Supervised Mid-Level Features for Word Image Representation

Albert Gordo
Computer Vision Group
Xerox Research Centre Europe
albert.gordo@xrce.xerox.com

Abstract

This paper addresses the problem of learning word image representations: given the cropped image of a word, we are interested in finding a descriptive, robust, and compact fixed-length representation. Machine learning techniques can then be supplied with these representations to produce models useful for word retrieval or recognition tasks. Although many works have focused on the machine learning aspect once a global representation has been produced, little work has been devoted to the construction of those base image representations: most works use standard coding and aggregation techniques directly on top of standard computer vision features such as SIFT or HOG.

We propose to learn local mid-level features suitable for building word image representations. These features are learnt by leveraging character bounding box annotations on a small set of training images. However, contrary to other approaches that use character bounding box information, our approach does not rely on detecting the individual characters explicitly at testing time. Our local mid-level features can then be aggregated to produce a global word image signature. When pairing these features with the recent word attributes framework of [4], we obtain results comparable with or better than the state-of-the-art on matching and recognition tasks using global descriptors of only 96 *dimensions.*

1. Introduction

In recent years there has been an increasing interest in tasks related to text understanding in natural scenes, and, amongst them, in word recognition: given a cropped image of a word, one is interested in obtaining its transcription. The most popular approaches to address this task involve detecting and localizing individual characters in the word image and using that information to infer the contents of the word, using for example conditional random fields and language priors [37, 20, 21, 5, 23]. As shown by Bissacco *et al.* [5], such approaches that learn directly from the annotated individual characters can obtain impressive accuracy if large volumes of training data are available. However, these approaches are not exempt from problems. First, to obtain a high accuracy, one needs to annotate very large amounts of words (in the order of millions) with character bounding boxes for training purposes, as done by Bissaco *et al.* [5]. When limited training data is available, the recognition accuracy of these approaches is much lower [20, 21]. Then, at testing time, one needs to localize the individual characters of the word image, which is slow and error prone. Also, these approaches do not lead to a final signature of the word image that can be used for other tasks such as word image matching and retrieval.

Rather than localizing and classifying the individual characters in a word, a new trend in word image recognition and retrieval has been to describe word images with global representations using standard computer vision features (*e.g.* HOG [8], or SIFT [18] features aggregated with bags of words [7] or Fisher vector [25] encodings) and apply different frameworks and machine learning techniques (such as using attribute representations, metric learning, or exemplar SVMs) on top of these global representations to learn models to perform tasks such as recognition, retrieval, or spotting [27, 30, 2, 1, 3, 29, 4]. The global approaches have important advantages: they do not require that words be annotated with character bounding boxes for training and they do not require that the characters forming a word be explicitly localized at testing time. They can also produce compact signatures which are faster to compute, store and index, or compare, while still obtaining very competitive results in many tasks. The use of off-the-shelf computer vision features and machine learning techniques also makes them very attractive. Yet, one may argue that not using character bounding box annotations during training, although very convenient, may be a limiting factor for their accuracy.

The main contribution of this paper is an approach to construct a global word image representation that unites the best properties of both main trends by leveraging character bounding boxes information at training time. This is achieved by learning mid-level local features that are correlated with the characters in which they tend to appear. We use a small external dataset annotated at the character level to learn how to transform small groups of locally aggregated low-level features into mid-level semantic features suitable for text, and then we encode and aggregate these mid-level features into a global representation. This unites advantages of both paradigms: a compact signature that does not require localizing characters explicitly at test time, but still exploits information about annotated characters. Although some works have already used supervised information (in the form of text transcriptions) to project a global image representation into a more semantic space [29, 3], to the best of our knowledge, no other approach that constructs global image representations has leveraged character bounding box information at training time to do so.

We test our approach on two public benchmarks of scene text showing that constructing representations using mid-level features yields large improvements over constructing them using SIFT features directly. When pairing these mid-level features with the recent attributes framework of [3, 4], we significantly outperform the state-of-the-art in word image retrieval (using both images or strings as queries), and obtain results comparable to or better than Google's PhotoOCR [5] in recognition tasks using a tiny fraction of training data: we use less than $5,000$ training words annotated at the character level, while [5] uses several millions.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 describes our method. Section 4 deals with the experimental evaluation. Finally, Section 5 concludes the paper.

2. Related Work

We now review those works which are most related to our approach.

Scene text recognition. Most works focusing on scene-text target only the problem of recognition, *i.e.*, given the image of a word, the goal is to produce its transcription. In such a case, a priori, there are no clear advantages with producing a global image representation. Instead, most methods aim at localizing and classifying characters or character regions inside the image and using this information to infer the transcription. For example, Mishra *et al.* [20, 21] propose to detect characters using a sliding window model and produce a transcription using a conditional random field model with language priors. Neumann and Matas [22, 23] use Extreme Regions or Strokes to localize and describe characters. Words are then recognized using a commercial OCR or by maximizing the characters' probability. The recent [39] uses a mid-level representation of strokes to produce more semantic descriptions of characters, that are then classified using random forests. PhotoOCR [5] learns a character classifier using a deep architecture with millions of annotated training characters, and produces very accurate transcriptions at the cost of requiring vast amounts of annotated data. In a different line, Jaderberg *et al.* [12] propose to learn the classification task directly from the image without localizing the characters using deep convolutional neural networks. Although the results on some tasks are impressive, this approach also requires millions of annotated training samples to perform well.

Global representations for word images. More directly related to our work are global representations. Producing image signatures opens the door to other tasks such as word retrieval, as well as easing tasks such as storing and indexing word images. Rusiñol *et al.* [30] construct a bag of words over SIFT descriptors to encode word images, and use it to perform segmentation-free spotting on handwritten documents. In [1], this framework is enriched using textual information. In [2], Exemplar SVMs and HOG descriptors are also used to perform segmentation-free spotting on documents. Goel *et al.* [9] propose to recognize scene-text images by synthesizing a dataset of annotated images and finding the nearest neighbor in that dataset. Rodriguez and Perronnin propose in [29] a label embedding approach that puts word images (represented with Fisher vectors) and text strings in the same vectorial space. Similarly, Almazán *et al.* [3, 4] propose a word attributes framework that can perform both retrieval and recognition in a low dimensional space. Although they perform well in retrieval tasks, they are outperformed in recognition by methods that exploit annotated character information, such as PhotoOCR [5].

Learning mid-level features. Our work is also related to the use of mid-level features (*e.g.* [6, 15, 39]), where "blocks" that contain some basic semantic information are discovered, learned, and/or defined. The use of mid-level features has been shown to produce large improvements in different tasks. Of those works, the most related to ours is the work of Yao *et al.* [39], which learns Strokelets, a mid-level representation that can be understood as "parts" of characters. These are then used to represent characters in a more semantic way. The main distinctions between the Strokelets and our work are that i), we exploit supervised information to learn a more semantic representation, and ii), we do not explicitly classify the character blocks, and instead use this semantic representation to construct a high-level word image signature. We show that our approach leads to significantly better results than the Strokelets [39]. The embedding approaches of [29, 3] could also be understood as producing supervised mid-level features, but do not use character bounding box information to do so. To learn the semantic space, we perform supervised dimensionality reduction of local Fisher vectors that are then encoded and aggregated into a global Fisher vector. This could be seen as

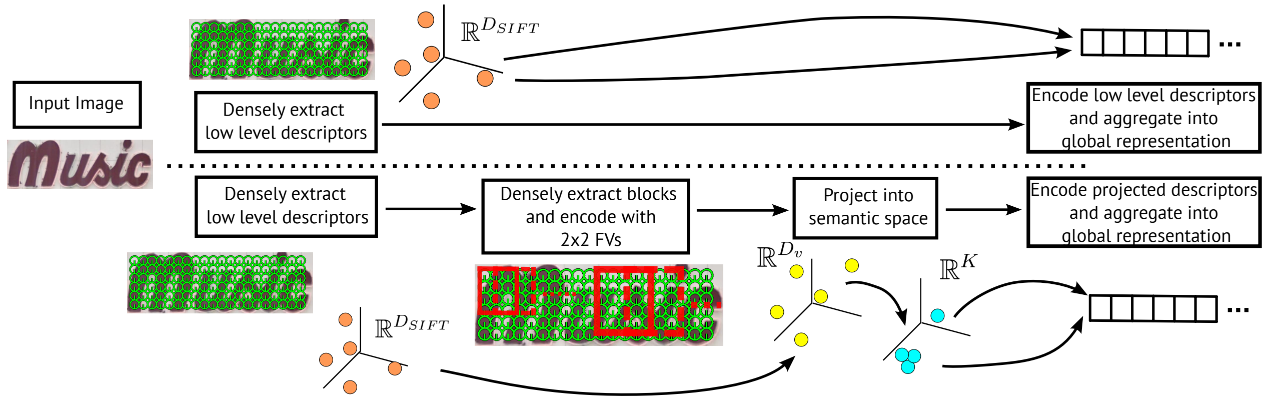
Input Image
Densely extract low level descriptors
Encode low level descriptors and aggregate into global representation
Densely extract low level descriptors
Densely extract blocks and encode with 2x2 FVs
Project into semantic space
Encode projected descriptors and aggregate into global representation
$\mathbb{R}^{D_{SIFT}}$
$\mathbb{R}^{D_v}$
$\mathbb{R}^{K}$

Figure 1. **Top.** Standard word image description flow: low-level descriptors (*e.g.* SIFT) are first densely extracted and then encoded and aggregated into a global representation using *e.g.* Fisher vectors (FV). Spatial pyramids may be used to add some weak geometry. **Bottom.** Proposed approach: we first densely extract low-level descriptors. Then we densely extract blocks of different sizes, and represent each block by aggregating the low-level descriptors it contains into a local FV with a 2×2 spatial pyramid. These local FV representations are then projected into a mid-level space correlated with characters. Finally, these mid-level features are aggregated into a global FV.

a deep Fisher network for image recognition [34], also used very recently for action recognition [24]. The main difference is that in [34, 24] the supervised dimensionality reduction step is learned using the image labels, the same labels that will be used for the final classification step. In our case, the goal is to transfer knowledge from the individual character bounding boxes, which are only annotated in the training set, to produce features that are correlated with characters, and exploit this information in the target datasets, where these bounding boxes are not available. This would be similar to learning the extra layer of the deep Fisher network using the labels of the bounding boxes of the objects, instead of using the whole image label as [34] does. Although the resulting architectures are similar, the motivation behind them is very different. In that sense, our work can also be related to works on learning with privileged information [35, 33, 10], where the information available at training time to construct the representations (in our case, character bounding boxes) is not available at test time.

3. Mid-Level Features for Word Images

A standard approach to construct a global word image representation is to i) extract low-level descriptors, ii) encode the descriptors, and iii) aggregate them into a global representation, potentially with spatial pyramids [17] to add some weak geometry. This representation can then be used as input for different learning approaches, as done *e.g.* in [30, 3, 29]. Figure 1 (top) illustrates this process.

In our proposed method, we aim at constructing global representations based on semantic mid-level features instead of using the low-level descriptors directly. The goal is to produce features that might not be good enough to predict the individual characters directly, but are more correlated with the individual characters than SIFT or other local descriptors.

This is achieved not by finding and classifying characteristic blocks as in [39], but by projecting *all* possible image blocks into a lower-dimensional space where our mid-level visual features and the characters are more correlated. By projecting all possible blocks in an image, one obtains a set of mid-level descriptors that can then be aggregated into a global image representation. The process is illustrated in Figure 1 (bottom).

In what follows, we first describe the learning process and describe how to project the image blocks into the semantic space correlated with the characters (Section 3.1). We then describe how to extract these mid-level features from a new image, and how to combine them with the word attributes framework [4] to obtain very compact, discriminative word representations (Section 3.2).

3.1. Learning Mid-Level Features

Let us assume that, at training time, one has access to a set of N word images and their respective annotations. Let $\mathcal{I}$ be one word image containing $|\mathcal{I}|$ characters, and let its annotation be a list of character bounding boxes $char\_bb$ and character labels $char\_y$, $\mathcal{C}_{\mathcal{I}} = \{(char\_bb_i, char\_y_i), i = 1 \ldots |\mathcal{I}|\}$. Each character label $char\_y_i$ belongs to one of the 62 characters in the following alphabet: $\Sigma = \{\mathtt{A}, \ldots, \mathtt{Z}, \mathtt{a}, \ldots, \mathtt{z}, \mathtt{0}, \ldots, \mathtt{9}\}$.

Let us denote with $block\_bb$ a square block in the image represented as a bounding box. For training purposes, let us randomly sample from every image S blocks of different sizes (*e.g.* 32×32 pixels, 48×48, etc) at different positions. Some of these blocks may contain only background, but most of them will contain parts of a character or parts of two consecutive characters.

We then describe these blocks using two modalities. The first one is based on visual features. The second one is based on character annotations. This second modality is more discriminative but is only available during training.

Then, one can learn a mapping between the visual features and the character annotation space. Once this mapping has been learned, at testing time one can extract all possible blocks in an image, represent them first with low-level visual features, and then map them into this semantic space to obtain mid-level features, as seen in Figure 1 bottom.

Block visual features. To encode the visual aspect of the blocks we use Fisher vectors (FV) [25] over SIFT descriptors [18], with a 2×2 spatial pyramid [17] to add some basic structure to the block. The descriptors are then ℓ_2-normalized. It has been shown that power- and ℓ_2- normalizing the Fisher vector usually leads to more discriminative representations [28]. In this case, however, we only apply an ℓ_2 normalization: the non-linearity introduced by the power-normalization would make it more difficult to efficiently aggregate the block statistics (see Section 3.2 and appendix for details). Fortunately, power normalization is most useful when using large vocabularies [26]. With small vocabularies (8 Gaussians in our case) the improvements due to power normalization are very limited.

Finally, stacking the descriptors of all the sampled blocks of all the training images leads to a matrix $\mathbf{X}$ of size $NS \times D_v$, where we denote with D_v the dimensionality of these visual representations. In our experiments we will use visual descriptors of $D_v = 4,096$ dimensions.

Block annotations. The second view is based on the annotation and contains information about the overlaps between the blocks and the characters in the word image. The goal is to encode with which characters the sampled blocks tend to overlap. In particular, we are interested in encoding what percentage of the character regions are covered by the blocks. As a first approach, we construct a D_a-dimensional label y for each sampled block, with $D_a = |\Sigma| = 62$. Given a block, its label y is encoded as follows: for each character Σ_d in the alphabet Σ, we assign at dimension d of the label vector the normalized overlap between the bounding box of the block and the bounding boxes of the characters of the word whose label $char\_y_i$ equals Σ_d. Since the word may have repeated characters overlapping with the same block, we take the maximum overlap:

$$y^d = \max_{(char\_bb_i, char\_y_i) \in \mathcal{C}_\mathcal{I}} \delta_{i,d} \frac{|\text{Intersection}(block\_bb, char\_bb_i)|}{|char\_bb_i|}, \tag{1}$$

where $|\cdot|$ represents the area of the region, and $\delta_{i,d}$ equals 1 if $char\_y_i = \Sigma_d$ and 0 otherwise. Figure 2 illustrates this with an annotated image and a sampled block with its corresponding computed label y.

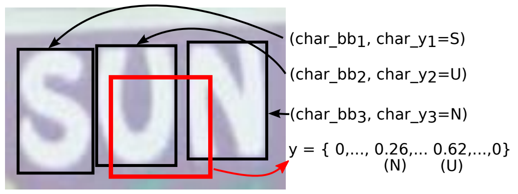

Figure 2. Example of annotated word and a sampled block with its label. The characters of the word contain bounding boxes (in black) and label annotations ('S','U','N'). The image also shows a sampled block (in red) with its respective computed label y. All the elements of y are set to 0 except the ones corresponding to the 'U' and 'N' characters.

As with the first view, it is possible to encode all the block labels of all the sampled blocks of all the training images in a matrix $\mathbf{Y} \in \mathbb{R}^{NS \times D_a}$.

Adding character spatial information. The labels that we introduced present a shortcoming: although they correlate blocks with characters, they do not encode *which part of the character* they are correlated with. This is important, since this type of information can have great discriminative power. To address this problem, we split the ground-truth bounding box character annotations in $R \times R$ character regions (CR). One can consider a 1×1 region (*i.e.*, the whole character), but also 2×2 regions, 3×3 regions, etc. Then, the labels y encode the overlap of the block with each region of the character independently, leading to a label of size $\mathbb{R}^{R^2 \times D_a}$:

$$y^{d,r} = \max_{(char\_bb_i, char\_y_i) \in \mathcal{C}_\mathcal{I}} \delta_{i,d} \frac{|\text{Intersection}(block\_bb, char\_bb_{ir})|}{|char\_bb_{ir}|}, \tag{2}$$

with $r = 1 \dots R^2$, and where $char\_bb_{ir}$ denotes the r-th region of bounding box $char\_bb_i$. This label is flattened into $\mathbb{R}^{R^2 D_a}$ dimensions. One may also consider computing labels at different character levels and concatenating the results in a final label before learning the projection, but we did not find this to improve the results.

Learning the mid-level space. To find a mapping between $\mathbf{X}$ and $\mathbf{Y}$ one can use, for example, canonical correlation analysis (CCA) [11]. CCA finds two projection matrices $\mathbf{U} \in \mathbb{R}^{D_v \times K}$ and $\mathbf{V} \in \mathbb{R}^{D_a \times K}$ such that the correlation in the projected space is maximized. To find $\mathbf{U}$, one only needs to solve a generalized eigenvalue problem. To avoid numerical instabilities, a small regularization parameter η is typically used. When D_v and D_a are small as in our case, this is very fast, and needs to be solved only once, offline. The K leading eigenvectors of the solution constitute the columns of matrix $\mathbf{U}$. One can easily choose the output dimensionality K by keeping only a certain number of eigenvectors. Analogously, one can solve a related eigenvalue problem to find $\mathbf{V}$, although we do not use it here since we will not have access to character bounding

Algorithm 1 Learn embedding

```
Input: Training images $\mathcal{I} = \{\mathcal{I}_1, \dots, \mathcal{I}_N\}$ and their annotations. Output dimensionality $K$. Number of sampled blocks per image $S$. Regularization parameter $\eta$.
Output: Embedding matrix $\mathbf{U}$.
  for $\mathcal{I} \in \mathcal{I}$ do
    Densely extract SIFT descriptors.
    Sample $S$ blocks at different sizes and positions.
    for each block do
      i) Aggregate the SIFT descriptors inside the block bounding box into a FV with a $2 \times 2$ spatial pyramid. Add as a new row of $\mathbf{X}$.
      ii) Construct label of block using Equation (1) or Equation (2). Add as a new row of $\mathbf{Y}$.
    end for
  end for
  Compute $\mathbf{U}$ using CCA. The $k$-th column of $\mathbf{U}$ is the eigenvector solution of the generalized eigenvalue problem $\mathbf{X}^T\mathbf{Y}(\mathbf{Y}^T\mathbf{Y} + \eta\mathbf{I})^{-1}\mathbf{Y}^T\mathbf{X}u_k = \lambda(\mathbf{X}^T\mathbf{X} + \eta I)u_k$ associated with the $k$-th largest eigenvalue.
End
```

boxes at test time. This process allows one to learn a matrix $\mathbf{U}$ that projects $\mathbf{X}$ into a subspace of K dimensions that is correlated with the labels $\mathbf{Y}$. This is depicted in Algorithm 1.

Discussion. In the described system, the block labels encode which percentage of the characters or the character regions are covered by the blocks. However, this is only one possible way to encode the labels. Other options could include *e.g.* encoding which percentage of the block is covered, the intersection over union, or working at the pixel level instead of the region level. Any representation that relates the visual block with the character annotation could be considered. We found that the proposed approach worked well in practice and deemed the search for the optimum representation out of the scope of this work.

3.2. Representing Word Images with Mid-Level Features

Once the matrix $\mathbf{U}$ has been learned, one can use it to compute the set of mid-level features of a new word image. A naive approach would involve extracting all possible blocks, encoding them with Fisher vectors, and projecting them with $\mathbf{U}$ into the mid-level space. This is shown in Algorithm 2.

Fast computation. In practice, however, applying this algorithm directly would be slow, requiring one to encode all possible blocks independently. Instead, we propose an approach to compute these descriptors *exactly* in an efficient manner. The key aspect is to notice that if the block Fisher vectors are not power-normalized (as in our case), all operations involved in the computation of the mid-level features are linear: one can compute the mid-level representation of each individual SIFT descriptor, and the mid-level representation of a block is exactly the ℓ_2-normalized sum of the mid-level representations of the SIFT descriptors contained in that block. Therefore, one can i) extract all SIFT descriptors of the image, ii) compute their mid-level representation, and iii) aggregate into an integral representation. Given that structure, one can compute the mid-level descriptor of an arbitrary block exactly with just two sums and two subtractions of low-dimensional vectors plus a final ℓ_2 normalization. A more detailed description of the efficient aggregation approach is provided in the appendix.

Building word representations. The mid-level features can then be encoded and aggregated into a global image representation using *e.g.* Fisher vectors. These global image representations can then be used by themselves, but can also be used as building blocks to more advanced methods that use global image signatures as input (*e.g.*, [29, 4]). We focus on the recent *word attributes* work of Almazán *et al.* [4], with available source code and state-of-the-art results in word image matching. This work uses Fisher vectors on SIFT descriptors as a building block to predict character attributes. The character attributes represent the presence or absence of a given character at a given relative position of the word (*e.g.*, "word contains an *a* in the second half of the word" or "word contains a *d* in the first third of the word"). Word images are then described by the predicted attribute scores. This attribute representation is then projected into an embedded space correlated with embedded text strings using CCA, which improves its discriminative power while reducing its dimensionality. In our experiments, we will use this to produce global image signatures of only 96 dimensions.

A great advantage of this framework is that representations of images and strings can then be compared using a cosine similarity, providing a unified framework to perform query-by-example (QBE) matching (*i.e.*, retrieve images of a dataset given a query image), query-by-string (QBS) matching (*i.e.*, retrieve images of a dataset given a query text string), and recognition (*i.e.*, retrieve text strings given a query image). Since the approach is already based on Fisher vectors, it is easy to replace the SIFT descriptors in the pipeline of [4] with our mid-level features and measure exactly their contribution.

4. Experiments

We start by describing the data used for learning the mid-level features and for evaluation purposes. We then describe the evaluation protocols and report the experimental results.

4.1. Datasets

Evaluation datasets. We evaluate our approach on two public benchmarks: IIIT5K [20] and Street-View Text (SVT) [36]. IIIT5K is the largest public annotated scene-text dataset to date, with $5,000$ cropped word images: $2,000$ training words and $3,000$ testing words. Each test-

Algorithm 2 Extract mid-level features

```
Input: Input image $\mathcal{I}$. Embedding matrix $\mathbf{U}$. Block sizes (e.g. $16 \times 16$, $32 \times 32$, etc). Block step size (e.g. 4 pixels).
Output: Mid-level features of image $\mathcal{I}$
  for each block size do
    for each possible block position inside the image (according to the block step size) do
      i) Densely extract SIFT descriptors inside the block.
      ii) Encode and aggregate the SIFT descriptors into a FV with a $2 \times 2$ spatial pyramid and $\ell_2$ normalize.
      iii) Project the FV with $\mathbf{U}$ and $\ell_2$ normalize again to construct the mid-level feature of the block.
    end for
  end for
  Return all mid-level features along with the position and scale of the blocks.
End
```

ing word is associated with a small text lexicon (SL) of 50 words and a medium text lexicon (ML) of $1,000$ words used for recognition tasks. Note that each word has a different lexicon associated with it. SVT is another popular dataset with about 350 images harvested from Google Street View. These images contain annotations of 904 cropped words: 257 for training purposes and 647 for testing purposes. Each testing image has an associated lexicon of 50 words. We also construct a *combined* lexicon (CL), that contains every possible word that appears in the lexicons of each dataset ($1,787$ unique words on IIIT5K and $4,282$ on SVT).

Learning dataset. Learning the proposed transformations requires gathering training words annotated at the character level. Fortunately, pixel level annotations and character bounding boxes exist for several standard datasets [16]. We gathered annotations for ICDAR 2003 [19], Sign Recognition 2009 [38], ICDAR 2011 [32], and IIIT5K [20] – for IIIT5K, only the training set annotations were used. In total, we gathered $3,829$ words annotated with approximately $22,500$ character bounding boxes that are used to learn the mid-level features transformation.

4.2. Implementation details

To construct our block FVs, we extract SIFTs [18] at 6 different scales, project them with PCA down to 64 dimensions, and aggregate them using FVs (gradients *w.r.t.* means and variances) with 8 Gaussians and a spatial pyramid of 2×2, leading to a dimensionality D_v of $2 \times 64 \times 8 \times 4 = 4,096$. During training, we sample 150 blocks per training image. In total, we sampled approximately $600,000$ blocks. To learn the projection matrix U with CCA, we use a regularization of $\eta = 1e^{-4}$ in all our experiments.

To construct a global representation, we first extract all mid-level blocks at 5 block sizes (16×16, 24×24, $\ldots$, 48×48) with a step size p of 4 pixels and $K = 62$ CCA dimensions using the efficient approach described in Section 3.2. Given images of 120 pixels in height, on average, we can extract and describe all blocks in less than a second using a MATLAB implementation and a single core. Then we append the normalized x and y coordinates of the center of the block as suggested by Sánchez *et al.* [31], and aggregate using a global FV with a 2×6 spatial pyramid with 16 Gaussians, leading to $24,576$ dimensions. These global Fisher vectors are then power- and ℓ_2- normalized [28], and can be compared using the dot-product as a similarity measure.

To construct the baseline global representation based only on SIFT (with no mid-level features) we follow a very similar approach, but instead of computing the mid-level blocks and reducing their dimensionality down to 62 dimensions with CCA, we use the SIFT features directly, reducing their dimensionality down to 62 dimensions with PCA and appending the normalized x and y coordinates of the center of the patch. This mimics the setup of the word attributes framework of [4]. We also experiment with reducing the dimensionality of our mid-level features in an unsupervised manner with PCA instead of CCA, to separate the influence of the extra layer of the architecture from the supervised learning.

When using word attributes, one has control of the final dimensionality of the representations. We set this output dimensionality to 96 dimensions. We observed that, in general, accuracy reached a plateau around that point: after that, increasing the number of dimensions does not significantly affect the performance. We found only one exception, where increasing the number of output dimensions beyond 96 in one of the SVT experiments significantly improved the results.

4.3. Evaluation

We evaluate our approach with two different setups. In the first one, we are interested in observing the effect of using supervised and unsupervised mid-level features instead of SIFT descriptors directly when computing global word image representations, *without applying any further supervised learning*. We compute Fisher vectors using a) SIFT features, b) unsupervised mid-level features (*i.e.*, dimensionality reduction of the block FVs with PCA), and c) supervised mid-level features (*i.e.*, dimensionality reduction with CCA). In this last case, we also explore the effect of the number of character regions (CR) used to compute the labels during training (*cf*. Equation (2)), from a 1×1 to a 4×4 region split. These FVs can be compared using the dot-product as a similarity measure.

We measure the accuracy in a query-by-example retrieval framework, where one uses a word image as a query, and the goal is to retrieve all the images of the same word in the dataset. We use each image of the test set in a leave-

Table 1. Baseline results on query-by-example using Fisher vectors (FV). In the case of the supervised mid-level features, we evaluate the effect of the number of character regions (CR) used during the learning of the features. See text for more details.

	IIIT5K		SVT	
	mAP	P@1	mAP	P@1
SIFT + FV	25.52	46.34	23.20	30.82
Unsup. mid-level + FV	20.32	38.10	18.38	25.68
Sup. mid-level (CR $= 1 \times 1$) + FV	33.96	52.20	29.75	37.46
Sup. mid-level (CR $= 2 \times 2$) + FV	41.35	60.20	34.29	42.60
Sup. mid-level (CR $= 3 \times 3$) + FV	42.73	61.48	37.38	45.92
Sup. mid-level (CR $= 4 \times 4$) + FV	**43.34**	**61.72**	**37.98**	**46.83**

one-out fashion to retrieve all the other images in the test set. Images that do not have any relevant item in the dataset are not considered as queries.

We report results in Table 1 using both mean average precision and precision at one as metrics. We highlight three aspects of the results. First, *using supervised mid-level features significantly improves over using SIFT features directly*, showing that the representation encodes more semantic information. Second, *the improvement in the mid-level features comes from the supervised information and not from the extra layer of the architecture*: the unsupervised mid-level features perform worse than the SIFT baseline. And, third, *encoding* which part *of the characters the blocks overlap with is more informative than only encoding which characters they overlap with*. We will use the 4×4 split for the rest of our experiments.

In the second set of experiments, we are interested in measuring how state-of-the-art global word image representations can benefit from these supervised features. In particular, we focus on the recent word attributes work of Almazán *et al.* [4], described in Section 3.2. We also explore the option of combining both FV representations (one based on SIFT and the other based on mid-level features), since their information may be complementary. To do so, we concatenate the global representations based on SIFT and mid-level features before learning the word attributes. To ensure the fairness of the comparisons, we learned the word attributes of [4] using a dataset comprised of the learning dataset described in Section 4.1 plus IIIT5K and SVT (excluding the test set of the target dataset). In this manner, word attributes based on SIFT and word attributes based on mid-level features have been trained using exactly the same images, and only the type of annotations (text transcriptions only vs text transcriptions and character bounding boxes) differs.

As in [4], we evaluate on three tasks: query-by-example (QBE) – *i.e.*, image-to-image retrieval –, query-by-string (QBS) – *i.e.*, text-to-image –, and recognition –*i.e.*, image-to-text. The accuracy of the QBE and QBS tasks is measured in terms of mean average precision. As is standard practice, we evaluate recognition using the small lexicon

Table 2. Retrieval results on IIIT5K and SVT datasets on the query-by-example (QBE) and query-by-string (QBS) tasks.

Dataset	Method	QBE	QBS
IIIT5K	Label Embedding [29]	43.70	-
	[SIFT] + FV + Atts [4]	68.37	72.62
	[**Prop. Mid-features**] + FV + Atts	75.77	78.61
	[**Prop. Mid-features** + SIFT] + FV + Atts	**75.91**	**78.62**
SVT	[SIFT] + FV + Atts [4]	60.20	81.96
	[**Prop. Mid-features**] + FV + Atts	62.68	83.83
	[**Prop. Mid-features** + SIFT] + FV + Atts	**65.94**	**85.36**

(SL) in IIIT5K and SVT, and the medium lexicon (ML) in IIIT5K. We also evaluate on our more challenging combined lexicon (CL). The recognition task is measured in terms of precision at 1.

Retrieval results. In Table 2 we report the retrieval results on IIIT5K and SVT. On both datasets using supervised mid-level features significantly improves over using SIFT features directly both on the query-by-example and query-by-string tasks. Combining the mid-level features and SIFT yields even further improvements on SVT. To the best of our knowledge, the best reported results on query-by-example and query-by-string on both datasets were those of [4], and using mid-level features significantly improves those results.

Recognition results. Table 3 shows the recognition results of our approach compared to the state-of-the-art. On IIIT5K, [4] held the best results, and these results are improved thanks to the mid-level features. In the more difficult medium and combined lexicons, these improvements are very noticeable: from 82.07 to 85.93 and from 77.77 to 83.03. A similar trend can be observed on SVT, where the improvement on the combined lexicon is very significant. Increasing the output dimensionality from 96 to 192 dimensions also significantly improves the accuracy on the SVT small lexicon recognition task up to an 91.81%. We only observed this behavior in this particular case; other datasets and tasks do not require larger representations.

The best reported results on SVT are those of [5] and [12], both of which use millions of training samples. Using a fraction of the training data we obtain results better than PhotoOCR [5], but are outperformed by Jaderberg *et al.* [12]. However, as seen on Figure 4, the performance of our method has not saturated: using more data as [12] does will likely increase the accuracy on both datasets. Our method also has other advantages such as leading to tiny (96-192 dimensions) signatures that can be used for image-to-image and text-to-image matching. Compared to methods that do not use millions of training samples [21, 9, 39, 13] , our results are significantly better.

Qualitative results. Figure 3 shows qualitative results of the query-by-example task for some IIIT5K queries. In the more difficult queries (*e.g.* “Before” or “HOUSE”) the mid-level features are clearly superior. In general, even when the

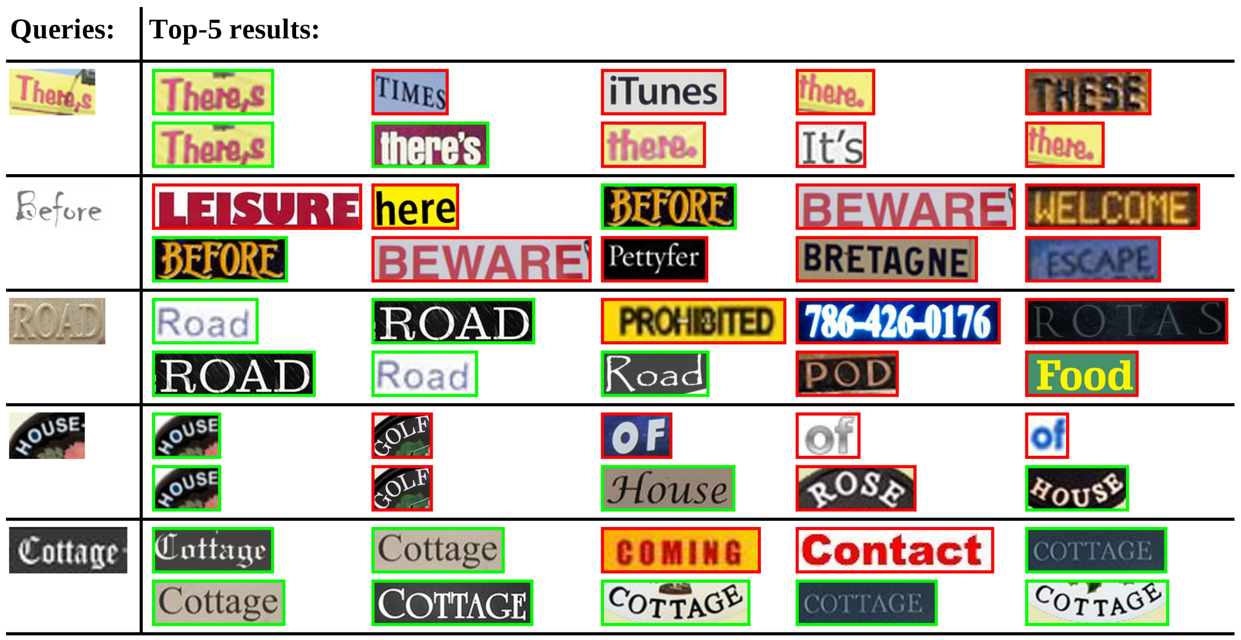

Figure 3. Qualitative query-by-example results for some IIIT5K queries. For each query, the first row displays the top retrieved images using SIFT word attributes, and the second row displays the top retrieved images using the proposed mid-level features with word attributes. Correct results are outlined in green.

Table 3. Comparison with the state-of-the-art on recognition accuracy on the IIIT5K and SVT datasets with small (SL), medium (ML), and combined (CL) lexicons. Methods marked with an $*$ use several millions of training samples.

Dataset	Method	SL	ML	CL
IIIT5K	High Order Language Priors [20]	64.10	57.50	-
	Label Embedding [29]	76.10	57.40	-
	Strokelets [39]	80.20	69.3	-
	[SIFT] + FV + Atts [4]	91.20	82.07	77.77
	[**Prop. Mid-features**] + FV + Atts	92.67	85.93	83.03
	[**Prop. Mid-features** + SIFT] + FV + Atts	**93.27**	**86.57**	**83.07**
SVT	ABBY [9]	35.00	-	-
	Mishra *et al.* [21]	73.26	-	-
	Synthesized Queries [9]	77.28	-	-
	Strokelets [39]	75.89	-	-
	*PhotoOCR [5] *(in house training data)*	90.39	-	-
	Deep CNN [13]	86.1	-	-
	*Deep CNN [12] *(synthetic training data)*	**95.4**	-	-
	[SIFT] + FV + Atts [4]	89.18	-	72.49
	[**Prop. Mid-features**] + FV + Atts	89.49	-	73.42
	[**Prop. Mid-features** + SIFT] + FV + Atts	90.73	-	**76.51**
	[**Prop. Mid-features** + SIFT] + FV + Atts (192d)	91.81	-	**76.51**

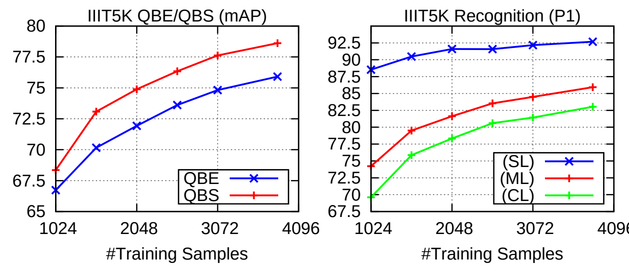

Figure 4. Accuracy on the IIIT5K dataset as a function of the number of annotated training words used to learn the mid-level space and the word attributes. Left: retrieval accuracy for the query-by-example (QBE) and query-by-string (QBS) tasks. Right: recognition accuracy using a small-sized lexicon (SL), a medium-sized lexicon (ML), and a combined lexicon (CL). In all cases the accuracy has not yet saturated, and more training data will likely improve the results.

retrieved results are not correct, they are closer to the query than when using SIFT features directly.

5. Conclusions

In this paper we have introduced supervised mid-level features for the task of word image representation. These features are learned by leveraging character bounding box annotations at training time, and correlate visual blocks with the characters (and, most importantly, the character regions) in which such blocks tend to appear. Despite using character information at training time, one key advantage of our approach is that it does not require localizing characters explicitly at testing time. Instead, our mid-level features can be densely extracted in an efficient manner. We used these mid-level features as a building block of the word attributes framework of [4]. The proposed mid-level features outperform equivalent representations based on SIFT on two standard benchmarks using tiny signatures of only 96 dimensions, and obtain state-of-the-art results on retrieval and recognition tasks. We finally note that the proposed approach can be seen as a way to learn mid-level features from annotated parts at training time (characters and character regions in our case) without explicitly localizing them at testing time. We believe that the key ideas behind these mid-level features are not limited to text, and could be exploited well beyond the scene-text domain, *e.g.*, for generic object categorization.

A. Efficient Word Representation

This appendix describes an approach to compute all mid-level features of a new image exactly in an efficient manner.

Once the matrix $\mathbf{U}$ has been learned as described in Section 3.1, one can use it to compute the mid-level features of a new image. A naive process would be as follows: i) Using a given stepsize (*e.g.* 4 pixels), extract all possible blocks at all sizes (see parameters used during the offline learning step). This leads to up to tens of thousands of overlapping blocks per image. ii) For each block, extract SIFT descriptors, compute a FV with a 2×2 spatial pyramid, and ℓ_2 normalize. iii) Project the FVs with $\mathbf{U}$, and ℓ_2 normalize again – ℓ_2 normalization after PCA/CCA usually improves the results since it accounts for the missing energy in the dimensionality reduction step [14]. This was shown in Algorithm 2.

Unfortunately, this process is not feasible in practice, as it would take a long time to compute and project all the FVs for all the blocks independently in a naive way. Instead, we propose an approach to compute these descriptors exactly in an efficient manner. The key idea is to isolate the contribution of each individual SIFT feature towards the mid-level feature of a block that included such a SIFT feature. If that contribution is linear, one can compute the individual contribution of each SIFT feature in the image and accumulate them in an integral representation. Then, the mid-level representation of an arbitrary block could be computed with just 2 additions and 2 subtractions over vectors, and computing all possible descriptors of all possible blocks becomes feasible and efficient.

At first sight the contribution of the features is not linear due to two reasons: the spatial pyramid, and the ℓ_2 normalization of the FVs *before* the projection with $\mathbf{U}$. Fortunately, both problems can be overcome. Our approach works thanks to three key properties:

1) The FV is additive when not performing any normalization. Given a set of descriptors $\mathcal{S}$, $fv(\mathcal{S}) = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} fv(s)$, since the FV aggregates the encoded descriptors using average pooling. This is key since it implies that we can compute FVs independently and then aggregate them. This is also true if the descriptors are projected linearly, *i.e.* $\mathbf{U}^T fv(\mathcal{S}) = \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} \mathbf{U}^T fv(s)$. However, it is *not* true if we ℓ_2 normalize the FVs, *i.e.*, $fv(\mathcal{S})/||fv(\mathcal{S})||_2 \neq \frac{1}{|\mathcal{S}|}\sum_{s\in\mathcal{S}} fv(s)/||fv(s)||_2$. Fortunately, ℓ_2 normalizing the FVs will not be necessary, *cf.* next property.

2) The normalization of the FV is absorbed by the normalization after the projection with $\mathbf{U}$. If we let f be an unnormalized FV and let f_2 be the ℓ_2 normalized version, then $\mathbf{U}^T f_2/||\mathbf{U}^T f_2||_2 = \mathbf{U}^T f/||\mathbf{U}^T f||_2$. Even if we ℓ_2 normalize the FVs when learning the projection with CCA (since ℓ_2 normalization greatly increases its discriminative power), it is not necessary to ℓ_2 normalize them when representing the words if they are also going to be normalized after projection, which makes the aggregation of FVs seen in property 1 possible. Note how the $\frac{1}{|\mathcal{S}|}$ factor is also absorved by the ℓ_2 normalization.

3) The projection of a FV with spatial pyramid is also additive. Projecting a FV f with spatial pyramid with $\mathbf{U}$ is equivalent to projecting each spatial region independently with the corresponding rows of $\mathbf{U}$ and then aggregating the results. Assuming spatial pyramids of 2×2, it is also possible to rearrange $\mathbf{U}$ into $\hat{\mathbf{U}} \in \mathbb{R}^{\frac{D_v}{4} \times 4K}$. In this case, projecting a FV *without spatial pyramid* with $\hat{\mathbf{U}}$ leads to a descriptor of $4K$ dimensions. Each group of K dimensions represents the results of projecting f assuming that it was representing one of the 4 spatial quadrants of a larger block.

By exploiting these properties, we can compute the mid-level descriptors of all possible blocks in an image in an efficient manner. In our approach, we begin by dividing the target image in contiguous, non-overlapping cells of $p \times p$ pixels, where p controls the step size between two consecutive blocks. In our experiments we set $p = 4$. We compute a FV in each of these regions *with neither spatial pyramids nor ℓ_2 normalization* and project them with $\hat{\mathbf{U}}$ into a space of $4K$ dimensions. This leads to a grid representation of the image $\mathcal{G} \in \mathbb{R}^{\frac{H}{p} \times \frac{W}{p} \times 4K}$, where H and W are the height and the width of the image. The "depth" of the representation can be separated into 4 groups that represent the projection of that particular grid cell depending on its position on the pyramid. Figure 5 illustrates this. Note that, since the cells do not overlap, computing the FVs of all those cells has approximately the same cost as computing one single FV using the descriptors of all the image, and does not involve any particular extra cost. Keeping in memory $\frac{H}{p} \times \frac{W}{p} \times 4K$ elements at the same time is also not an issue. Finally, we compute an integral representation over the height and the width, *i.e.*, $\hat{\mathcal{G}}_{i,j,k} = \sum_{1\leq a\leq i, 1\leq b\leq j} \mathcal{G}_{a,b,k}$.

At this point, one can easily compute the descriptor of any given block by i) separating the block in 4 different spatial regions. ii) Computing the descriptor of each region independently with two sums and two subtractions on vectors of dimension K by accessing the corresponding rows and columns of the integral representation $\hat{\mathcal{G}}$, and by keeping only the group of K dimensions corresponding to the particular spatial regions. iii) Aggregating the descriptors of each spatial region, and iv) ℓ_2 normalizing the final result.

In our experiments we use images of 120 pixels in height,

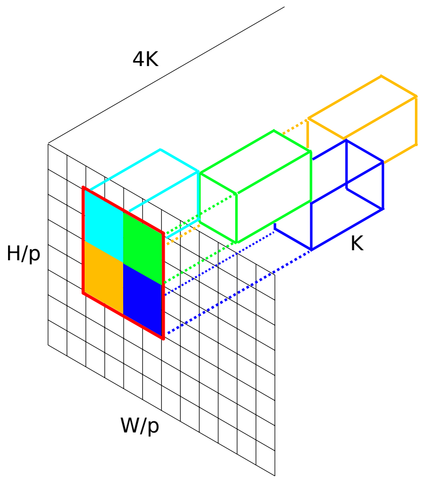

Figure 5. Grid $\mathcal{G}$ representing the influence of each descriptor in an image towards the mid-level features. To compute the mid-level descriptor of a block (*e.g.* the red block), one has to sum the K-dimensional descriptors inside the colored cuboids –done efficiently using an integral representation–, and ℓ_2 normalize the final result.

a step size of $p = 4$, a block spatial pyramid of 2×2, and $K = 62$ projections. We also extract blocks at 5 different block sizes: 16×16, 24×24, $\ldots$, 48×48. With this setup, we can extract and describe all blocks in an image in less than a second using a MATLAB implementation and a single core.

References

[1] D. Aldavert, M. Rusiñol, R. Toledo, and J. Lladós. Integrating Visual and Textual Cues for Query-by-String Word Spotting. In *ICDAR*, 2013. 1, 2

[2] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Efficient exemplar word spotting. In *BMVC*, 2012. 1, 2

[3] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Handwritten word spotting with corrected attributes. In *ICCV*, 2013. 1, 2, 3

[4] J. Almazán, A. Gordo, A. Fornés, and E. Valveny. Word spotting and recognition with embedded attributes. *TPAMI*, 2014. http://www.cvc.uab.es/~almazan/index.php/projects/words-att/. 1, 2, 3, 5, 6, 7, 8

[5] A. Bissacco, M. Cummins, Y. Netzer, and H. Neven. PhotoOCR: Reading Text in Uncontrolled Conditions. In *ICCV*, 2013. 1, 2, 7, 8

[6] Y.-L. Boureau, F. Bach, Y. LeCun, and J. Ponce. Learning mid-level features for recognition. In *CVPR*, 2010. 2

[7] G. Csurka, C. R. Dance, L. Fan, J. Willamowski, and C. Bray. Visual Categorization with Bags of Keypoints. In *ECCV SLCV Workshop*, 2004. 1

[8] N. Dalal and B. Triggs. Histograms of Oriented Gradients for Human Detection. In *CVPR*, 2005. 1

[9] V. Goel, A. Mishra, K. Alahari, and C. V. Jawahar. Whole is greater than sum of parts: Recognizing scene text words. In *ICDAR*, 2013. 2, 7, 8

[10] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik. A multi-view embedding space for modeling internet images, tags, and their semantics. *IJCV*, 2013. 3

[11] H. Hotelling. Relations between two sets of variables. *Biometrika*, 1936. 4

[12] M. Jaderberg, K. Simonyan, A. Vedaldi, and A. Zisserman. Synthetic Data and Artificial Neural Networks for Natural Scene Text Recognition . *CoRR*, abs/1406.2227, 2014. 2, 7, 8

[13] M. Jaderberg, A. Vedaldi, and A. Zisserman. Deep Features for Text Spotting. In *ECCV*, 2014. 7, 8

[14] H. Jégou and O. Chum. Negative evidences and co-occurrences in image retrieval: the benefit of PCA and whitening. In *ECCV*, 2012. 9

[15] M. Juneja, A. Vedaldi, C. Jawahar, and A. Zisserman. Blocks that shout: Distinctive parts for scene classification. In *CVPR*, 2013. 2

[16] D. Kumar, M. A. Prasad, and A. Ramakrishnan. Benchmarking recognition results on camera captured word image datasets. In *Document Analysis and Recognition*, 2012. http://mile.ee.iisc.ernet.in/bench/. 6

[17] S. Lazebnik, C. Schmid, and J. Ponce. Beyond bags of features: Spatial Pyramid matching for recognizing natural scene categories. In *CVPR*, 2006. 3, 4

[18] D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 2004. 1, 4, 6

[19] S. Lucas, A. Panaretos, L. Sosa, A. Tang, S. Wong, and R. Young. ICDAR 2003 Robust Reading Competition. In *ICDAR*, 2003. 6

[20] A. Mishra, K. Alahari, and C. V. Jawahar. Scene text recognition using higher order language priors. In *BMVC*, 2012. 1, 2, 5, 6, 8

[21] A. Mishra, K. Alahari, and C. V. Jawahar. Top-down and bottom-up cues for scene text recognition. In *CVPR*, 2012. 1, 2, 7, 8

[22] L. Neumann and J. Matas. Real-Time Scene Text Localization and Recognition. In *CVPR*, 2012. 2

[23] L. Neumann and J. Matas. Scene Text Localization and Recognition with Oriented Stroke Detection. In *ICCV*, 2013. 1, 2

[24] X. Peng, C. Zou, Y. Qiao, and Q. Peng. Action Recognition with Stacked Fisher Vectors. In *ECCV*, 2014. 3

[25] F. Perronnin and C. R. Dance. Fisher Kernels on Visual Vocabularies for Image Categorization. In *CVPR*, 2007. 1, 4

[26] F. Perronnin, Y. Liu, J. Sánchez, and H. Poirier. Large-scale image retrieval with compressed fisher vectors. In *CVPR*, 2010. 4

[27] F. Perronnin and J. A. Rodríguez-Serrano. Fisher kernels for handwritten word-spotting. In *ICDAR*, 2009. 1

[28] F. Perronnin, J. Sánchez, and T. Mensink. Improving the Fisher kernel for large-scale image classification. In *ECCV*, 2010. 4, 6

[29] J. A. Rodríguez-Serrano and F. Perronnin. Label embedding for text recognition. In *BMVC*, 2013. 1, 2, 3, 5, 7, 8

[30] M. Rusiñol, D. Aldavert, R. Toledo, and J. Lladós. Browsing heterogeneous document collections by a segmentation-free word spotting method. In *ICDAR*, 2011. 1, 2, 3

[31] J. Sánchez, F. Perronnin, and T. de Campos. Modeling the spatial layout of images beyond spatial pyramids. *PRL*, 2012. 6

[32] A. Shahab, F. Shafait, and A. Dengel. ICDAR 2011 Robust Reading Competition - Challenge 2: Reading Text in Scene Images. In *ICDAR*, 2011. 6

[33] V. Sharmanska, N. Quadrianto, and C. Lampert. Learning to Rank Using Privileged Information. In *ICCV*, 2013. 3

[34] K. Simonyan, A. Vedaldi, and A. Zisserman. Deep fisher networks for large-scale image classification. In *NIPS*, 2013. 3

[35] V. Vapnik and A. Vashist. A New Learning Paradigm: Learning Using Privileged Information. *Neural Networks*, 2009. 3

[36] K. Wang, B. Babenko, and S. Belongie. End-to-end Scene Text Recognition. In *ICCV*, 2011. 5

[37] K. Wang and S. Belongie. Word Spotting in the Wild. In *ECCV*, 2010. 1

[38] J. Weinman and A. H. E. Learned-Milles. Scene Text Recognition using Similarity and a Lexicon with Sparse Belief Propagation. *TPAMI*, 2009. 6

[39] C. Yao, X. Bai, B. Shi, and W. Liu. Strokelets: A learned multi-scale representation for scene-text recognition. In *CVPR*, 2014. 2, 3, 7, 8