are trained using ImageNet where images contain a single roughly centered object filling most of the image. Then, for datasets containing images having such roughly centered objects, as MIT indoor or Caltech 101, it is better to use the stride reduction alone to keep the size of features unchanged. Conversely, when objects are much smaller and can appear at various locations as in 15Scenes and Pascal VOC 2007, the scale reduction is a better strategy. For Pascal VOC 2007, we have also augmented the training set by adding flipped images (indicated by "F Augm"). The corresponding refined results are given in the last SPP-ED line of tables 1, 2, 3 and 4.

In the last part of tables, we have reported state-of-the-art results obtained with CNN-based methods for each dataset. All methods are based on the training of a CNN on ImageNet. Most of them use a SVM classifier to learn the classes of the target dataset. Some methods use data augmentation (adding either cropped (C), rotated (R) or flipped (F) samples) to learn the target dataset (indicated by "Augm" in the tables). Additionally the authors of [7, 3] use a fine tuning of the CNN combined with a softmax classifier instead of the SVM classifier (indicated by "fine tune"). Lastly, the authors of [8, 6] use a different CNN architecture or dataset for training, Koskela and Laaksonen [8] also combine four different CNNs to achieve their best result.

15Scenes (table 1): This dataset enables the comparison of our results with those presented in [8]. As seen in section 2.2, the latter method uses a multiscale approach similar to [5] which requires to decompose an image into 10 multiscale crops, each of them being further processed by the CNN and combined in the final feature vector. We note that our improved SPP-ED method outperforms the single crop and multiscale cases with 91.57% of accuracy. The last result provided by Koskela and Laaksonen [8] (indicated with *) is a little higher than ours (0.4%). But note that this result is not directly comparable to ours. Indeed, the CNN features they used are different, they are obtained from four different CNN outputs computed through two different CNN architectures, each of them being trained with two different versions of ImageNet datasets (2010 and 2012 versions). Note that our approach is still competitive with this approach and could certainly be further improved by applying comparable multi-CNN strategies.

MIT indoor (table 2): For this dataset, our method outperforms all state-of-the-art methods. Even our standard results outperform single and multiscale approaches proposed in [5, 7]. Our improved SPP-ED result outperforms the result reported in [34] where the SVM training set was augmented by adding rotated image crops.

Pascal VOC 2007 (table 3): Again, our method outperforms all concurrent methods. Our standard SPP-sED and SPP-mED outperform all methods combining pre-trained CNN and SVM. In particular, they outperform the SPP-Net of He et al. [6] which is specifically designed for spatial pyramid pooling and requires the design and the training of a specific CNN architecture. Note that our refined SPP-ED outperforms the outstanding result obtained by Chatfield et al. [3] which requires a CNN fine tuning training with a specific ranking hinge loss, flip images for data augmentation and 10 passes through the CNN to deal with geometric variations. In comparison, our refined method uses standard CNN features computed in one single pass (with double scale input image) and an edit kernel SVM for training (with additional flipped images for training).

Caltech 101 (table 4): To take into account the strong spatial structure of this dataset, in our refined results, we increased the number of bands to $W = 6$ together with dense features. In table 4, we notice that our standard and refined SPP-ED outperform all methods that use the output of full connected layers as features. In particular, they outperform results from Chatfield et al. [3] with data augmentation (flip images in the training) and fine tuning. They also outperform the SPP-Net of He et al. [6] when the features are extracted at the output of the first fully connected layer (layer 6/8). However, if features are extracted at the output of the last convolution layer and pooled in a spatial pyramid, He et al. [6] method is slightly better than our structural approach (for 30 train images per class). Note that this strategy is exactly what we propose. The only difference is that their CNN is specifically trained with a spatial pyramid pooling step whereas ours is trained with a normal pooling. Therefore, it is not directly comparable to ours and it is very likely that we can achieve better by using SPP-Net features with our edit distance. Finally, we note that even with $Overfeat$ features, our result for 30 train images remains competitive (only less than 0.4% lower) and, to the best of our knowledge, our result for 15 training images per class (87.48%), is the best reported until now.

In conclusion, these results show that our approach outperforms all state-of-the-art methods which use a comparable CNN as feature extractor either combined with a SVM classifier or fine tuned on the target dataset. Even recent improvements using multiscale approaches are less successful. The SPP-Net from He et al. [6] performs slightly better on Caltech101 but really worse on Pascal VOC 2007. Moreover, this method is not directly comparable to ours because it uses a CNN specifically trained with spatial pyramid pooling.

5.3.4. Computational considerations

To quantify more precisely the computational benefit of our approach, we have reported in table 6 its execution time compared with some concurrent methods. Our reference computer is a standard PC with a 16 cores 3.7 Ghz CPU (no GPU used) and we consider a classification problem with $n_t = 1500$ train images. Processing times are estimated from the time t_f of one forward pass through the CNN, the number of crops n_c used for feature extraction, and times t_{train} and t_{test} taken respectively for training the classifier or testing an image. The total training time T_{train} and the per image testing time T_{test} are given by:

$$\begin{array}{rcl} T_{train} & = & n_t\, n_c\, t_f + t_{train} \\ T_{test} & = & n_c\, t_f + t_{test} \end{array} \tag{12}$$

For all methods, we took the value $t_f = 2.5s$ measured with our $Overfeat$ implementation. For our structural methods, times t_{train} and t_{test} times are depending on edit kernel processing times. They are measured using our standard parameters with our raw C++ implementation. For other SVM based methods, they are neglected. For methods using a fine tuned CNN we have reported *GPU* in the training column to indicate that a GPU architecture is needed. Note that the initial CNN training time is not taken into account in the purpose of using a pre-trained CNN.

It is clear from table 6 that our method has one of the shortest processing time thanks to the single pass through the CNN. Both training and testing can be processed efficiently with a standard PC without using any GPU. It is a little slower than SPP-Net

Method	n_c	Training (min)		Testing (s/im)	
		t_{train}	Total	t_{test}	Total
SPP-sED and SPP-cED	1	6.4	**68.9**	0.5	**3**
SPP-mED	1	132	194.5	11	13.5
Koskela et al.[8], Multiscale	10	-	625	-	25
Koskela et al.[8], 4 CNN	40	-	2500	-	100
Razavian et al. [34], C+R Augm	10	-	625	-	25
Gong et al. [5], Multiscale	21	-	1312.5	-	52.5
Chatfield et al. [3], C+F Augm	10	-	625	-	25
Chatfield et al. [3], fine tune	10	GPU	GPU	-	25
He et al. [6], SPP-Net	1	-	**62.5**	-	**2.5**

Table 6: Estimated processing time on a standard PC with a 16 cores 3.7 Ghz CPU (no GPU used).

because of the time taken to compute the edit kernel. However, let's note that using SPP-Net as a pre-trained CNN is not as flexible as using $Overfeat$ because in SPP-Net, the parameters of SPP pooling are fixed into the network, and may not be optimal for all datasets. In particular, we have previously noticed that SPP-Net performs bad for Pascal VOC 2007.

In conclusion, considering all the datasets, our method is the best performing one with the shortest execution time. One great advantage is that both training and testing can be achieved quickly without requiring a specific GPU architecture.

6. Conclusion

Approximate matching is an important area of pattern recognition research. In this paper, we demonstrate that the powerful edit distance can be very effective for image classification if we carefully choose the edit operations and their associated costs. We have formulated two edit distance variants, cED and mED. Combined with the recent dominant CNN features and the spatial pyramid pooling, it is clear that using our structural approaches perform better than standard static approaches. Not only they take into account the statistical properties of regions, but also their spatial dependencies. It makes them more robust to image transformations, and therefore, improves image classification. The gain is particularly significant in datasets such as Pascal VOC 2007 (around 2%) where objects are particularly misaligned or MIT indoor (around 5%) where contents are very complex and vary in scale and position. Moreover, our method is shown to be competitive with all state-of-the-art approaches of the literature, with one of the shortest execution time. Additionally, thanks to the use of a pre-trained CNN and a single feature extraction step, both training and testing can be achieved quickly in a standard CPU without the need of handling the complex training of the whole network.

There are different ways to extend this work. First of all, mED only works with the merge and substitution operations. Obviously, it could be interesting to combine them with insertion, deletion operations that are not used anymore. Second, ideas used for image similarity appear adequate to other application tasks, especially near duplicate image detection, where it is required to measure, close but slightly different visual content. Third, it would seem pertinent to integrate our edit operations into graph representations to better take into account relationships between neighboring strings. These issues are left for future research.

References

[1] A. Krizhevsky, I. Sutskever, G. E. Hinton, Imagenet classification with deep convolutional neural networks, in: Advances in Neural Information Processing Systems, pp. 1097–1105.

[2] M. D. Zeiler, R. Fergus, Visualizing and understanding convolutional networks, in: Computer Vision–ECCV 2014, Springer International Publishing, 2014, pp. 818–833.

[3] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the devil in the details: Delving deep into convolutional nets, arXiv preprint arXiv:1405.3531 (2014).

[4] M. Cimpoi, S. Maji, A. Vedaldi, Deep convolutional filter banks for texture recognition and segmentation, arXiv preprint arXiv:1411.6836 (2014).

[5] Y. Gong, L. Wang, R. Guo, S. Lazebnik, Multi-scale orderless pooling of deep convolutional activation features, arXiv preprint arXiv:1403.1840 (2014).

[6] K. He, X. Zhang, S. Ren, J. Sun, Spatial pyramid pooling in deep convolutional networks for visual recognition, in: Computer Vision–ECCV 2014, Springer International Publishing, 2014, pp. 346–361.

[7] Z. Jie, S. Yan, Robust scene classification with crosslevel LLC coding on CNN features, in: Asian Conference on Computer Vision (ACCV), 2014.

[8] M. Koskela, J. Laaksonen, Convolutional network features for scene recognition, in: Proceedings of the ACM International Conference on Multimedia, ACM, New-York, pp. 1169–1172.

[9] G. Seni, V. Kripasundar, R. K. Srihari, Generalizing edit distance to incorporate domain information: Handwritten text recognition as a case study, Pattern Recognition 29 (1996) 405–414.

[10] M. Christodoulakis, G. Brey, Edit distance with combinations and splits and its applications in OCR name matching, International Journal of Foundations of Computer Science 20 (2009) 1047–1068.

[11] K. Khurshid, C. Faure, N. Vincent, A novel approach for word spotting using merge-split edit distance, in: 13th International Conference on Computer Analysis of Images and Patterns, CAIP 2009, Germany, volume 1, pp. 213–220.

[12] P. N. Klein, T. B. Sebastian, B. B. Kimia, Shape matching using edit-distance: an implementation, in: Proceedings of the Twelfth Annual ACM-SIAM Symposium on Discrete Algorithms, Society for Industrial and Applied Mathematics, Philadelphia, pp. 781–790.

[13] A. Muzameel, Rashmi, M. Shravya, N. Sindhu, S. Supritha, Shape classification using shape context and dynamic programming, International Journal of Scientific & Engineering Research 4 (2013).

[14] J. Ros, C. Laurent, J.-M. Jolion, I. Simand, Comparing string representations and distances in a natural images classification task, in: GbR'05, 5th IAPR-TC-15 Workshop on Graph-Based Representations, pp. 72–81.

[15] C. Barat, C. Ducottet, E. Fromont, A.-C. Legrand, M. Sebban, Weighted symbols-based edit distance for string-structured image classification, in: Machine Learning and Knowledge Discovery in Databases, Springer Berlin Heidelberg, 2010, pp. 72–86.

[16] M.-C. Yeh, K.-T. Cheng, Fast visual retrieval using accelerated sequence matching, Multimedia, IEEE Transactions on 13 (2011) 320–329.

[17] H.-T. Nguyen, C. Barat, C. Ducottet, Approximate image matching using strings of bag-of-visual words representation, in: International Conference on Computer Vision Theory and Applications (VISAPP 2014), Lisbon, Portugal, pp. 345–353.

[18] L. Ballan, M. Bertini, A. Del Bimbo, G. Serra, Video event classification using string kernels, Multimedia Tools and Applications 48 (2010) 69–87.

[19] K. Chatfield, V. Lempitsky, A. Vedaldi, A. Zisserman, The devil is in the details: an evaluation of recent feature encoding methods, in: Proceedings of the British Machine Vision Conference (BMVC), pp. 76.1–76.12.

[20] C. Iovan, D. Picard, N. Thome, M. Cord, Classification of urban scenes from geo-referenced images in urban street-view context, in: 11th International Conference on Machine Learning and Applications (ICMLA), volume 2, pp. 339–344.

[21] H. E. Tasli, R. Sicre, T. Gevers, Geometry-constrained spatial pyramid adaptation for image classification, in: International Conference on Image Processing (ICIP), pp. 1051–1055.

[22] G. Sharma, F. Jurie, Learning discriminative spatial representation for image classification, in: BMVC 2011 - British Machine Vision Conference, BMVA Press, Dundee, United Kingdom, 2011, pp. 1–11.

[23] X. Li, Y. Song, Y. Lu, Q. Tian, Spatial pooling for transformation invariant image representation, in: Proceedings of the 19th ACM international conference on Multimedia, ACM, New-York, 2011, pp. 1509–1512.

[24] Y. Cao, C. Wang, Z. Li, L. Zhang, L. Zhang, Spatial-bag-of-features, in: 2010 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 3352–3359.

[25] Y. Jiang, J. Yuan, G. Yu, Randomized spatial partition for scene recognition, in: Computer Vision–ECCV 2012, volume 7573, Springer Berlin Heidelberg, 2012, pp. 730–743.

[26] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, Y. LeCun, Overfeat: Integrated recognition, localization and detection using convolutional networks, arXiv preprint arXiv:1312.6229 (2013).

[27] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, T. Darrell, Caffe: Convolutional architecture for fast feature embedding, arXiv preprint arXiv:1408.5093 (2014).

[28] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, A. C. Berg, L. Fei-Fei, ImageNet Large Scale Visual Recognition Challenge, arXiv preprint arXiv:1409.0575 (2014).

[29] M. Christodoulakis, G. Brey, Edit distance with single-symbol combinations and splits, in: J. Holub, J. Žďárek (Eds.), Proceedings of the Prague Stringology Conference 2008, Czech Technical University in Prague, Czech Republic, pp. 208–217.

[30] H. Li, T. Jiang, A class of edit kernels for svms to predict translation initiation sites in eukaryotic mrnas, Journal of Computational Biology 12 (2005) 702–718.

[31] S. Lazebnik, C. Schmid, J. Ponce, Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories, in: 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pp. 2169–2178.

[32] A. Quattoni, A. Torralba, Recognizing indoor scenes, 2013 IEEE Conference on Computer Vision and Pattern Recognition (2009) 413–420.

[33] L. Fei-Fei, R. Fergus, P. Perona, Learning generative visual models from few training examples: an incremental Bayesian approach tested on 101 object categories, in: IEEE CVPR Workshop of Generative Model Based Vision (WGMBV).

[34] A. S. Razavian, H. Azizpour, J. Sullivan, S. Carlsson, CNN features off-the-shelf: an astounding baseline for recognition, arXiv preprint arXiv:1403.6382 (2014).