Learning and calibrating per-location classifiers for visual place recognition

Petr Gronát[1,3] · **Guillaume Obozinski**[2] · **Josef Sivic**[1] · **Tomáš Pajdla**[3]

July, 2016

Abstract The aim of this work is to localize a query photograph by finding other images depicting the same place in a large geotagged image database. This is a challenging task due to changes in viewpoint, imaging conditions and the large size of the image database. The contribution of this work is two-fold. First, we cast the place recognition problem as a classification task and use the available geotags to train a classifier for each location in the database in a similar manner to per-exemplar SVMs in object recognition. Second, as only one or a few positive training examples are available for each location, we propose two methods to calibrate all the per-location SVM classifiers without the need for additional positive training data. The first method relies on p-values from statistical hypothesis testing and uses only the available negative training data. The second method performs an affine calibration by appropriately normalizing the learnt classifier hyperplane and does not need any additional labelled training data. We test the proposed place recognition method with the bag-of-visual-words and Fisher vector image representations suitable for large scale indexing. Experiments are performed on three datasets: 25,000 and 55,000 geotagged street view images of Pittsburgh, and the 24/7 Tokyo benchmark containing $76,000$ images with varying illumination conditions. The results show improved place recognition accuracy of the learnt image representation over direct matching of raw image descriptors.

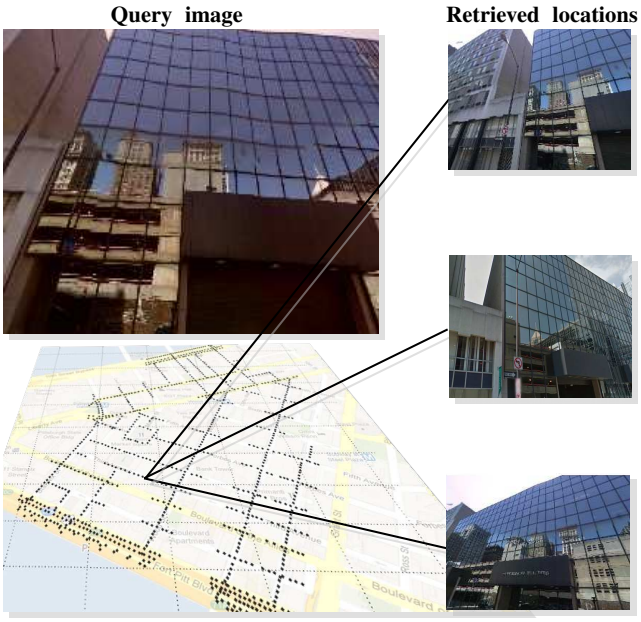


Fig. 1: The goal of this work is to localize a query photograph (left top) by finding other images of the same place in a large geotagged image database (right column). We cast the problem as a classification task and learn a classifier for each location in the database. We develop two procedures to calibrate the outputs of the large number of per-location classifiers without the need for additional labeled training data.

1 Inria, Willow project, Departement d'Informatique de l'Ecole Normale Superieure, ENS/INRIA/CNRS UMR 8548, 72012 Paris, France

2 Ecole des Ponts ParisTech, 6, av. Blaise Pascal, 77455 Marne-la-Valle, France

3 Czech Technical University in Prague, Faculty of Electrical Engineering, Department of Cybernetics, Czech Republic

1 Introduction

Visual place recognition ([11, 29, 43]) is a challenging task as the query and database images may depict the same 3D structure (e.g. a building) from a different camera viewpoint, under different illumination, or the building can be partially occluded. In addition, the geotagged database may be very large. For example, we estimate that Google Street View of France alone contains more than 60 million panoramic images. It is, however, an important problem as automatic, accurate and fast visual place recognition would have many practical applications in robotics, augmented reality or navigation.

Similar to other work in large scale place recognition ([11, 29, 43, 52]) and image retrieval ([36, 37, 47, 26]), we describe each image by a set of local invariant features ([5, 33]) that are encoded and aggregated into a fixed-length single vector descriptor for each image. In particular, in this work we consider the sparse tf-idf weighted bag-of-visual-words representation ([47, 37]) and the compact Fisher vector descriptors ([26]).

The resulting vectors are then normalized to have unit L_2 norm and the similarity between the query and a database vector is measured by their dot product. This representation has some desirable properties such as robustness to background clutter and partial occlusion. Efficient retrieval can then achieved using inverted file indexing ([24]).

While in image retrieval databases are typically unstructured collections of images, place recognition databases are usually structured: images have geotags, are localized on a map and depict a consistent 3D world. Knowing the structure of the database can lead to significant improvements in both speed and accuracy of place recognition. Examples include: (i) building an explicit 3D reconstruction of the scene ([23, 31, 32]); (ii) constructing an image graph ([6, 38, 53]), where images are nodes and edges connect close-by images on the map ([51]), or (iii) using the geotagged data as a form of supervision to select local features that characterize a certain location ([29, 43]) or re-rank retrieved images ([55]).

In this work, we also take advantage of geotags as an available form of supervision and investigate whether the place recognition problem can be cast as a classification task. Learning visual classifiers has been investigated for landmark recognition ([30]) where consumer photographs were clustered into landmark classes based on geo-tags. In this work we wish to recognize individual street locations rather than a small number of landmarks, and as a consequence have only a few (1-5) photographs capturing the same location. In particular, we train a classifier *for each location on the map* in a similar manner to per-exemplar classification in object recognition ([34]). (By location we mean a scene captured by a camera, hence using our terminology, a camera heading to the north captures different location than a camera heading to the south, even though these two cameras share the same GPS location. In the following text we use terms place and location interchangeably.)

This is beneficial as each classifier can learn which features are discriminative for a particular place. The classifiers are learnt offline. At query time, the query photograph is localized by transferring the GPS tag of the best scoring location classifier.

While learning classifiers for each place may be appealing, calibrating outputs of the individual classifiers is a critical issue. In object recognition ([34]), it is addressed in a separate calibration stage on a held-out set of training data. This is not possible in the place recognition set-up as only a small number, typically one to five, of positive training images are available for each location (e.g. street view images viewing the same building facade). To address this issue, we propose two calibration methods. The first method relies on p-values from statistical hypothesis testing and uses only the available negative training data. The second method performs a simple affine calibration by appropriately normalizing the learnt classifiers and does not need any additional labelled calibration examples.

2 Related work

The task of geo-localizing a given input query photograph has recently received considerable attention. The output can be a coarse geo-localization on the level of continents and cities ([14, 22, 27]) or a name of the depicted landmark ([30]). In this work we focus on visually recognizing the "same place" by finding an image in geo-tagged database that depicts the same building facade or street-corner as shown in the query ([8, 11, 29, 43, 52, 55]).

This *visual place recognition* problem is typically treated as large-scale instance-level retrieval ([11, 8, 29, 43, 52, 55]), where images are represented using local invariant features ([33]) encoded and aggregated into the bag-of-visual-words ([10, 47]) or Fisher vector ([26]) representations. The image database can be further augmented by 3D point clouds ([28]), automatically reconstructed by large-scale structure from motion (SfM) ([1, 28]), which enables accurate prediction of query image camera position ([32, 40]). In contrast, in this work we investigate learning a discriminative place-specific image representation. A similar idea has been recently explored in ([6]) who learn a graph-based discriminative representation for landmark image collections where typically many images are available for each landmark. In this work, we focus on street-level images such as Google Street View, which have greater coverage, but typically only one or a small number of images are available for each location. To address this issue we learn a discriminative reweighting of the descriptor specific to each image in the database using per-exemplar support vector machine ([34]).

2.1 Per-exemplar support vector machines

The exemplar support vector machines (e-SVM) has been used in a number of visual recognition tasks including category-level recognition ([34]), cross-domain retrieval ([45]), scene parsing ([48]) or as an initialization for more complex discriminative clustering models ([14, 46]). The main idea is to train a linear support vector machine (SVM) classifier from a single positive example and a large number of negatives. The intuition is that the resulting weight vector will give a higher weight to the discriminative dimensions of the positive training data point and will down weight dimensions that are non-discriminative with respect to the negative training data.

The exemplar support vector machine can be learnt at query time where the weight vector is used as a new query image representation ([45]). However, this requires training a new classifier afresh for each query that is computationally very demanding. In this work, similar to ([34]) who learn per-exemplar object category representation, we learn per-exemplar classifiers for each place in the database off-line. A key advantage is that each per-exemplar classifier is trained independently and hence the learning can be heavily parallelized. The per-exemplar training brings, however, also an important drawback. As each classifier is trained independently a careful calibration of the resulting classifier scores is required ([34]).

2.2 Calibrating classifier scores

Several calibration approaches have been proposed in the literature (see ([16]) and references therein for a review). The most known consists of fitting a logistic regression to the output of the SVM ([39]). This approach, however, has a major drawback as it imposes a parametric form (the logistic a.k.a. sigmoid function) of the likelihood ratio of the two classes, which typically leads to biased estimates of the calibrated scores. Another important calibration method is the isotonic regression ([54]), which allows for a non-parametric estimate of the output probability. Unfortunately, the fact that we have only a single positive example (or only very few of them which are almost identical, and which are all used for training) essentially prevents us from using any of these methods. To address these issues, we develop two classifier calibration methods that do not need additional labelled positive examples. Related to ours is also the recent work of Scheirer et al. ([42]) who develop a classifier calibration method for face attribute similarity search. Their method (discussed in more detail in section 4) also does not require labelled positive examples but, in contrast to us, uses a parametric model (the Weibull distribution) for the scores of negative examples.

2.3 Linear discriminant analysis and whitening

Our work is also related to linear discriminative transformations of feature space that have shown good performance in object recognition ([17, 21]) and 2D-3D alignment ([4, 3]). While conceptually the idea of finding a discriminative projection of the original feature space is similar to our work, the main difference is in the used loss function. While we use hinge loss ([44]) to train the new discriminative representation of each place, ([4, 17, 21]) use the Euclidean loss. The advantage of using the Euclidean loss is that the discriminative projection can be computed in closed form. The resulting projection is tightly related to Linear Discriminant Analysis and whitening the feature space ([4, 17, 21]). Such whitened representations have shown promise for image retrieval ([25]) or matching HOG ([12]) descriptors ([13]), however, we have found they do not perform well for place recognition.

2.4 Contributions

This paper has two main contributions. First, we cast the place recognition problem as a classification task where we use the available geo-tags as a weak form of supervision to train a classifier for each location in the database (section 3). These classifiers are subsequently used for ranking the database images at query time.

Second, as only a few positive training examples are available for each location, we propose two methods to calibrate all the per-location SVM classifiers without the need for additional positive training data. The first method (section 4) relies on p-values from statistical hypothesis testing. The second method (section 5) performs an affine calibration by appropriately normalizing the learnt decision hyperplane. We also describe a memory efficient classifier representation for the sparse bag-of-visual-word vectors (section 6) and experimentally demonstrate benefits of the proposed approach (sections 7 and 8).

3 Per-location classifiers for place recognition

We are given an image descriptor $\mathbf{x}_j$, one for each database image j. This representation can be a sparse tf-idf weighted bag-of-visual-words vector ([47]) or a dense compact descriptor such as the Fisher vector (FV) ([26]). The goal is to learn a score function f_j for each database image j, so that, at test time, given the descriptor $\mathbf{q}$ of the query image, we can either retrieve the correct target image as the image j^* with the highest score

$$j^* = \arg\max_j f_j(\mathbf{q}) \tag{1}$$

or use these scores to rank candidate images and use geometric verification to identify the correct location in an n-best list. Instead of approaching the problem directly as a large multiclass classification problem, we tackle the problem by learning a per-exemplar linear SVM classifier ([34]) for each database image j. Similar to ([29]), we use the available geotags to construct the negative set $\mathcal{N}_j$ for each image j. The negative set is constructed so as to concentrate difficult negative examples, i.e. from images that are far away from the location of image j and similar to the target image as measured by the dot product between their feature vectors. The details of the construction procedure will be given in section 7. The positive set $\mathcal{P}_j$ is represented by a single positive example, which is $\mathbf{x}_j$ itself. For some locations,the positive set can be expanded, details of this procedure will be given in 7.3. Each SVM classifier produces a classification score s_j which is a priori not comparable with the score of the other classifiers. A calibration of these classification scores will therefore be key to convert them to comparable scores f_j. This calibration problem is more difficult than usual given that we only have a single positive example and will be addressed in section 4.

3.1 Learning per-location SVM classifiers

Each linear SVM classifier generates a classification score s_j of the form

$$s_j(\mathbf{q}) = \mathbf{q}^T \mathbf{w}_j + b_j \tag{2}$$

where $\mathbf{w}_j$ is a weight vector re-weighting contributions of individual visual words and b_j is the bias specific for image j. Given the training sets $\mathcal{P}_j$ and $\mathcal{N}_j$, the aim is to find a vector $\mathbf{w}_j$ and bias b_j such that the score difference between $\mathbf{x_j}$ and the closest neighbor from its negative set $\mathcal{N}_j$ is *maximized*. Learning the weight vector $\mathbf{w}_j$ and bias b_j is formulated as a minimization of the convex objective

$$\begin{aligned}\Omega(\mathbf{w}_j, b_j) = ||\mathbf{w}_j||^2 + C_1 \sum_{\mathbf{x}\in\mathcal{P}_j} h(\mathbf{w}_j^T\mathbf{x} + b_j) \\ + C_2 \sum_{\mathbf{x}\in\mathcal{N}_j} h(-\mathbf{w}_j^T\mathbf{x} - b_j),\end{aligned} \tag{3}$$

where the first term is the regularizer, the second term is the loss on the positive training data weighted by scalar parameter C_1, and the third term is the loss on the negative training data weighted by scalar parameter C_2. This is a standard SVM formulation (3), also used in exemplar-SVM ([34]). In our case h is the squared hinge loss, which we found to work better in our setting than the standard hinge-loss. Parameters $\mathbf{w}_j$ and b_j are learned separately for each database image j in turn.

3.2 The need for calibrating classifier scores

Since the classification scores s_j are learned independently for each location j, they cannot be directly used for place recognition as in eq. (1). As illustrated in figure 2, for a given query $\mathbf{q}$, a classifier from an incorrect location (b) can have a higher score (eq. (2)) than the classifier from the target location (a). Indeed, the SVM score is a signed distance from the discriminating hyperplane and is a priori not comparable between different classifiers. This issue is addressed by calibrating scores of the learnt classifiers. The goal of the calibration is to convert the output of each classifier into a probability (or in general a "universal" score), which can be meaningfully compared across classifiers. In the following two sections we develop two classifier calibration methods that do not need additional labelled positive examples.

4 Non-parametric calibration of the SVM-scores from negative examples only

In this section we describe a classifier calibration method that exploits the availability of large amounts of negative data, i.e. images from other far away locations in the database. In particular, the method estimates the significance of the classification score of a test example compared to the typical classification score of the (plentifully available) negative examples. Intuitively, we will use a large dataset of negative examples to calibrate the individual classifiers so that they *reject the same number of negative examples* at each level of the calibrated score. We will expand this idea in detail using the concepts from hypothesis testing.

4.1 Calibration *via* significance levels

In the following, we view the problem of deciding whether a query image matches a given location based on the corresponding SVM classification score as a hypothesis testing problem. In particular, we appeal to ideas from the traditional frequentist hypothesis testing framework also known as Neyman-Pearson (NP) framework (see e.g. ([7]), chap. 8).

We define the null hypothesis as $H_0 = \{$the image is a random image$\}$ and the alternative as $H_1 = \{$the image matches the particular location$\}$. The NP framework focuses on the case where the distribution of the data under H_0 is well known, whereas the distribution under H_1 is not accessible or too complicated to model, which matches perfectly our setting.

In the NP framework, the *significance level* of a score is measured by the p-value or equivalently by the value of the cumulative density function (cdf) of the distribution of the negatives at a given score value. The cdf is the function F_0

defined by $F_0(s) = \mathbb{P}(S_0 \leq s)$, where S_0 is a random variable corresponding to the classification scores of negative data (see figure 3 for an illustration of the relation between the cdf and the density of the function). The cdf (or the corresponding p-value[1]) is naturally estimated by the empirical cumulative density function $\hat{F}_0$, which is computed as:

$$\hat{F}_0(s) = \frac{1}{N_c} \sum_{n=1}^{N_c} 1_{\{s_n \leq s\}}, \quad (4)$$

where $(s_n)_{1 \leq n \leq N_c}$ are the SVM classification scores associated with N_c negative examples used for calibration. Note that no positive examples are involved in the construction of the cumulative density function. $\hat{F}_0(s)$ is the fraction of the negative examples used for calibration (ideally held out negative examples) that have a score below a given value s. Computing $\hat{F}_0$ exactly would require to store all the SVM scores for all the calibration data for all classifiers, so in practice, we only keep a fraction of the larger scores. We also interpolate the empirical cdf between consecutive datapoints so that instead of being a staircase function it is a continuous piecewise linear function such as illustrated in figure 2. Given a query, we first compute its SVM classification score s_q and then compute the calibrated probability, the score function $f(q) = \hat{F}_0(s_q)$. We obtain a similar calibrated score function $f_j(q)$ for each of the SVMs associated with each of the target locations, which can now be ranked. Two other examples of score calibration functions are shown in figure 6 in section 8. Note that while figure 2 illustrates only few points on the cdf, the two plots in figure 6 show a complete cdf that contains on the order of $25k$ data points. Note also that the two cumulative density functions in figure 6 are similar but not identical.

4.2 Summary of the calibration procedure

For each trained place-specific classifier s_j we construct the empirical cumulative density function (4) of classification scores of the negative examples and keep only its top K values. This can be done offline and the procedure is summarized in Algorithm 1. At query time, given a query image descriptor $\mathbf{q}$, we compute the uncalibrated classification score $s_j(\mathbf{q})$ and then use the stored cdf values to compute the calibrated score $f_j(\mathbf{q})$. This procedure is performed for each database image j and is summarized in Algorithm 2. Finally, the best candidate database image is selected by

[1] The notion most commonly used in statistics is in fact the p-value. The p-value associated to a score is the quantity $\alpha(s)$ defined by $\alpha(s) = 1 - F_0(s)$; so the more significant the score is, the closer to 1 the cdf value is, and the closer to 0 the p-value is. To keep the presentation simple, we avoid the formulation in terms of p-values and we only talk of the probabilistic calibrated values obtained from the cdf F_0.

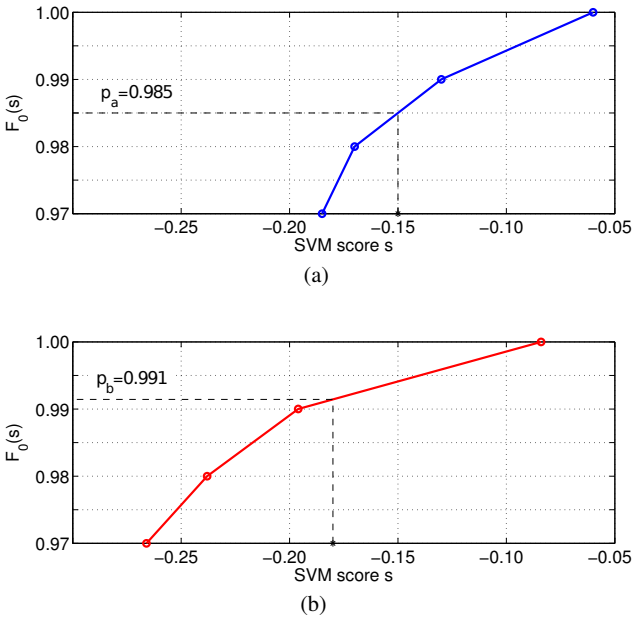


Fig. 2: **An illustration of the proposed normalization of SVM scores for database images.** In each plot, the x-axis shows the raw SVM score. The y-axis shows the calibrated output. For the given query, the raw SVM score of image (b) is lower than for image (a), but the calibrated score of image (b) is higher than for image (a).

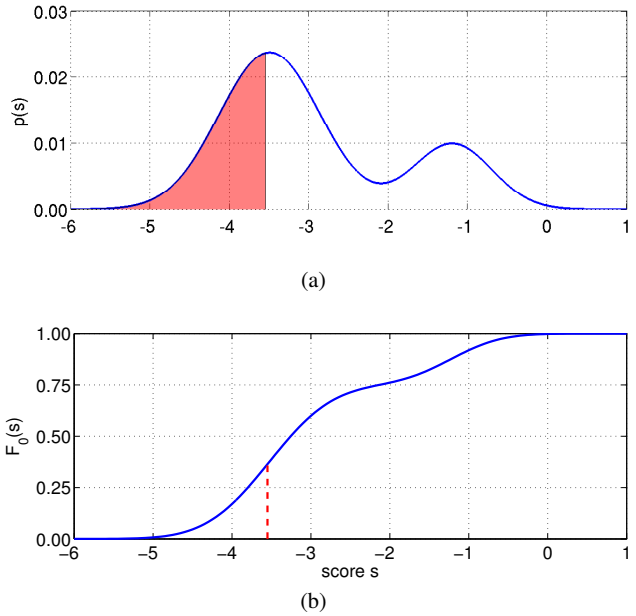


Fig. 3: **Cumulative density function.** Illustration of the relation between (a) the probability density of the random variable S_0 modeling the scores of the negative examples and (b) the corresponding cumulative density function $F_0(s) = \mathbb{P}(S_0 \leq s)$.

equation (1). Alternatively, candidate database images can be also ranked according to the calibrated score.

Algorithm 1 P-value calibration: offline stage

Input: $\mathcal{X}$ ... column wise matrix of image descriptors
$\mathbf{w}_j, b_j$ ... learnt SVM weights and biases
Output: $\hat{F}_{0j}$ ... calibration functions

procedure P-VALUE CALIBRATION
  $N \leftarrow$ database size
  $\mathcal{X} \leftarrow$ descriptor matrix of negative examples
  for $\forall j \in 1 \dots N$ **do**
    $N_c \leftarrow$ number of negative examples
    $\mathbf{w} \leftarrow$ learned SVM weight for image j
    $b \leftarrow$ learned SVM bias for image j
    $\boldsymbol{\sigma} \leftarrow \mathbf{w}^T\mathcal{X} + b$
    Compute the cdf:
    $\mathbf{s_j} \leftarrow$ sorted $\boldsymbol{\sigma}$ in descending order
    $\hat{F}_{0j} \leftarrow [N_c \dots 0]/N_c$

Algorithm 2 P-value calibration: online stage

Input: $\mathbf{q}$ ... query image descriptor
$\mathbf{w}_j, b_j$ ... learnt SVM weights and biases
$\hat{F}_{0j}$ ... learnt calibration function
Output: $f_j(\mathbf{q})$ ... calibrated score

procedure CALIBRATING SCORES
  $\mathbf{q} \leftarrow$ query image descriptor
  $N \leftarrow$ database size
  for $\forall j \in 1 \dots N$ **do** // for each database image
    $\mathbf{w} \leftarrow$ learned SVM weight for image j
    $b \leftarrow$ learned SVM bias for image j
    $\hat{F}_0 \leftarrow \hat{F}_{0j}$ // Empirical cdf
    $\mathbf{s} \leftarrow \mathbf{s_j}$ // Corresponding sorted scores
    $s_q \leftarrow \mathbf{q}^T\mathbf{w} + b$ // compute uncalibrated classifier score
    Find n such that $s_n \leq s_q < s_{n+1}$
    Compute the interpolated empirical cdf value:
    $\hat{F}_0(s_q) \approx \hat{F}_0(s_n) + \frac{s_q - s_n}{s_{n+1} - s_n}(\hat{F}_0(s_{n+1}) - \hat{F}_0(s_n))$.
    $f_j(\mathbf{q}) = \hat{F}_0(s_q)$ // output the calibrated score

4.3 Discussion

It should be noted that basing the calibration only on the negative data has the advantage that we privilege precision over recall, which is justified given the imbalance of the available training data (many more negatives than positives). Indeed, since we are learning with a single positive example (or a very few), intuitively, we cannot guarantee that the learned partition of the space will generalize well to other positives, whose scores in the test set can potentially drop significantly. By contrast, since we are learning from a comparatively large number of negative examples, we can trust the fact that new negative examples will stay in the half-space containing the negative training set, so that their scores are very unlikely to be large. Our method is therefore based on the fact that we can measure reliably how surprising a high classification score would be if it was the score of a negative example. This exactly means that we can control false positives (type I error) reasonably well but not false negatives (type II error or equivalently the power of our test/classifier), exactly as in the Neyman-Pearson framework.

An additional reason for not relying on positive examples for the calibration in our case is that (even if we had sufficiently many of them) the positive examples that we collect using location and geometric verification from the geotagged database typically have illumination conditions that are extremely similar to each other and not representative of the distribution of test positives which can have very different illuminations. This is because of the controlled nature of the capturing process of geotagged street-level imagery (e.g. Google Street View) used for experiments in this work. Close-by images are typically captured at a similar time (e.g. on the same day) and under similar imaging conditions.

Scheirer et al. ([42]) propose a method, which is related to ours, and calibrate SVM scores by computing the corresponding cdf value of a Weibull distribution fitted to the top negative scores. The main difficulty is that the Weibull model should be fitted only to the tail of the distribution of the negatives, which is in general difficult to identify. As a heuristic, Scheirer et al. propose to fit the Weibul model to false positives (i.e. the negative samples classified incorrectly as positives). But in our case, most of the exemplar SVMs that we are training have zero false positives in a held out set, which precludes the application of their method.

Finally, we should remark that we are not doing here calibration in the same sense of the word as the calibration based on logistic regression (or isotonic regression), since logistic regression estimates the probability of making a correct prediction by assigning a new data to class 1, while we are estimating how unlikely it would be for a negative example to have such a high score. The calibration with either methods yields "universal" scores in the sense that they are comparable from one SVM to another, but the calibrated values obtained from logistic regression are not comparable to the values obtained from our approach.

5 Affine calibration by normalizing the classification hyperplane

The non-parametric calibration method described in the previous section has two computational disadvantages, which make it hard to scale-up to very large datasets. First, the method requires storing the non-parametric model of the calibration function for each learned classifier. This has memory complexity of $O(NK)$, where N is the number of images (classifiers) in the database and K the number of stored

elements of the non-parametric model. For typical values of $K = 1000$ and $N = 1$M this would require additional 4GB of memory, comparable to the size of the inverted index itself. Second, computing the cumulative density function requires applying all N learnt classifiers to the entire set of negative examples, which has also size N. As a result computing the cdf has complexity $O(N^2)$, which becomes quickly infeasible already for datasets with N larger than $100,000$.

To address these issues we first describe an affine calibration model that calibrates the classifier score with a simple linear function defined by only two parameters: its slope and offset, greatly reducing the required storage. Second, we show that the parameters of the affine calibration function can be obtained by normalizing the learnt classification hyper-plane without applying the classifiers on the negative data and thus bringing down the computational complexity to $O(N)$. As a result, computing and storing the calibration functions becomes feasible for very large datasets with 1M images.

5.1 Affine calibration model

Using the affine calibration model, we transform the uncalibrated classification score $s_j(\mathbf{q})$ of query $\mathbf{q}$ with a linear function

$$f_j(\mathbf{q}) = \alpha_j s_j(\mathbf{q}) + \beta_j, \tag{5}$$

where $f_j(\mathbf{q})$ is the output calibrated score, and α_j and β_j are scalar calibration parameters specific to each classifier j. In this work we use linear classifiers, hence substituting for $s_j(\mathbf{q})$ the linear classifier from (2) results also in a linear calibrated classifier

$$f_j(\mathbf{q}) = \widetilde{\mathbf{w}}_j^T \mathbf{q} + \widetilde{b}_j, \tag{6}$$

where $\widetilde{\mathbf{w}}_j = \alpha_j \mathbf{w}_j$ and $\widetilde{b}_j = \alpha_j b_j + \beta_j$. Note that the calibrated classifier (6) has the same form as the original classifier (2) and hence this representation does not require any additional storage compared to storing the original classifier. The question remains how to set the parameters α_j and β_j of the calibration function (5), which is discussed next.

5.2 Calibration by normalization

Parameters of the affine calibration function (5) could be learnt from negative training data in a similar manner to, for example, ([3]). We have tried to estimate the parameters in a similar manner by fitting a line to the tail of the cdf, however this procedure did not yield satisfactory results. In addition, as discussed above, in our case this requires running all N classifiers on all N images, which is prohibitive for large datasets. Instead, we have found that a good calibration can be obtained by normalizing the learnt hyperplane $\mathbf{w}$. In particular, we set

$$\alpha_j = \frac{1}{||\mathbf{w}_j||}, \tag{7}$$

$$\beta_j = -b_j \alpha_j, \tag{8}$$

where $\mathbf{w}_j$ and b_j are the parameters of the learnt SVM hyperplane for location j and $||\mathbf{w}||$ is the L_2 norm of $\mathbf{w}$. Given this choice of α_j and β_j the bias term in Eq. (2) cancels out and and the calibrated classification score (2) reduces to

$$f_j(\mathbf{q}) = \frac{1}{||\mathbf{w}_j||} \mathbf{w}_j^T \mathbf{q} = \widetilde{\mathbf{w}}_j^T \mathbf{q}. \tag{9}$$

Since $\mathbf{q}$ is L_2 normalized, the outcome of the particular choice of α_j and β_j is that the Eq. (9) is equivalent to computing the normalized dot-product between vectors $\mathbf{q}$ and $\mathbf{W}$. This was found to work well in image retrieval ([47]) or matching whitened HOG descriptors ([13]). In this work we investigate whether this intuition about descriptor matching can be used as a form of calibration for the learnt place-specific classifier.

Note that this form of calibration by normalization is scalable to very large datasets as it (i) requires only $O(N)$ computations offline to pre-compute the calibration parameters for each of the N learnt classifiers (equations (7) and (8)), and (ii) does not need any additional storage or computation at query time as the calibration parameters can be included in the classifier (6). In Appendix we examine the per-exemplar SVM cost and give an additional intuition why calibration by re-normalization works.

6 Memory efficient classifier representation

We learn a (calibrated) linear discriminative classifier with weight vector $\mathbf{w}_j$ and bias b_j for each image j in the database. [2] These classifier parameters become the new representation for each image. In this section we discuss how the classifier parameters can be stored in a memory efficient manner that is amenable for indexing. The goal is to apply all the learnt classifiers to the query descriptor $\mathbf{q}$

$$\mathbf{s} = \mathbf{q}^T \mathcal{W} + \mathbf{b}, \tag{10}$$

where $\mathcal{W}$ is $d \times N$ matrix storing all the learnt $\mathbf{w}_j$ classifiers as columns, $\mathbf{b}$ is a $1 \times N$ vector storing all the learnt bias values b_j, $\mathbf{q}$ is the input query descriptor, $\mathbf{s}$ is a $1 \times N$ vector of output scores for all classifiers in the database, N

[2] When the calibration by re-normalization method is used, the $\mathbf{w}_j$ contains the re-normalized weights and the bias b_j is zero. However, to cover both calibration methods, we include the bias term in the derivation in this section.

is the number of images in the database and d is the dimensionality of the image representation. As discussed in detail in section 7 we investigate two different image representations: (i) the compact Fisher vectors ([26]) and (ii) the bag-of-visual-word vectors ([47]). The learnt classifiers for these two image descriptors have different statistics and require different methods for storing and indexing. Next, we discuss the classifier representations for the two types of image representations.

Fisher vectors: The Fisher vector descriptors are not sparse, but have a relatively low-dimension $d \in \{128, 512, 2048\}$ hence it is possible to store directly the (non-sparse) matrix $\mathcal{W}$ containing the learnt classifier parameters $\mathbf{w}$. In this work we exhaustively compute the classifier scores for all images in the database (given by equation (10)) using efficient (but exact) matrix-vector multiplication routines. However, this computation can be further sped-up using product quantization indexing as described in ([24]).

Bag-of-visual-words: In the bag-of-visual-words representation, each image is represented by a high dimensional vector $\mathbf{x}$, where the dimensionality d is typically $100,000$, but the vector is very sparse with only about 2,000 non-zero entries. The learnt $\mathbf{w}_j$ are of the same (high) dimension d but are *not sparse*. As a result, directly storing the learnt classifiers becomes quickly infeasible. To illustrate this, consider a database of $N = 1,000,000$ images. Storing the original descriptors with about 2,000 non-zero entries for each image would take around 8GB. However, directly storing the learnt non-sparse $100,000 \times 1,000,000$ matrix $\mathcal{W}$ would require 400GB of memory. To address this issue we have developed an alternative indexing structure taking advantage of the dual form of the linear classifier as a sparse linear combination of a small number of support vectors ([44]). The key observation is that the number of support vectors k is significantly lower then dimensionality d of the original image descriptor. In the following we omit index j for clarity. In detail, we represent each $\mathbf{w}$ by its corresponding coefficients α_i of the linear combination of the support vectors (individual image descriptors) $\mathbf{x}_i$ such that

$$\mathbf{w} = \sum_i \alpha_i \mathbf{x}_i = \mathcal{X} \cdot \boldsymbol{\alpha}, \tag{11}$$

where α_i, the elements of vector $\boldsymbol{\alpha}$, are coefficients of the linear combination of the training data points $\mathbf{x}_i$ and the matrix $\mathcal{X}$ contains (as columns) descriptors of the entire database. Note that the vector $\boldsymbol{\alpha}$ is sparse and the number of non-zero elements depends on the number of support vectors k.

As a result, matrix $\mathcal{W}$ containing all learned classifier weights can be expressed in the dual form as

$$\mathcal{W} = \mathcal{X}\mathcal{A}, \tag{12}$$

where $\mathcal{X}$ is the (sparse) matrix of the bag-of-visual-words image descriptors and $\mathcal{A}$ is the (sparse) matrix of α coefficients, where each column corresponds to vector $\boldsymbol{\alpha}$ from (11). Instead of storing all (non-sparse) weight vectors $\mathcal{W}$, which has memory complexity $O(dN)$ where d (= 100,000) is the dimensionality of the image representation and N is the size of the database, we store two sparse matrices $\mathcal{X}$ and $\mathcal{A}$, which has memory complexity $O(mN + kN)$ where m (=2,000) is the number on non-zero elements in the original bag-of-visual-word descriptors, and k is the typical number of support vectors. In our case k is about the size of the training data which is around 500. As a result, the storage requirements are significantly reduced. For example, for a database of 1M images the dual representation requires only about 10 GB of storage compared to 400GB for directly storing classifiers $\mathcal{W}$. Note that sparsity can be imposed directly on the learnt classifiers $\mathbf{w}$ by appropriate regularization ([44]). However, we found this approach did not yield competitive results in terms of accuracy.

7 Experimental setup and implementation details

In this section we describe the experimental datasets, outline the two types of used image descriptors, and finally give implementation details of the classifier learning procedure.

7.1 Image datasets

Experiments are performed on two datasets, the Pittsburgh place recognition dataset ([19]) and the Tokyo 24/7 dataset ([50]).

Pittsburgh dataset. The first dataset **??** contains Google Street View panoramas downloaded from the Internet covering an area of 1.3×1.2 km^2 of the city of Pittsburgh (U.S.). Similar to ([8]), we generate for each panorama 12 overlapping perspective views corresponding to two different elevation angles 4° and 28° to capture both the street-level scene and the building façades. This results in a total of 24 perspective views each with 90° FOV and resolution of 960×720 pixels. In this manner we generate two versions of this dataset. The first version covers a smaller area and contains 25k perspective images. The second larger dataset contains 55k images. As a query set with known ground truth GPS positions, we use 8999 panoramas from the Google Street View research dataset ([18]), which cover approximately the same area, but were captured at a different time, and typically depict the same places from different viewpoints and under different illumination conditions. We generate a test query set such that we first select a panorama at random, and second, we generate a perspective image with a random orientation and random elevation pitch. This way we synthesize 4,000 query

test images. Both the query and database images are available upon request at ([20]).

24/7 Tokyo dataset. The 24/7 Tokyo dataset ([50]) contains Google Street View panoramas downloaded from the Internet covering an area of 1.6×1.6 km^2 of the city of Tokyo. The dataset contains 76k perspective views. The query set contains 315 query images from 105 distinct locations captured by different types of camera phones. This dataset is very challenging as each location is captured at three different times: during day, at sunset and during night. The dataset is available upon request at ([49]).

7.2 Image descriptors

We perform experiments with two types of image descriptors: the sparse high-dimensional bag-of-visual-word vectors ([47]) and the compact (not-sparse) Fisher vectors ([26]). Details of each are given next.

Bag-of-visual-word representation. We extract SURF descriptors ([5]) for each image and learn a vocabulary of 100k visual words by approximate k-means clustering ([37]) from a subset of features from $5,000$ randomly selected database images. Then, a tf-idf weighted vector ([47]) is computed for each image by assigning each descriptor to the nearest cluster center. Finally, all database vectors are normalized to have unit L_2 norm.

Fisher vectors. Following ([26]) we project the extracted 128-dimensional rootSIFT ([2]) descriptors to 64 dimensions using PCA. The projection matrix is learnt on a set of descriptors from 5,000 randomly selected database images. This has also the effect of decorrelating the rootSIFT descriptor. The 64-dimensional descriptors are then aggregated into Fisher vectors using a Gaussian mixture model with $N = 256$ components, which results in a $2 \times 256 \times 64 = 32,768$-dimensional descriptor for each image. The Gaussian mixture model is learnt from descriptors extracted from 5,000 randomly sampled database images. The high-dimensional Fisher vector descriptors are then projected down to dimension using PCA learnt from all available images in the database. The resulting low dimensional Fisher vectors are then normalized to have unit L2-norm, which we found to be important in practice.

7.3 Parameters of per-location classifier learning

To learn the exemplar support vector machine for each database image j, the positive and negative training data are constructed as follows. The *negative training set* $\mathcal{N}_j$ is obtained by: (i) finding the set of images with geographical distance greater than 200 m; (ii) sorting the images by decreasing value of similarity to image j measured by the dot product between their respective descriptors; (iii) taking the top $N = 500$ ranked images as the negative set. In other words, the negative training data consists of the hard negative images, i.e. those that are similar to image j but are far away from its geographical position, hence, cannot have the same visual content. The *positive training set* $\mathcal{P}_j$ consist of the descriptor $\mathbf{x}_j$ of the target image j.

We found that for the bag-of-visual-words representation it was useful to further expand ([9]) positive training set by close by images that view the same scene structures. These images can be identified by geometric verification ([37]) as follows. We first build a graph where each image in the database represents a node and an edge represents a spatial adjacency in the world. An edge is present if the positions of the two images are within $50m$ of each other. Then, we score each edge by the number of geometrically verified matches ([37]). Finally, we remove edges with score below a threshold of $t_m = 40$ matches. It is worth noting that the graph contains many isolated nodes. This typically indicates that the viewpoint change between two adjacent panoramas is large. For each image in the database, we include between zero and five extra positive examples that are directly connected in the graph.

For the support vector machine classifier (SVM) training we use `libsvm` ([15]). We use the same C_1 and C_2 parameters for all per-exemplar classifiers, but find the optimal value of the parameters for each image representation by a cross-validation evaluating performance on a held out query set.

For the calibration by re-normalization, we L_2 normalize the learned $\mathbf{w}_j$ using equation (9) and use this normalized vector as the new image descriptor $\mathbf{x}'_j$ for image j. At query time we compute the descriptor $\mathbf{q}$ of the query image and measure its similarity score to the learnt descriptors $\mathbf{x}'_j$ for each database image by equation (1).

For the p-value calibration, we take the learnt classifier for each database image j and compute its SVM classification score for all other database images to construct its empirical cumulative density function (4). We keep only the top 1,000 values that, in turn, represent the calibration function. At query time, given the query descriptor $\mathbf{q}$, we compute the SVM score (2) for each database image j, and compute its calibrated SVM score f_j (4).

8 Results

We evaluate the proposed per-location classifier learning approach on two different image descriptors: the bag-of-visual-words model (section 8.1) and Fisher vectors (section 8.2). We also compare the recognition accuracy of the two learnt

representations relative to their compactness measured by their memory footprint (section 8.3). Finally, we compare results to linear discriminant analysis (LDA) and whitening baselines (section 8.4), outline the main failure modes (section 8.5) and discuss the scalability of our method (section 8.6).

Since the ground truth GPS position for each query image is available, for each method we measure performance using the percentage of correctly recognized queries (Recall) similarly to, e.g., ([8, 29, 41]). We deem the query as correctly localized if at least one of the top K retrieved database images is within 20 meters from the ground truth position of the query.

8.1 Bag-of-visual-words model

Results for the bag-of-visual-words image representation are shown in table 1. Learning per-location classifiers with either calibration method (*p-val* and *w-norm*) clearly improves over the standard bag-of-visual-words baseline (BOW) that does not perform any learning. In addition, both calibration methods significantly improve over the learnt SVM classifiers without any calibration (BOW SVM no calib) underscoring the importance of calibration for the independently learnt per-location classifiers. In table 1, we also compare performance to our implementation of the confuser suppression approach (Conf. supp.) of ([29]) that, in each database image, detects and removes features that frequently appear at other far-away locations (using parameters $t = 3.5$ and $w = 70$). Our results show an improvement, specially at recall@1.

Inspecting the detailed plots in figure 4 we further note that the *p-val* calibration performs slightly better than the *w-norm* calibration for shorter top K shortlists but this effect is reversed for larger K. This could be attributed to the fact that the *p-val* calibration uses the negative data to control false positive errors, but has less control over false negatives, as discussed in section 4.3.

In figure 6 we visualize the learnt SVM weights on BOW for *p-val*. We visualize the contribution of each feature to the SVM score for the corresponding query image. Red circles represent features with negative weights while green circles correspond to features with positive weights. The area of each circle is proportional to the contribution of the corresponding feature to the SVM score. For instance for the left figure notice that the correctly localized queries (c) contain more green colored features than queries from other places (b) and (a). Query (b) gets a high score because the building has orange and white stripes similar to the the sun-blinds of the bakery, which are features that also have large positive weights in the query image (c) of the correct place. In the top row we visualize the calibration of raw SVM score for three different queries. The calibration function of the target

Method:	25k Pittsburgh				
recall@K [%]	1	2	5	10	20
BOW SVM no calib.	6.4	8.1	13.5	17.5	20.5
BOW	28.7	35.7	45.8	53.7	61.5
BOW Conf. supp [29]	29.6	37.3	48.9	59.3	69.2
BOW w-norm	31.8	38.7	49.7	**60.2**	**69.4**
BOW p-val	**33.0**	**40.3**	**50.2**	58.7	66.4

Table 1: **Evaluation of the learnt bag-of-visual-words representation on the Pittsburgh 25k dataset.** The table shows the fraction of correctly recognized queries (recall@K) for the different values of $K \in \{1, 2, 5, 10, 20\}$ retrieved database images. The learnt representations (BOW w-norm and BOW p-val) outperform the raw bag-of-visual-words baseline (BOW) as well as the learnt representation without calibration (BOW SVM no calib).

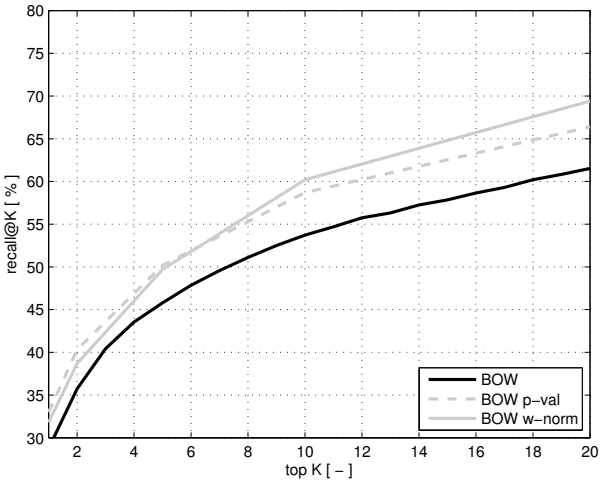


Fig. 4: **Evaluation of the learnt bag-of-visual-words representation on the Pittsburgh 25k ([19]) dataset.** The graph shows the fraction of correctly recognized queries (recall@K, y-axis) vs. the number of top K retrieved database images for the raw bag-of-visual-words baseline (BOW) and the learnt representation with two different calibration methods (p-val and w-norm).

image j is shown in the blue and the corresponding SVM scores of the three queries are denoted by red circles. Notice that both images (b) and (c) have high calibrated score even their respective SVM score was different.

Finally, examples of correctly and incorrectly localized queries are shown in figure 9.

8.2 Fisher vectors

Results of the proposed per-location learning method for the Fisher vector image representation for different dimensions

recall@K [%]	1	2	5	10	20
Method / Dataset:	25k Pittsburgh				
FV128	33.6	41.8	52.0	59.8	67.7
FV128 w-norm	**38.3**	**47.5**	**57.7**	**65.8**	**72.7**
FV512	44.3	51.7	61.4	68.7	75.2
FV512 w-norm	**47.6**	**55.4**	**65.1**	**72.4**	**78.8**
FV2048	46.9	54.1	63.8	70.5	76.8
FV2048 w-norm	**50.2**	**57.3**	**67.0**	**73.8**	**78.0**
FV16384	45.3	54.1	63.8	69.4	75.3
FV16384 w-norm	**49.3**	**56.0**	**65.9**	**72.5**	**76.8**
	55k Pittsburgh				
FV128	10.9	14.1	20.2	26.4	33.2
FV128 w-norm	**13.5**	**17.7**	**25.0**	**31.8**	**39.0**
FV512	17.3	21.1	28.4	34.2	40.3
FV512 w-norm	**19.8**	**25.1**	**32.7**	**38.7**	**46.0**
FV2048	19.2	23.5	29.9	35.2	41.9
FV2048 w-norm	**20.8**	**25.9**	**33.1**	**38.7**	**45.9**
	24/7 Tokyo				
FV128	14.2	20.0	27.9	34.2	41.5
FV128 w-norm	**16.9**	**22.0**	**29.6**	**37.2**	**44.8**
FV512	35.2	40.3	43.8	48.2	57.1
FV512 w-norm	**36.1**	**42.0**	**46.8**	**52.8**	**61.4**
FV2048	37.4	42.5	48.5	53.9	58.7
FV2048 w-norm	**42.9**	**46.7**	**52.8**	**58.8**	**66.7**
FV4096	42.9	46.3	54.0	59.0	64.8
FV4096 w-norm	**44.3**	**47.1**	**54.7**	**61.1**	**66.5**

Table 2: **Evaluation of the learnt Fisher vector representation on the Pittsburgh ([19]) and 24/7 Tokyo ([50]) datasets.** The table shows the fraction of correctly recognized queries (recall@K) for the different values of $K \in \{1, 2, 5, 10, 20\}$ retrieved database images. The learnt Fisher vector representation (FV *w-norm*) consistently improves over the standard Fisher vector matching baseline (FV) for all target dimensions.

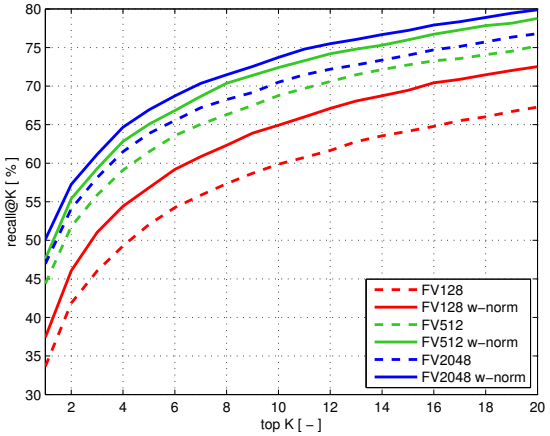


Fig. 5: **Evaluation of the learnt Fisher vector representation on the Pittsburgh 25k ([19]) dataset.** The graph shows the fraction of correctly recognized queries (recall@K, y-axis) vs. the number of top K retrieved database images for the raw Fisher vector baseline (FV) for different dimensions compared to the learnt representation (*w-norm*). Note the consistent improvements over all lengths of shortlist K for all dimensions.

are shown in table 2 and figure 5. Similar to bag-of-visual-words, the learnt representation (w-norm) significantly improves the place recognition performance over the baseline Fisher vector (FV) matching without learning. The improvements are consistent across different lengths of shortlist K and for different dimensionality of the Fisher vector representation. We report results only for the w-norm calibration as we found that the p-val calibration did not perform well for the learnt Fisher vector classifiers (top 1 recall of 25.3% compared to baseline performance of 33.6% for dimension 128). When examining the results we have observed that for bag-of-visual-words the cdf estimated on the database well represents the scores of (unseen) negative query images at test time. However, this is not the case for Fisher vectors where estimated cdf on the database does not represent well the scores of negative query images at test time. The scores of (unseen) negative query images often fall outside of the estimated cdf or at the very tail that is only sparsely sampled. As a result the estimated query image p-values for Fisher vectors are often over-confident and incorrect. Notice that the proposed per-location learning method consistently improves performance over the raw Fisher vector descriptors on the larger Pittsburgh 55k dataset and the challenging 24/7 Tokyo dataset (76k images). Examples of correctly and incorrectly localized queries are shown in figure 8. Next, we compare the performance of the two learnt representations relative to their memory footprints.

8.3 Analysis of recognition accuracy vs. compactness

Here we analyze the recognition accuracy of the learnt representations vs. their compactness measured by their memory footprint on the Pittsburgh 25k image dataset. Ideally, we wish to learn a more compact representation, that still improves the recognition accuracy. However, usually there is a trade-off between the discriminative power of the representation and its size, where having a more compact representation reduces the recognition accuracy ([26]). We observe a similar behavior but our learnt representation results in a higher recognition accuracy for a given size, or alternatively, significantly reduces the size of the representation for a given accuracy. The results are summarized in figure 7. The figure shows the recognition performance (y-axis) for the different dimensionality of the Fisher vector representation, which corresponds to different memory footprints (x-axis). For example, for $d = 128$ the memory footprint is about 24 MB, whereas for $d = 2048$ the memory footprint is about 384 MB. Note that the x-axis is in log-scale. The bag-of-visual-words representation has a fixed dimensionality (and fixed memory footprint) and hence each bag-

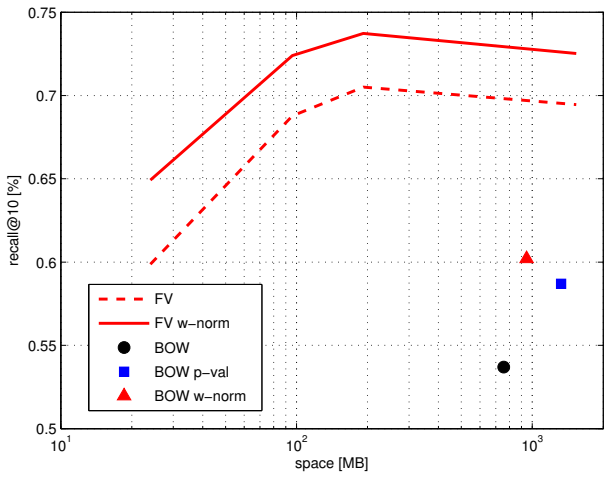


Fig. 7: **The recognition performance vs. the memory requirements for the Pittsburgh 25k dataset.** The fraction of correctly localized queries at the top 10 retrieved images (y-axis) vs. the memory footprint (x-axis) for the different representations. For Fisher vectors, the learnt descriptor (FV *w-renorm*) clearly outperforms the raw Fisher vector descriptor (FV) for all dimensions corresponding to different memory footprints (x-axis). Learnt per-location representations for the bag-of-visual-words model (BOW p-val and BOW w-norm) also improve performance over the raw bag-of-visual-words (BOW). However, the Fisher vectors provide much better recognition performance for the same memory footprint.

of-visual-words method is shown only as a single point on the graph. For Fisher vectors, the results demonstrate that for a given level of accuracy (y-axis) the proposed method learns a more compact (lower-dimensional) representation (x-axis). For example, our learnt 128-dimensional descriptor (memory footprint of 24 MB) achieves a similar accuracy (around 65%) as the 256-dimensional raw Fisher descriptor (memory footprint of 51MB, interpolated from figure 7). This corresponds to 50% memory savings for the same level of recognition performance. Note that similar to ([26]), we observe decrease in performance at high-dimensions for both the FV baseline and our method. The results also demonstrate the benefits of using the compact FV descriptors compared to the bag-of-visual-words baseline achieving significantly better recognition accuracy for a similar memory footprint (figure **??**).

8.4 Comparison to linear discriminant analysis (LDA) and whitening baselines

We have compared our method to the linear discriminant analysis (LDA) ([4, 21, 17]) and whitening ([25]) baselines. Results are reported on the Pittsburgh $25k$ dataset. The LDA baseline finds a discriminative linear projection of the feature space by minimizing an Euclidean loss, rather than hinge loss used in our work. In detail, following ([4]) we have used all available database to learn the covariance matrix and used calibrated LDA score (see ([4]) eq. 11) to obtain a classifier for each database image. We have applied the LDA method on the 128-dimensional Fisher vector descriptor but have obtained significantly worse performance (31.9% for recall@1) than our method (recall@1 of 38.3%). We believe the better performance of our method can be attributed to (i) the use of hinge-loss and (ii) training using the top scoring hard negative examples that are specific for each place.

Next, we compare results to PCA compression followed by whitening as suggested in ([25]). For bag-of-visual-words, we follow ([25]) and compare performance to PCA whitening to a target dimension of 4096. We have observed performance drop compared to the raw bag-of-visual-words baseline (28.7% to 26.1% for recall@1). We hypothesize this could be attributed to the large dictionary size used in our work (100k), whereas ([25]) report improved results for single dictionary whitening only for dictionaries of up to 32k visual words. Finally, we have also applied PCA whitening on Fisher vector descriptors of dimensions 128, 512 and 2048, but have not observed significant improvements over the baseline raw descriptors. In fact, for the highest dimension (2048) we have observed a performance drop (49.6% to 41.3%), which could be attributed to amplification of low-energy noise as also reported in ([25]).

8.5 Analysis of improvement and failure cases

We have examined the improvement and failures of the w-norm method w.r.t. the Fisher vector baseline on the Pittsburgh 25k dataset. We analyzed the cases for which the w-norm method improves the rank of the first true positive compared to the baseline and for which the rank of the first true positive is made worse. In detail, considering a shortlist of the size 20 we want to identify when: (i) an image with the rank of 20 is attracted into the short list (improvement), and (ii) an image with the rank of ≤ 20 is pushed out of the short list using our method (failure).

We observe that in 237 cases a low-ranked true positive image by the baseline (ranked $40 - 70$) is attracted into the short list by the w-norm method, resulting in an improvement. Note that in many other cases our method improves ranking but here we only count the cases for which the baseline method does not have any true positives in the top 20

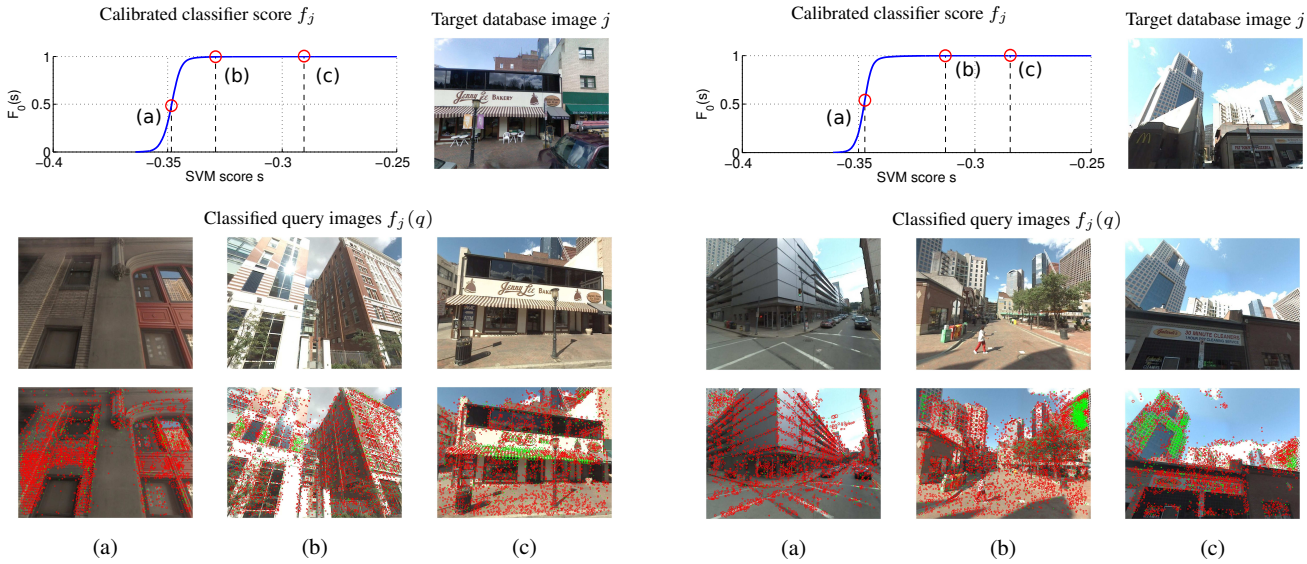


Fig. 6: **A visualization of learnt feature weights for two database images. In each panel:** *first row:* (Right) Target database image j. (Left) Cumulative density function (or calibrated score) learnt for the SVM scores of the corresponding classifier f_j; three query images displayed on the *second row* are represented by their SVM scores and cdf values $F_0(s)$, denoted (a)-(c) on the graph. *Third row:* A visualization of the contribution of each feature to the SVM score for the corresponding query image. Red circles represent features with negative weights while green circles correspond to features with positive weights. The area of each circle is proportional to the contribution of the corresponding feature to the SVM score. Notice that the correctly localized queries (c) contain more green colored features than queries from other places (b) and (a). Please note also that the calibration *cdf*s in the left and right panel are similar but not identical. *Left panel:* Query (b) gets a high score because the building has orange and white stripes similar to the the sun-blinds of the bakery, which are features that also have large positive weights in the query image (c) of the correct place. *Right panel:* Query (b) is in fact also an image of the same location with a portion of the left skyscraper in the target image detected in the upper left corner and the side of the rightmost building in the target image detected in the top right corner. Both are clearly detected by the method as indicated by a large quantity of green circles in the corresponding regions.

short-list. On the other hand, in 39 cases our method downranks correct images appearing in the top 20 shortlist. However, these images are typically downranked only slightly and still occur in the top 40 shortlist.

Finally, we observe that the downranking typically occurs on hard examples where the baseline performance is already bad. When visually inspecting the failure cases we observed that our method typically fails on queries containing a large amount of clouds or vegetation and images containing narrow streets and tunnels. Our method sometimes retrieves images capturing the same building from a different side or a large distance.

8.6 Scalability

In the offline stage, our method collects hard negative examples for each location in the database, which are consequently used to train exemplar SVM classifiers. As only a constant number of examples (1-5 positives and 500 negatives) is used to train each per-location classifier the overall complexity of training is linear, $O(N)$, i.e. we need to train one classifier (with constant training time) for each of N images in the database. The bottleneck of the offline stage is collecting the negative examples that is quadratic $O(N^2)$ in the database size. In other words, for each of N database images, we need to find the top 500 most similar negatives among all N database images. However, we believe that even finding negatives can be scaled-up to very large datasets with standard compression techniques such as product quantization (PQ) ([24]) combined with sub-linear approximate nearest neighbor search ([35]).

At query time our method needs to compute the calibrated e-SVM score (equation (2)) of the query for each image in the database. In the case of w-norm method the calibration weights can be included in the classifier weight matrix, as discussed in section 6. For the p-val calibration method, each e-SVM score must be calibrated using K stored values of the non-parametric CDF model. This requires a search for the two closest values and subsequent interpolation, which yields complexity of $O(N \log K)$. Since K is only a constant both the w-norm and p-val methods have a

Fig. 8: **Examples of correctly and incorrectly localized queries for the learnt bag-of-visual-words representation.** Each example shows a query image (left) together with correct (green) and incorrect (red) matches from the database obtained by learnt bag-of-visual-words representation *p-val* method (top) and the standard bag-of-visual-words baseline (bottom). Note that the proposed method is able to recognize the place depicted in the query image despite changes in viewpoint, illumination and partial occlusion by other objects (trees, lamps) and buildings. Note also that bag-of-visual-words baseline is often confused by repeating patterns on facades and walls.

linear time complexity (in the size of the database) at query time but with different constants. However, in practice the constant in the p-val method can be quite large. The actual running time per query is $340ms$ for the bag-of-visual-words representation with p-val calibration and $3ms$ for the FV128 descriptor with w-norm calibration. Both timings are on the 25k Pittsburgh dataset on a desktop with CPU Intel Xenon E5 using a single thread. Hence in practice, the p-val method may be scalable only to medium size datasets. For the w-norm method, the query time can be further sped-up using sub-linear approximate nearest neighbor search ([35]) on compressed descriptors ([24]), making the method scalable to very large datasets.

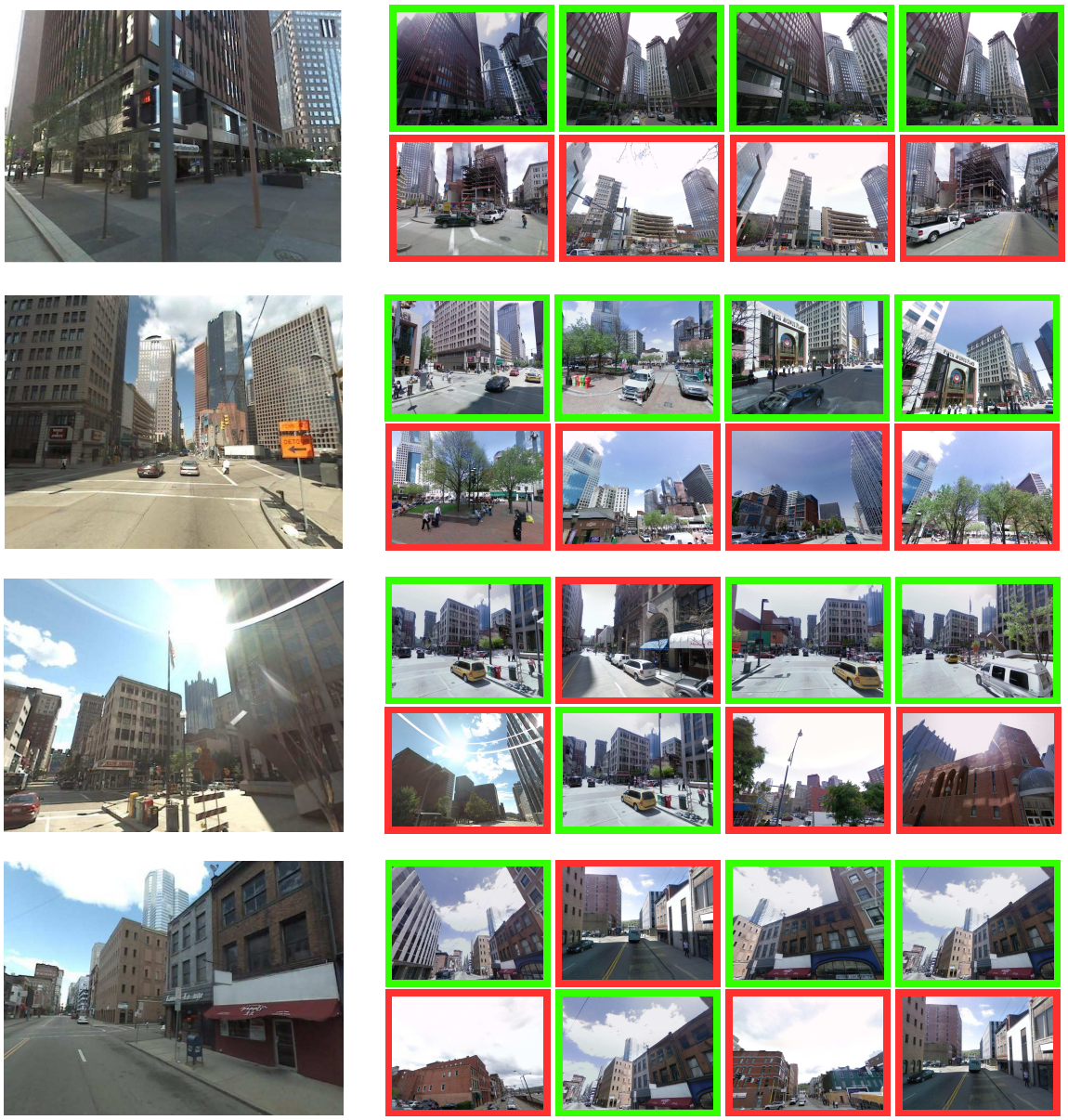


Fig. 9: **Examples of correctly and incorrectly localized queries for the learnt Fisher vector representation.** Each example shows a query image (left) together with correct (green) and incorrect (red) matches from the database obtained by the learnt Fisher vector representation *w-norm* method (top) and the standard Fisher vector baseline (bottom) for dimension 128. Note that the proposed method is able to recognize the place depicted in the query image despite changes in viewpoint, illumination and partial occlusion by other objects (trees, lamps) and buildings. Note that the baseline methods often finds images depicting the same buildings but in a distance whereas our learnt representation often finds a closer view better matching the content of the query.

9 Conclusions

We have shown that place recognition can be cast as a classification problem and have used geotags as a readily-available supervision to train an ensemble of classifiers, one for each location in the database. As only few positive examples are available for each location, we have developed two procedures to calibrate the output of each classifier without the need for additional positive training data. We have shown that learning per-location representations improves the place recognition performance over the raw bag-of-visual-words and Fisher vector matching baselines. The developed calibration methods are not specific to place recognition and can

be useful for other per-exemplar classification tasks, where only a small number of positive examples are available ([34]).

Acknowledgements This work was supported by the MSR-INRIA laboratory, the EIT-ICT labs, Google, the ERC project LEAP and the EC project RVO13000 - Conceptual development of research organization. Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Air Force Research Laboratory, contract FA8650-12-C7212. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, AFRL or the U.S. Government.

References

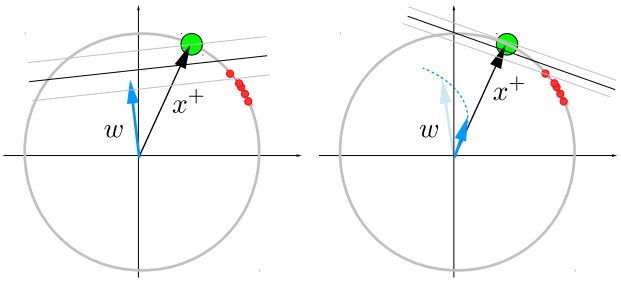
1. S. Agarwal, N. Snavely, I. Simon, S. Seitz, and R. Szeliski. Building Rome in a day. In *ICCV*, pages 72–79, 2009.
2. R. Arandjelović and A. Zisserman. Three things everyone should know to improve object retrieval. In *IEEE PAMI*, 2012.
3. M. Aubry, D. Maturana, A. Efros, B. Russell, and J. Sivic. Seeing 3D chairs: exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR*, 2014.
4. M. Aubry, B. Russell, and J. Sivic. Painting-to-3D model alignment via discriminative visual elements. *ACM Transactions on Graphics*, 2014.
5. H. Bay, T. Tuytelaars, and L. Van Gool. SURF: Speeded up robust features. In *ECCV*, 2006.
6. Song Cao and Noah Snavely. Graph-based discriminative learning for location recognition. In *CVPR*, pages 700–707. IEEE, 2013.
7. G. Casella and R.L. Berger. Statistical inference. 2001.
8. David Chen, Georges Baatz, Köser, Sam Tsai, Ramakrishna Vedantham, Timo Pylvanainen, Kimmo Roimela, Xin Chen, Jeff Bach, Marc Pollefeys, Bernd Girod, and Radek Grzeszczuk. City-scale landmark identification on mobile devices. In *CVPR*, 2011.
9. O. Chum, J. Philbin, J. Sivic, M. Isard, and A. Zisserman. Total recall: Automatic query expansion with a generative feature model for object retrieval. In *ICCV*, 2007.
10. G. Csurka, C. Bray, C. Dance, and L. Fan. Visual categorization with bags of keypoints. In *Workshop on Statistical Learning in Computer Vision, ECCV*, pages 1–22, 2004.
11. M. Cummins and P. Newman. Highly scalable appearance-only SLAM - FAB-MAP 2.0. In *Proceedings of Robotics: Science and Systems*, Seattle, USA, June 2009.
12. N. Dalal and B Triggs. Histogram of oriented gradients for human detection. In *CVPR*, 2005.
13. Carl Doersch, Abhinav Gupta, and Alexei A. Efros. Mid-level visual element discovery as discriminative mode seeking. In *NIPS*, 2013.
14. Carl Doersch, Saurabh Singh, Abhinav Gupta, Josef Sivic, and Alexei A. Efros. What makes paris look like paris? *SIGGRAPH*, 31(4), 2012.
15. Rong-En Fan, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. LIBLINEAR: A library for large linear classification. *J. Machine Learning Research*, 9:1871–1874, 2008.
16. M. Gebel and C. Weihs. Calibrating classifier scores into probabilities. *Advances in Data Analysis*, pages 141–148, 2007.
17. M. Gharbi, T. Malisiewicz, S. Paris, and F. Durand. A Gaussian approximation of feature space for fast image similarity. Technical report, MIT, 2012.
18. Google. Icmla 2011 streetview recognition challenge, 2011.
19. P. Gronat, G. Obozinski, J. Sivic, and T. Pajdla. Learning and calibrating per-location classifiers for visual place recognition. In *CVPR*, 2013.
20. Petr Gronát. Project webpage: Learning and calibrating per-location classifiers for visual place recognition, 2015.
21. B. Hariharan, J. Malik, and D. Ramanan. Discriminative decorrelation for clustering and classification. In *ECCV*, 2012.
22. James Hays and Alexei A. Efros. im2gps: estimating geographic information from a single image. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2008.
23. Arnold Irschara, Christopher Zach, Jan-Michael Frahm, and Horst Bischof. From structure-from-motion point clouds to fast location recognition. In *CVPR*, 2009.
24. H. Jegou, M. Douze, and C. Schmid. Product quantization for nearest neighbor search. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):117–128, 2011.
25. Hervé Jégou and Ondřej Chum. Negative evidences and cooccurences in image retrieval: The benefit of pca and whitening. In *ECCV*, pages 774–787, 2012.
26. Hervé Jégou, Florent Perronnin, Matthijs Douze, Jorge Sánchez, Patrick Pérez, and Cordelia Schmid. Aggregating local image descriptors into compact codes. *IEEE PAMI*, 34:1704–1716, 2012.
27. E Kalogerakis, O. Vesselova, J. Hays, A. Efros, and A. Hertzmann. Image sequence geolocation with human travel priors. In *IEEE 12th International Conference on Computer Vision (ICCV)*, pages 253–260, 2009.
28. B. Klingner, D. Martin, and J. Roseborough. Street view motion-from-structure-from-motion. In *ICCV*, 2013.
29. J. Knopp, J. Sivic, and T. Pajdla. Avoidng confusing features in place recognition. In *ECCV*, 2010.
30. Y. Li, D. Crandall, and D. Huttenlocher. Landmark classification in large-scale image collections. In *ICCV*, 2009.
31. Y. Li, N. Snavely, and D. Huttenlocher. Location recognition using prioritized feature matching. In *ECCV*, 2010.
32. Y. Li, N. Snavely, D. Huttenlocher, and P. Fua. Worldwide pose estimation using 3d point clouds. In *ECCV*, 2012.
33. D. Lowe. Distinctive image features from scale-invariant keypoints. *IJCV*, 60(2):91–110, 2004.
34. Tomasz Malisiewicz, Abhinav Gupta, and Alexei A. Efros. Ensemble of exemplar-svms for object detection and beyond. In *ICCV*, 2011.
35. Marius Muja and David G. Lowe. Scalable nearest neighbor algorithms for high dimensional data. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 36, 2014.
36. D. Nister and H. Stewenius. Scalable recognition with a vocabulary tree. In *CVPR*, 2006.
37. J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman. Object retrieval with large vocabularies and fast spatial matching. In *CVPR*, 2007.
38. J. Philbin, J. Sivic, and A. Zisserman. Geometric latent dirichlet allocation on a matching graph for large-scale image datasets. *IJCV*, 2010.
39. J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In *Advances in Large Margin Classifiers*, 1999.
40. T. Sattler, B. Leibe, and L. Kobbelt. Improving image-based localization by active correspondence search. In *ECCV*, 2012.
41. Torsten Sattler, Tobias Weyand, Bastian Leibe, and Leif Kobbelt. Image retrieval for image-based localization revisited. In *Proc. BMVC*, 2012.
42. Walter Scheirer, Neeraj Kumar, Peter N. Belhumeur, and Terrance E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *CVPR*, 2012.
43. G. Schindler, M. Brown, and R. Szeliski. City-scale location recognition. In *CVPR*, 2007.

44. Bernhard Scholkopf and Alex Smola. *Learning with kernels*. MIT press, Cambridge, 2002.
45. A. Shrivastava, T. Malisiewicz, A. Gupta, and A. A. Efros. Data-driven visual similarity for cross-domain image matching. In *SIGGRAPH ASIA*, 2011.
46. S. Singh, A. Gupta, and A. A. Efros. Unsupervised discovery of mid-level discriminative patches. In *ECCV*, 2012.
47. J. Sivic and A. Zisserman. Video Google: A text retrieval approach to object matching in videos. In *ICCV*, 2003.
48. J. Tighe and S. Lazebnik. Finding things: Image parsing with regions and per-exemplar detectors. In *CVPR*, 2013.
49. A. Torii. Project webpage: 24/7 place recognition by view synthesis, 2015.
50. A. Torii, R. Arandjelović, J. Sivic, M. Okutomi, and T. Pajdla. 24/7 place recognition by view synthesis. In *CVPR*, 2015.
51. A. Torii, J. Sivic, and T. Pajdla. Visual localization by linear combination of image descriptors. In *IEEE Workshop on Mobile Vision*, 2011.
52. A. Torii, J. Sivic, T. Pajdla, and M. Okutomi. Visual place recognition with repetitive structures. In *CVPR*, 2013.
53. P. Turcot and D. Lowe. Better matching with fewer features: The selection of useful features in large database recognition problem. In *WS-LAVD, ICCV*, 2009.
54. B. Zadrozny and C. Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *ACM SIGKDD*, 2002.
55. A. Zamir and M. Shah. Accurate image localization based on google maps street view. In *ECCV*, 2010.

Appendix

In section 8 we show that the simple calibration by normalization often results in surprisingly good place recognition performance without the need for any additional positive or negative calibration data. In this appendix, we give a possible explanation why this simple calibration works. We focus on the case of a single positive training example, i.e. when training set $\mathcal{P} = \mathbf{x}^+$, which is the typical case for place recognition where only one positive example is available for each place. The analysis holds also for the case of multiple expanded positive examples as in our case the positive examples are coming from the same database of Street View images, and hence have very similar statistics (illumination, capturing conditions, the same camera, etc.).

In particular, we first analyze the SVM objective and show that the learnt hyperplane $\mathbf{w}$ can be interpreted as a new descriptor $\mathbf{x}^*$ that replaces the original positive example $\mathbf{x}^+$ and is re-weighted to increase its separation from the negative data. Second, we show that when $\mathbf{x}^*$ is normalized, i.e. $\mathbf{x}^* = \frac{\mathbf{w}}{||\mathbf{w}||}$, the dot-product $\mathbf{q}^T\mathbf{x}^*$ corresponds to measuring the cosine of the angle between the (normalized) query descriptor $\mathbf{q}$ and the new descriptor $\mathbf{x}^*$, which was found to work well in the literature for descriptor matching, as discussed in section 5.2. The two steps are given next.



(a) $C_2 > 0$ (b) $C_2 \to 0$

Fig. 10: **An illustration of the effect of decreasing parameter C_2 in the exemplar support vector machine objective.** The positive exemplar $\mathbf{x}^+$ is shown in green. The negative data points are shown in red. All training data is L2 normalized to lie on a hyper-sphere. (a) For $C_2 > 0$, the normal $\mathbf{w}$ of the optimal hyper-plane moves away from the direction given by the positive example $\mathbf{x}^+$ in a manner that reduces the loss on the negative data. (b) As the parameter C_2 decreases the learnt $\mathbf{w}$ becomes parallel to the positive training example $\mathbf{x}^+$ and its magnitude $||\mathbf{w}||$ goes to 0.

Analysis of per-exemplar SVM objective

For a single positive example $\mathcal{P} = \mathbf{x}^+$, the per-exemplar SVM objective (3) can be written as

$$\Omega(\mathbf{w}, b) = ||\mathbf{w}||^2 + C_1 \cdot h(\mathbf{w}^T\mathbf{x}^+ + b) + C_2 \sum_{\mathbf{x}\in\mathcal{N}} h(-\mathbf{w}^T\mathbf{x} - b). \quad (13)$$

In the following, we analyze the objective (13) and provide intuition why *re-normalized* weight vector $\mathbf{w}$ can be interpreted as a new descriptor. In particular, we show first that when the weight C_2 of the negative data in objective (13) goes to zero the learnt normalized $\widetilde{\mathbf{w}}$ is identical to the original positive training data point $\mathbf{x}^+$. Second, when $C_2 > 0$, the learnt vector $\widetilde{\mathbf{w}}$ moves away from the positive vector $\mathbf{x}^+$ to increase its separation from the negative data. The two cases are detailed next.

Case I: $C_2 \to 0$. The goal is to show that when the weight C_2 of the negative data in objective (13) goes towards zero, the resulting hyperplane vector $\mathbf{w}$ is parallel with the vector of positive training descriptor $\mathbf{x}^+$. When $\mathbf{w}$ is normalized to have unit L2 norm the two vectors are identical. First, let us decompose $\mathbf{w}$ into parallel and orthogonal part with respect to the positive training data point $\mathbf{x}^+$, i.e. $w = \mathbf{w}^\perp + \mathbf{w}^{||}$, where $(\mathbf{w}^\perp)^T\mathbf{x}^+ = 0$. Next, we observe that when the weight of the negative data diminishes ($C_2 \to 0$), any non-zero component $\mathbf{w}^\perp$ will increase the value of the objective. As a result, for $C_2 \to 0$ the objective is minimized by $\mathbf{w}^{||}$, i.e. the optimal $\mathbf{w}$ is parallel with $\mathbf{x}^+$.

In detail, for $\mathbf{w} = \mathbf{w}^{\perp} + \mathbf{w}^{||}$, the objective (3) can be written as

$$||\mathbf{w}^{\perp} + \mathbf{w}^{||}||^2 + C_1 \cdot h\left((\mathbf{w}^{\perp} + \mathbf{w}^{||})^T \mathbf{x}^{+} + b\right) + C_2 \sum_{\mathbf{x} \in \mathcal{N}} h\left(-(\mathbf{w}^{\perp} + \mathbf{w}^{||})^T \mathbf{x} - b\right). \tag{14}$$

Note that the orthogonal part $\mathbf{w}^{\perp}$ does not change the value of the second term in (14) because $(\mathbf{w}^{\perp} + \mathbf{w}^{||})^T \mathbf{x}^{+} = (\mathbf{w}^{||})^T \mathbf{x}^{+}$, and hence (14) reduces to

$$||\mathbf{w}^{\perp} + \mathbf{w}^{||}||^2 + C_1 \cdot h\left(\mathbf{w}^{||T} \mathbf{x}^{+} + b_j\right) + C_2 \sum_{\mathbf{x} \in \mathcal{N}} h\left(-(\mathbf{w}^{\perp} + \mathbf{w}^{||})^T \mathbf{x} - b\right). \tag{15}$$

In the limit case as $C_2 \rightarrow 0$ any non-zero component $\mathbf{w}^{\perp}$ will increase the value of the objective (15). This can be seen by noting that the third term vanishes when $C_2 \rightarrow 0$ and hence the objective is dominated by the first two terms. Further, the second term in (15) is independent of $\mathbf{w}^{\perp}$. Finally, the first term will always increase for any non-zero value of $w^{\perp}$ as $||\mathbf{w}^{\perp} + \mathbf{w}^{||}||^2 \geq ||\mathbf{w}^{||}||$ for any $\mathbf{w}^{\perp} \neq 0$.

As a result, in the limit case when $C_2 \rightarrow 0$ the optimal $\mathbf{w}$ is parallel with $\mathbf{x}^{+}$. Note also, that when C_2 is exactly equal to zero, $C_2 = 0$, the optimal $\mathbf{w}$ vanishes, i.e. the objective (15) is minimized by trivial solution $||\mathbf{w}|| = 0$ and $b = -1$. The effect of decreasing the parameter C_2 is illustrated in figure 10.

Case II: $C_2 > 0$. When the weight C_2 of the negative data in the objective (15) increases the direction of the optimal w will be different from $\mathbf{w}^{||}$ and will change to take into account the loss on the negative data points. Explicitly writing the hinge-loss $h(x) = \max(1 - x, 0)$ in the last term of (15), we see that $\mathbf{w}$ will move in the direction that reduces $\sum_{\mathbf{x} \in \mathcal{N}} \max\left(1 + \mathbf{w}^T x + b, 0\right)$, i.e. that reduces the dot product $\mathbf{w}^T \mathbf{x}$ on the negative examples that are active (support vectors).

The need for normalization of $\mathbf{w}$

Above we have shown that the learnt hyperplane $\mathbf{w}$ moves away from the positive example $\mathbf{x}^{+}$ in a manner that reduces the loss on the negative data. The aim is to use this learnt vector $\mathbf{w}$ as a new descriptor $\mathbf{x}^{*}$ replacing the original positive example $\mathbf{x}^{+}$. However, we wish to measure the cosine of the angle between the the new descriptor $\mathbf{x}^{*}$and the query image $\mathbf{q}$. This is equivalent to the normalized dot product, hence the vector $\mathbf{w}$ needs to be normalized.