Int J Comput Vis (2017) 121:126–148
DOI 10.1007/s11263-016-0927-0

Image Annotation by Propagating Labels from Semantic Neighbourhoods

Yashaswi Verma[1] · **C. V. Jawahar**[1]

Received: 24 April 2015 / Accepted: 23 June 2016 / Published online: 12 July 2016

Abstract Automatic image annotation aims at predicting a set of semantic labels for an image. Because of large annotation vocabulary, there exist large variations in the number of images corresponding to different labels ("class-imbalance"). Additionally, due to the limitations of human annotation, several images are not annotated with all the relevant labels ("incomplete-labelling"). These two issues affect the performance of most of the existing image annotation models. In this work, we propose 2-pass k-nearest neighbour (2PKNN) algorithm. It is a two-step variant of the classical k-nearest neighbour algorithm, that tries to address these issues in the image annotation task. The first step of 2PKNN uses "image-to-label" similarities, while the second step uses "image-to-image" similarities, thus combining the benefits of both. We also propose a metric learning framework over 2PKNN. This is done in a large margin set-up by generalizing a well-known (single-label) classification metric learning algorithm for multi-label data. In addition to the features provided by Guillaumin et al. (2009) that are used by almost all the recent image annotation methods, we benchmark using new features that include features extracted from a generic convolutional neural network model and those computed using modern encoding techniques. We also learn linear and kernelized cross-modal embeddings over different feature combinations to reduce semantic gap between visual features and textual labels. Extensive evaluations on four image annotation datasets (Corel-5K, ESP-Game, IAPR-TC12 and MIRFlickr-25K) demonstrate that our method achieves promising results, and establishes a new state-of-the-art on the prevailing image annotation datasets.

Keywords Image annotation · Nearest neighbour · Metric learning · Cross-media analysis

1 Introduction

Automatic image annotation is a labelling problem that has potential applications in image classification (Wang et al. 2009), image retrieval (Feng et al. 2004; Makadia et al. 2008, 2010; Guillaumin et al. 2009), image caption generation (Gupta et al. 2012), etc. Given an (unseen) image, the goal of image annotation is to predict a set of textual labels describing the semantics of that image. Over the last decade, the outburst of multimedia content on the Internet as well as in personal collections has raised the demands for auto-annotation methods, thus making it an active area of research (Feng et al. 2004; Carneiro et al. 2007; Xiang et al. 2009; Guillaumin et al. 2009; Makadia et al. 2008, 2010; Zhang et al. 2010; Verma and Jawahar 2012; Fu et al. 2012; Verma and Jawahar 2013; Chen et al. 2013; Ballan et al. 2014; Moran and Lavrenko 2014; Murthy et al. 2014; Kalayeh et al. 2014).

In the past, several methods have been proposed for image auto-annotation that try to model image-to-image, image-to-label and label-to-label similarities. Our work falls under the category of supervised annotation models such as those listed above that work with large annotation vocabularies consisting of few hundreds of labels. Among these, the nearest neighbour based methods such as Makadia et al. (2008, 2010), Guillaumin et al. (2009), Verma and Jawahar (2012)

Communicated by Shin'ichi Satoh.

✉ Yashaswi Verma
yashaswi.verma@research.iiit.ac.in

C. V. Jawahar
jawahar@iiit.ac.in

[1] Center for Visual Information Technology, IIIT, Hyderabad, India

have been found to give some of the best results despite their simplicity. The intuition is that "similar images share similar labels" (Makadia et al. 2008, 2010). In most of the existing approaches, this similarity is determined using only visual features. In the nearest neighbour based scenario, since the labels co-occurring in an image are considered together, visual similarity can also handle correlations among labels to some extent. However, it fails to address the two important issues of "class-imbalance" (large variations in the frequency of different labels) and "incomplete-labelling" (many images are not annotated with all the relevant labels from the vocabulary) that are prevalent in the popular annotation datasets as well as real-world databases. To address these issues in the nearest neighbour based set-up, one needs to ensure that (a) for a given image, the (subset of) training images that are considered for label prediction/propagation should not have large variations in the frequency of different labels, and (b) the comparison criteria between two images should make use of both image-to-label and image-to-image similarities (as discussed above, image-to-image similarities can partially capture label-to-label similarities in the nearest neighbour based scenario). With this motivation, we present a two-step variant of the classical k-nearest neighbour (kNN) algorithm that fulfills both these requirements. We call this 2-pass k-nearest neighbour (2PKNN) algorithm. As part of the 2PKNN algorithm, for an image, we say that its few nearest neighbours from a given class constitute its *semantic neighbourhood* with respect to that class, and these neighbours are its *semantic neighbours*. Based on the above discussed intuition of Makadia et al. (2008, 2010) that similar images share similar labels, we hypothesize that the semantic neighbours of an image from a particular class are the samples that are visually and hence semantically most related with that image with respect to that class. Now, given a new image, in the first step of 2PKNN we identify its semantic neighbours corresponding to all the labels.[1] Then in the second step, only these samples are used for label prediction. In comparison to the conventional kNN algorithm, note that we additionally introduce an initial pruning step where we pick visually similar neighbours that cover all the labels. This also relates with the idea of "bottom-up pruning" common in day-to-day scenarios such as buying a car, or selecting a cloth to wear, where first the potential candidates are short-listed based on a preliminary analysis, and then another set of criteria is used for final selection.

It is well-known that the performance of kNN based methods largely depends on how two images are compared (Guillaumin et al. 2009; Makadia et al. 2008, 2010). Usually, this comparison is done using a set of features extracted from images and a specific distance metric for each feature (such as L_1 distance for colour histograms, or L_2 for GIST descriptor). As the 2PKNN algorithm works in the nearest neighbour setting, we would like to learn a distance metric that maximizes the annotation performance. With this goal, we perform metric learning over 2PKNN by extending the popular Large Margin Nearest Neighbour (LMNN) metric learning algorithm proposed by Weinberger and Saul (2009) for multi-label prediction. Since it requires to perform pairwise comparisons iteratively, scalability becomes one of the important concerns while working with thousands of images. To address this, we implement metric learning by alternating between stochastic sub-gradient descent and projection steps on subsets of training pairs, that has a motivation similar to the Pegasos algorithm (Shalev-Shwartz et al. 2007). This allows to optimize the weights iteratively using a small number of comparisons at each iteration, thus making our metric learning formulation scalable.

We evaluate and compare the proposed approach with existing methods on four image annotation datasets: Corel-5K (Duygulu et al. 2002), ESP-Game (von Ahn and Dabbish 2004), IAPR-TC12 (Grubinger 2007) and MIRFlickr-25K (Huiskes and Lew 2008). Our first set of results is based on the features provided by Guillaumin et al. (2009)[2] (we will refer to these features as "TagProp-features"). These features have become a de facto standard for comparing annotation performance, and are used by almost all the recent approaches. Next we extend this feature set by including deep learning based features extracted using a state-of-the-art pre-trained Convolutional Neural Network (CNN) model of Donahue et al. (2014) (we will refer to these features as "CNN-features"). We also compute features using two modern encoding techniques: Fisher vector (Perronnin et al. 2010) and VLAD (Jégou et al. 2010) (we will refer to these features as "Encoding-features"). Finally, we embed (different combinations of) these features into a common subspace learned using canonical correlation analysis (CCA) (Hotelling 1936), and kernelized canonical correlation analysis (KCCA). This is motivated by the well-known problem of *semantic gap*, because of which it is difficult to build meaningful associations between low-level visual features and high-level semantic concepts. Using cross-modal embeddings learned through (K)CCA, we try to address this partially by learning representations that maximize the correlation between visual and textual content in a common subspace.

Contributions: This paper is an extension of the conference version (Verma and Jawahar 2012). To our knowledge, this is the first published work that proposed to explicitly integrate label information while determining the neighbours of an image in the image annotation task. Here we extend this work in the following ways:

[1] We shall use the terms class/label interchangeably.

[2] These features are available at http://lear.inrialpes.fr/people/guillaumin/data.php.

1. We include an analytical and empirical discussion on the diversity and completeness of labels in the neighbours obtained after the first pass of 2PKNN, and also compare these with the conventional kNN algorithm.
2. In addition to the TagProp-features, we introduce and extensively evaluate our approach using the new features and feature embeddings as discussed above on all the datasets.
3. We include several additional studies that provide meaningful insights about the annotation problem, and our approach.
4. We additionally evaluate and compare using the modern MIRFlickr-25K dataset, that has a larger test set compared to the other three datasets.
5. For fair comparisons, we also extensively evaluate two state-of-the-art nearest neighbour based methods JEC (Makadia et al. 2008, 2010) and TagProp (Guillaumin et al. 2009) under similar set-up throughout.

Experiments demonstrate that compared to using only the TagProp-features, the new features/feature-combination(s) along with the cross-modal embedding learned using KCCA significantly improve the performance of all the compared methods. Moreover, the proposed approach achieves state-of-the-art results on two datasets (in terms of F1 score (Sect. 6.1)), and comparable on the other two. Specially on the well-known and challenging Corel-5K dataset, we now achieve an F1 score of 49.9 %, which is (absolute) 6.4 % better than the second best method TagProp (Guillaumin et al. 2009) that achieves an F1 score of 43.5 %.

The paper is organized as follows. In Sect. 2, we review some of the notable and recent works in this domain. Sections 3 and 4 describe the 2PKNN method and the metric learning formulation respectively. In Sect. 5, we discuss the datasets and features used in our experiments. In Sect. 6, we present the experimental analyses, and finally conclude in Sect. 7.

2 Related Works

The goal of an image annotation model is to formulate a mapping between the images and annotation labels. This was initially addressed using translation models such as Mori et al. (1999) and Duygulu et al. (2002). These treat it as a problem of machine translation, where an image region needs to be translated into a semantic label. In the relevance models such as CMRM (Jeon et al. 2003), CRM (Lavrenko et al. 2003) and MBRM (Feng et al. 2004), image annotation was modelled as a problem of computing the joint probability of image regions and labels. In MBRM (Feng et al. 2004), it was shown that using regular blocks rather than arbitrary shaped regions as in CRM/CMRM, and modelling the absolute presence/absence of labels using a a Bernoulli distribution rather than modelling their frequency using a Multinomial distribution could provide better performance. Xiang et al. (2009) proposed a Markov Random Field based approach that could flexibly accommodate most of the previous generative models. A few discriminative models such as those proposed by Carneiro et al. (2007), Fu et al. (2012), Verma and Jawahar (2013) treat each label as a class of a multi-class multi-labelling problem, and learn separate class-specific models.

The image annotation domain has primarily been dominated by nearest neighbour based approaches, that predict labels of a test image by computing its similarity with a (sub)set of training images. The Joint Equal Contribution (JEC) approach proposed by Makadia et al. (2008, 2010) treats the problem of image annotation as that of image retrieval. It demonstrated that a simple nearest-neighbour based greedy algorithm could outperform earlier, relatively complex models, though by using multiple high-dimensional global features rather than simple region-based features. Although JEC is conceptually simple, it achieved the best results on benchmark annotation datasets when it was proposed. Inspired from the success of JEC, a weighted kNN based method called TagProp was proposed by Guillaumin et al. (2009). This transfers labels to a test image by taking a weighted average of keywords' presence among the neighbouring (training) images. To address the class-imbalance problem, logistic discriminant (sigmoid) models are wrapped over the weighted kNN method. This boosts the importance given to infrequent/rare labels and suppresses it for frequent labels. They also proposed a metric learning approach for learning weights that combines multiple distances computed using different features. As part of this work, the authors released a set of pre-computed features for the standard annotation datasets. Since then, these features have been used by almost all the annotation approaches. Li et al. (2009) proposed a measure to compute the relevance of a tag to an image by taking into account its frequency in the neighbouring samples of that image, and the entire (training) collection. Another nearest-neighbour based method (Zhang et al. 2010) tries to benefit from feature sparsity and clustering properties using a regularization based algorithm for feature selection.

Among the recent methods, Moran and Lavrenko (2014) proposed a greedy approach to identify the best kernel for each feature while computing image similarity using a set of features. This in turn results into a sparse subset of features that maximize annotation performance (in terms of F1 score). Kalayeh et al. (2014) proposed a formulation based on weighted multi-view non-negative matrix factorization with the goal of learning a generative model specific to each query/test image using its neighbouring samples. Murthy et al. (2014) combined a discrete variant of the MBRM model (Feng et al. 2004) with binary one-vs.-rest

SVM models to formulate a hybrid approach for annotating images. This combination particularly helped in increasing the number of labels that were correctly recalled. Ballan et al. (2014) showed that learning (kernelized) cross-modal feature embedding can significantly improve the performance of nearest neighbour based methods such as those of Makadia et al. (2008, 2010), Li et al. (2009), Guillaumin et al. (2009), Verma and Jawahar (2012).

In parallel to the above advances, there have been several works such as Jin et al. (2009), Wang et al. (2011) that target the problem of multi-label classification. However, usually such methods are shown to work on small vocabularies containing a few tens of labels. Our work falls under the category of supervised image annotation methods (Feng et al. 2004; Makadia et al. 2008, 2010; Guillaumin et al. 2009; Zhang et al. 2010; Carneiro et al. 2007; Xiang et al. 2009; Nakayama 2011; Fu et al. 2012; Chen et al. 2013; Moran and Lavrenko 2014; Murthy et al. 2014) that address a more realistic and challenging scenario where the vocabulary contains few hundreds of labels and the datasets seriously suffer from class-imbalance and incomplete-labelling.

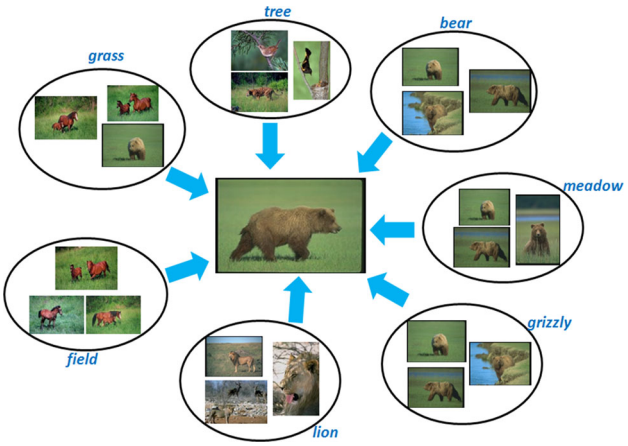

Fig. 1 In the first pass of 2PKNN, for a given image to be annotated (*center*), we identify its semantic neighbours corresponding to each semantic group, and only these samples are considered during label prediction. Since each image can have multiple labels, an image can come from more than one semantic group. E.g., the semantic groups corresponding to the labels "bear", "meadow" and "grizzly" have two images in common. It is worth noticing that unlike the usual kNN based image annotation methods that make use of only feature-based similarity while determining the neighbours of a given image and ignore the label information, here we explicitly make use of both

3 Label Prediction Model

Now we present our 2PKNN method for image annotation. Let $\{I_1, \ldots, I_t\}$ be a collection of images and $\mathcal{Y} = \{y_1, \ldots, y_l\}$ be a vocabulary of l labels (or semantic concepts). The training set $\mathcal{T} = \{(I_1, Y_1), \ldots, (I_t, Y_t)\}$ consists of pairs of images and their corresponding label sets, with each $Y_i \subseteq \mathcal{Y}$. Similar to the Supervised Multiclass Labeling (or SML) method of Carneiro et al. (2007), we assume the conditional probabilities $P(A|y_i)$ that model the feature distribution of an image A given a semantic concept $y_i \in \mathcal{Y}$. Using this, we model image annotation as a problem of finding the posterior probability for each label:

$$P(y_i|A) = \frac{P(A|y_i)P(y_i)}{P(A)} \tag{1}$$

where $P(y_i)$ is the prior probability of the label y_i. Then, given an unannotated image J, the best label for it will be given by

$$y^* = \arg\max_i P(y_i|J) \tag{2}$$

Let $\mathcal{T}_i \subseteq \mathcal{T}$, $\forall i \in \{1, \ldots, l\}$ be the subset of training data that contains *all* the images annotated with the label y_i. Since each set $\mathcal{T}_i$ contains images with one semantic concept common among them, we call it a *semantic group*. It should be noted that the sets $\mathcal{T}_i$ are not disjoint, as an image usually has multiple labels and hence belongs to multiple semantic groups. Given an unannotated image J, from each semantic group we pick K_1 images that are most similar to J and form corresponding sets $\mathcal{T}_{J,i} \subseteq \mathcal{T}_i$. Thus, each $\mathcal{T}_{J,i}$ contains those images that are *most informative* in predicting the probability of the label y_i for J [as discussed in Sect. 1, our approach is motivated by the observation that "similar images share similar labels" (Makadia et al. 2008, 2010)]. The samples in each set $\mathcal{T}_{J,i}$ are the semantic neighbours of J corresponding to y_i, and help in incorporating image-to-label similarity. Once $\mathcal{T}_{J,i}$s are determined, we merge them all to form a set $\mathcal{T}_J = \{\mathcal{T}_{J,1} \bigcup \ldots \bigcup \mathcal{T}_{J,l}\}$. This way, we obtain a subset of the training data $\mathcal{T}_J \subseteq \mathcal{T}$ specific to J that contains its semantic neighbours corresponding to all the labels in the vocabulary $\mathcal{Y}$. This is the *first pass* of 2PKNN, as illustrated in Fig. 1.

In $\mathcal{T}_J$, each label would appear (at least) K_1 times, which in turn tries to address the class-imbalance issue. To understand how this step also tries to handle incomplete-labelling, we analyse the cause of this. Incomplete-labelling occurs because some (either too obvious, or highly context-specific) labels are often missed by human annotators while manually annotating a dataset, and hence many images depicting such concepts are actually not annotated with them. Under this situation, given an unseen image, if we use only its *few* nearest neighbours from the *entire* training data [as in Makadia et al. (2008, 2010), Guillaumin et al. (2009)], then such labels may not appear among these neighbours and hence would not get appropriate scores. In contrary, the first pass of 2PKNN builds a neighbourhood where all the labels are present explicitly. Therefore, now even those labels that did not appear among

Fig. 2 For a test image from the IAPR-TC12 dataset, the *top row on the right* shows its 4 nearest images (and their ground-truth labels) from the training data determined after the first pass of 2PKNN and the *bottom row* shows its 4 nearest images determined using JEC (Makadia et al. 2008, 2010). The labels in *bold* are the ones that match with the ground-truth labels of the test image. Note the frequency (9 vs. 6) and diversity ({sky, house, landscape, bay, road, meadow} versus {sky, house}) of matching labels for 2PKNN versus JEC

the neighbours determined using the usual kNN have better prediction chances.

The *second pass* of 2PKNN is a weighted sum over the samples in $\mathcal{T}_J$ to assign importance to labels based on image similarity. This gives the posterior probability for J given a label $y_k \in \mathcal{Y}$ as

$$P(J|y_k) \propto \sum_{(I_i,Y_i)\in\mathcal{T}_J} \theta_{J,I_i}.P(y_k|I_i) \tag{3}$$

where, $\theta_{J,I_i} = exp(-\pi D(J,I_i))$ denotes the contribution of image I_i in predicting the label y_k for J depending on their visual similarity, with π being a scalar that controls the decay of θ_{J,I_i}; $D(J,I_i)$ denotes the distance between J and I_i in feature space (see Eq. 7 for the definition of $D(J,I_i)$); and $P(y_k|I_i) = \delta(y_k \in Y_i)$ denotes the presence/absence of label y_k in the label set Y_i of I_i, with $\delta(\cdot)$ being 1 when the argument holds true and 0 otherwise. Assuming that the first pass of 2PKNN gives a subset of the training data where each label has comparable frequency, we set the prior probability in Eq. 1 as a uniform distribution; i.e., $P(y_i) \propto \frac{1}{|\mathcal{T}|}, \forall i \in \{1,\ldots,l\}$. Putting Eq. 3 in Eq. 1 provides a ranking of all the labels based on their probability of getting assigned to the unseen image J. Finally, the probability score $P(y_i|A)$ is regularized using the following normalization [similar to Moran and Lavrenko (2014)]:

$$P(y_i|A) = \frac{P(y_i|A)}{\max_{A'} P(y_i|A')} \tag{4}$$

Note that along with image-to-image similarities, the second pass of 2PKNN implicitly takes care of label-to-label dependencies since the labels appearing together in the same neighbouring image will get equal importance [analogous to Guillaumin et al. (2009)].

Figure 2 shows an example from the IAPR-TC12 dataset illustrating how the first pass of 2PKNN tries to address both class-imbalance and incomplete-labelling. For a given test image (first column) along with its ground-truth labels, we can notice the presence of rare labels {"landscape", "bay", "road","meadow"} among its four nearest images found after the first pass of 2PKNN (top row on the right), without compromising with frequent labels {"sky", "house"}. In contrary, the neighbours obtained using JEC (Makadia et al. 2008, 2010) (second row) contain only frequent labels. We can also observe that though the labels {"landscape", "meadow"} look obvious for the neighbours found using JEC, these are actually absent in their ground-truth annotations (incomplete-labelling), whereas the first pass of 2PKNN explicits their presence among the neighbours selected for label prediction. To discuss these aspects more formally, below we provide an analysis on the diversity and completeness of labels included in the neighbours identified using 2PKNN and classical kNN. We will provide an empirical analysis on this in Sect. 6.5.2.

3.1 Analysing Diversity and Completeness

Here we try to analyse two aspects related to the presence of labels in the neighbours selected for label prediction: *diversity* and *completeness*. In the present context, we define "diversity" as the number of distinct labels that are present in the selected neighbours, and "completeness" as the state when all the labels are present in the selected neighbours. In order to achieve completeness, we will require perfect

diversity, i.e., presence of all the labels in the neighbours. Along with 2PKNN, we will analyse and compare these aspects with respect to the conventional kNN algorithm.

Recall that in the first pass of 2PKNN, we identify K_1 nearest neighbours of a given (test) sample J from each semantic group and take their union, thus obtaining a subset of training samples $\mathcal{T}_J \subseteq \mathcal{T}$ specific to J. Let K_2 denote the number of nearest neighbours of J from $\mathcal{T}_J$ that are considered for label prediction. Also, with respect to kNN algorithm, we assume to pick K nearest neighbours of J from the complete training set for label prediction. We also assume to have sufficiently large number of samples for each label.

For simplicity, let us initially consider a vocabulary $\mathcal{Y} = \{y_1, y_2, y_3\}$ of three labels (i.e., a vocabulary of size $a = 3$), and assume that each sample in the (training) data is associated with exactly $b = 2$ distinct labels. For 2PKNN, if we have $K_1 = 1$, then we will get two samples in the set $\mathcal{T}_J$ (since one sample will be selected twice for two labels). This means that in $\mathcal{T}_J$, the frequency of two labels will be one, and that of the remaining one label will be two. In this case, if we consider $K_2 = 1$, then the diversity of labels in the selected neighbours will be 2, and if we consider $K_2 = 2$, then we will achieve perfect diversity and completeness of labels in the selected neighbours. Now, let us consider the case of kNN. If we consider $K = 1$, then the diversity of labels will be 2 similar to 2PKNN, and in the best scenario we will require $K = 2$ samples to achieve perfect diversity. However, since we do not take into account the label information of samples in kNN while selecting the neighbours and make use of only sample features, we may end-up getting neighbouring samples labelled with the same two labels even for very large values of K, and thus we can not guarantee the minimum value of K to achieve perfect diversity.

In general, for any (positive integral) values of a and K_1, if we assume each sample x_u to be labelled with $b_u \leq a$ distinct labels (b_u may be different for different samples in multi-label scenario), then in order to achieve perfect diversity using 2PKNN, we will require K_2 to be $\lfloor \frac{a-1}{\max(b_u)} \rfloor + 1$ in the best scenario (lower bound), and $(a - \min(b_u)) \times K_1 + 1$ in the worst scenario (upper bound). These will depend on the overlap of labels in the selected neighbours. In case of kNN algorithm, we will require K to be $\lceil \frac{a-1}{\max(b_u)} \rceil + 1$ to achieve perfect diversity in the best scenario, which is the same as that for 2PKNN. However, here we cannot bound the value of K to achieve perfect diversity in the worst scenario, following the same reasoning as above.

Note that in practice, since we consider all the samples in $\mathcal{T}_J$ during label prediction (i.e., $K_2 = |\mathcal{T}_J|$ in Eq. 3), we can assure perfect diversity and completeness of labels using 2PKNN.

4 Metric Learning

Most of the existing classification based metric learning algorithms try to increase inter-class and reduce intra-class distances, thus treating each pair of samples in a binary manner. Since image annotation is a multi-label prediction task, here the similarity between two samples need not be binary, and hence classification based metric learning cannot be applied directly. As part of metric learning, our aim is to learn a distance metric that maximizes the annotation performance for 2PKNN. For this purpose, we extend the LMNN algorithm (Weinberger and Saul 2009) for multi-label prediction. Below, first we provide an overview of the LMNN algorithm, and then present our formulation.

4.1 Large Margin Nearest Neighbour (LMNN)

The goal of LMNN is to learn a Mahalanobis metric such that the performance of kNN classification is improved. Let us assume a training set $\{\mathbf{x}_i, y_i\}$, where $\mathbf{x}_i \in \mathbb{R}^d$ denotes a sample and $y_i \in \{1, \ldots, C\}$ denotes its category/class from total C classes. Given two samples $\mathbf{x}_i$ and $\mathbf{x}_j$, the Mahalanobis distance between them is given by:

$$d_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) = ((\mathbf{x}_i - \mathbf{x}_j)^T \mathbf{M} (\mathbf{x}_i - \mathbf{x}_j))^{\frac{1}{2}}$$

where $\mathbf{M}$ is a symmetric positive semidefinite matrix ($\mathbf{M} \succeq 0$). In LMNN, $\mathbf{M}$ is learned such that the local neighbourhood of a sample belongs to the same category. To do this, for a given sample, its neighbours from the same class are pulled closer and those from different classes are pushed farther.

For a given sample $\mathbf{x}_i$, its *target* neighbours are defined as its k nearest samples from the same class, and *impostors* as its neighbours from other classes that are closer than the target neighbours. Using this information, a Mahalanobis metric is learned such that each sample is closer to its target neighbours than impostors by a margin. Let $\mathbf{x}_j$ and $\mathbf{x}_k$ be a target neighbour and an impostor respectively for $\mathbf{x}_i$, then this constraint can be expressed as:

$$d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 \tag{5}$$

Note that the above constraint is enforced only on local neighbours. Based on the above constraints, the objective function of LMNN is given by:

$$\begin{aligned} \min_{\mathbf{M}} \sum_{ij} \eta_{ij} d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) + \mu \sum_{ijk} \eta_{ij}(1 - \lambda_{ik})\xi_{ijk} \\ s.t. : d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_k) - d^2_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j) \geq 1 - \xi_{ijk} \\ \xi_{ijk} \geq 0 \\ \mathbf{M} \succeq 0 \end{aligned} \tag{6}$$

Here, the first term tries to pull target neighbours closer, and the second term penalizes violations of the constraints in Eq. 5. The variable η_{ij} is 1 if $\mathbf{x}_j$ is a target neighbour of $\mathbf{x}_i$ and 0 otherwise; λ_{ik} is 1 if $\mathbf{x}_i$ and $\mathbf{x}_k$ belong to the same class and 0 otherwise; and $\mu > 0$ manages tradeoff between the two terms.

4.2 Metric Learning for 2PKNN

Let there be two images A and B, each represented by n features $\{\mathbf{f_A^1}, \ldots, \mathbf{f_A^n}\}$ and $\{\mathbf{f_B^1}, \ldots, \mathbf{f_B^n}\}$ respectively. The distance between two images is computed by finding the distance between their corresponding features using some specialized distance measure for each feature (such as L_1 for colour histograms, χ^2 for bag-of-words histograms, etc.), and then combining them all. Let d^i_{AB} denote the distance between A and B computed using the ith feature. In order to optimally combine multiple feature distances, we use a linear distance metric $\mathbf{w} \in \mathcal{R}^n_+$ in the distance space. Based on this, we write the distance between A and B as:

$$D(A, B) = \sum_{i=1}^{n} \mathbf{w}(i)\, d^i_{AB} \tag{7}$$

Now we describe how to learn the metric $\mathbf{w}$ for multi-label image annotation task. For a given labelled sample $(I_p, Y_p) \in \mathcal{T}$, we define its (i) *target neighbours* as its K_1 nearest images from the semantic group $\mathcal{T}_q$, $\forall q$ such that $y_q \in Y_p$, and (ii) *impostors* as its K_1 nearest images from $\mathcal{T}_r$, $\forall r$ such that $y_r \in \mathcal{Y} \setminus Y_p$. Our objective is to learn the metric such that the distance of a sample from its target neighbours is minimized, and is also less than its distance from any of the impostors (i.e., *pull* the target neighbours and *push* the impostors). In other words, given an image I_p along with its labels Y_p, we want to learn the weights such that its nearest (K_1) semantic neighbours from the semantic groups $\mathcal{T}_q$'s (i.e., the groups corresponding to its ground-truth labels) are pulled closer, and those from the remaining semantic groups are pushed farther (Fig. 3). With this goal, for a sample image I_p, its target neighbour I_q and its impostor I_r, the loss function will be given by

$$E_1 = \sum_{pq} \eta_{pq} D(I_p, I_q) + \mu \sum_{pqr} \eta_{pq}(1 - \lambda_{pr})[1 + D(I_p, I_q) - D(I_p, I_r)]_+ \tag{8}$$

where $\mu > 0$ handles the trade-off between the two error terms. The variable η_{pq} is 1 if I_q is a target neighbour of I_p and 0 otherwise. $\lambda_{pr} = \frac{|Y_p \cap Y_r|}{|Y_r|} \in [0, 1]$, with Y_r being the label set of an impostor I_r of I_p, and $[z]_+ = max(0, z)$ is the hinge loss which will be positive only when $D(I_p, I_r) < D(I_p, I_q) + 1$ (i.e., when for a sample I_p, its impostor I_r is nearer than its target neighbour I_q). To make sure that a target neighbour I_q is much closer than an impostor I_r, a margin (of size 1) is used in the error function.

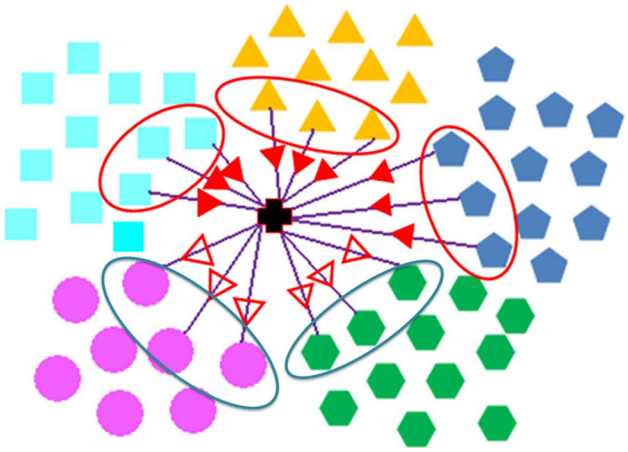

Fig. 3 Illustration of distance metric learning for 2PKNN. Let there be 5 labels $\mathcal{Y} = \{A, B, C, D, E\}$ denoted as A: *squares*, B: *triangles*, C: *pentagons*, D: *hexagons*, and E: *circles*. Each set $\mathcal{T}_\alpha, \alpha \in \mathcal{Y}$ consists of samples that have one label as α. For a given sample (denoted by *cross*), let its actual labels be $\{A, B, C\}$. During distance metric learning with $K = 3$, its three nearest neighbours from $\mathcal{T}_A$, $\mathcal{T}_B$ and $\mathcal{T}_C$ act as target neighbours that need to be pulled closer to it, while those from the remaining ones act as impostors that need to be pushed far from it

The above loss function is minimized by the following constrained optimization problem:

$$\begin{aligned} \min_{\mathbf{w}} \; & \textstyle\sum_{pq} \eta_{pq} D(I_p, I_q) + \mu \sum_{pqr} \eta_{pq}(1 - \lambda_{pr}) \xi_{pqr} \\ s.t. \quad & D(I_p, I_r) - D(I_p, I_q) \geq 1 - \xi_{pqr} \quad \forall p, q, r \\ & \xi_{pqr} \geq 0 \quad \forall p, q, r \\ & \mathbf{w}(i) \geq 0 \quad \forall i; \; \textstyle\sum_{i=1}^{n} \mathbf{w}(i) = n \end{aligned} \tag{9}$$

Here, the slack variables ξ_{pqr} represent the hinge loss in Eq. 8. By applying L_1 regularization on $\mathbf{w}$, we try to impose sparsity on the learned weights.

For large datasets on the order of tens of thousands of samples, the above optimization problem can have several millions of constraints. This makes the scalability difficult using conventional gradient descent. To overcome this, we implement it by alternatively using stochastic sub-gradient descent and projection steps (similar to Pegasos (Shalev-Shwartz et al. 2007)) on subsets of training data. This gives an approximate solution using a small number of comparisons, thus making our approach scalable to large datasets containing thousands of samples.[3]

[3] An implementation of 2PKNN and metric learning is available at http://researchweb.iiit.ac.in/~yashaswi.verma/eccv12/2pknn.zip.

Table 1 General (columns 2–5) and some insightful (columns 6–8) statistics of the four datasets considered in this work

Dataset	Images	Training	Testing	Labels	Labels/image	Images/label	Labels#
Corel-5K	4999	4500	499	260	3.4, 4, 5	58.6, 22, 1004	195 (75.0 %)
ESP-Game	20770	18689	2081	268	4.7, 5, 15	326.7, 172, 4553	201 (75.0 %)
IAPR-TC12	19627	17665	1962	291	5.7, 5, 23	347.7, 153, 4999	217 (74.6 %)
MIRFlickr-25K	25000	12500	12500	38	4.7, 5, 17	1560.7, 995.5, 5216	22 (57.9 %)

In columns 6 and 7, the entries are in the format "mean, median, maximum". Column 8 ("Labels#") shows the number of labels whose frequency is less than the mean label frequency

4.3 Comparison with LMNN

Similar to LMNN, the proposed metric learning formulation works on pairs and triplets of samples that are defined in terms of target neighbours and impostors. Recall that while LMNN is meant for single-label data, ours is of multi-label data. Below we discuss the differences between the two, that are primarily because of the differences in the tasks that each addresses:

- The primary difference between the two formulations lies in terms of the definition of target neighbours and impostors. In case of multi-label data, samples from multiple classes behave as target neighbours and impostors. Moreover, our definitions of such samples especially suit the 2PKNN algorithm, since these are defined based on the semantic neighbours of a sample.
- In LMNN, the variable λ is binary, whereas we define it to be in a continuous range [0, 1], thus scaling the hinge loss depending on the overlap between the label sets of a given image I_p and its impostor I_r. This means that for a given sample, the amount of push applied on its impostor varies depending on conceptual similarity with that sample. An impostor with large similarity will be pushed less, whereas one with small similarity will be pushed more. This makes our formulation suitable for multi-label tasks such as image annotation.
- While working with high-dimensional features, it becomes practically infeasible to learn a square Mahalanobis distance metric ($\mathbf{M}$) as done in LMNN. To overcome this, we learn a linear metric $\mathbf{w}$ in distance space rather than feature space. Since a sample is usually represented using a few tens of features, this makes the dimensionality of $\mathbf{w}$ practically feasible to deal with.

5 Datasets and Features

5.1 Datasets

We consider four image annotation datasets in our experiments:

- **Corel-5K:** This was introduced by Duygulu et al. (2002), and since then it has become a de facto evaluation benchmark for comparing the annotation performance.
- **ESP-Game:** This was published by von Ahn and Dabbish (2004). It contains images annotated using an on-line game, where two (mutually unknown) players are randomly given an image, and they need to predict the same keyword(s) in order to score points. This way, several people participate in the manual annotation task, thus making this dataset very challenging and diverse.
- **IAPR-TC12:** This was introduced by Grubinger (2007) for cross-lingual information retrieval. In this, each image is associated with a detailed description. Makadia et al. (2008, 2010) extracted nouns from these descriptions and treated them as annotations. Since then, it has been widely used for evaluating image annotation methods.
- **MIRFlickr-25K:** This dataset contains images downloaded from Flickr, and was introduced for evaluating keyword-based image retrieval (Huiskes and Lew 2008). Verbeek et al. (2010) got the images in this dataset manually annotated with 24 concepts for evaluating automatic annotation performance. In the first round of annotation, for each image, the annotators were asked whether it was at least partially relevant for each concept. In the second round, a stricter notion of relevance was used for 14 concepts. For each concept, the images that were annotated as relevant in the first round were considered, and marked as relevant only if that concept was depicted in a significant portion of the image. In this way, each image in this dataset is annotated by its relevance for total 38 labels. Compared to the other three datasets, this dataset has a larger test set, though the number of distinct labels is relatively quite small.

In Table 1, columns $2-5$ show some general statistics of the four datasets; and in columns $6-8$, we highlight some other statistics that provide better insights about the properties of these datasets. It can be noticed that for the first three datasets, around 75 % of the labels have frequency less than the mean label frequency (column 8), and also the median label frequency is far less than the corresponding mean frequency (column 7). Figure 4 shows the frequencies of the labels in these datasets sorted in descending order. It

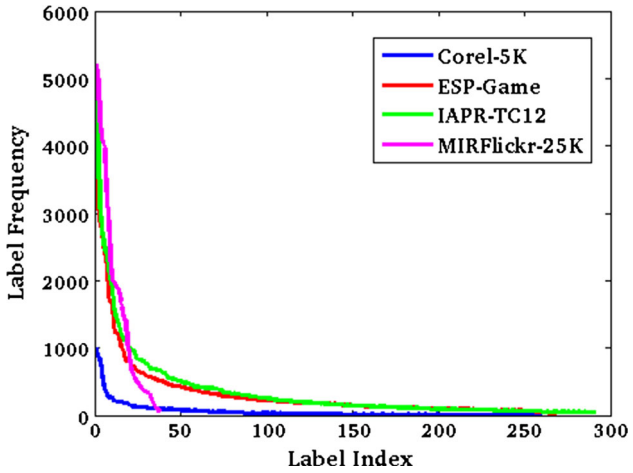

Fig. 4 Frequency of labels (or, number of images per label) in the training set of each of the four datasets sorted in decreasing order order (Color figure online)

can be observed that *tail* of the MIRFlickr-25K dataset is much higher than that of the other three datasets. Each of those three datasets contains only a small number of high frequency labels, and a huge number of labels with very low frequencies. This suggests that compared to the MIRFlickr-25K dataset, the other three datasets better model the aspect of class-imbalance. E.g., the most frequent label in the ESP-Game dataset is "man" that has 4, 553 occurrences. Whereas, the second most frequent label "white" is significantly less popular, with 3, 175 occurrences (a drop of around 30 %). The graphs for the three datasets drop rapidly, and almost level out near the tail [the *long tail* phenomenon (Anderson 2006)]. Moreover, for the Corel-5K, ESP-Game and IAPR-TC12 datasets, the cumulative frequency of the 25 most frequent labels is 52.3, 45.2 and 45.0 % respectively, whereas that of the 25 least frequent labels is just 0.6, 1.5, and 1.2 % respectively. This indicates that the labels appearing towards the tail (such as "bikini", "icon" and "mail" in the ESP-Game dataset) might be *individually unimportant* in the sense that there are very few images depicting each of these concepts. However, since there are lots of such labels, they become *collectively significant*, and prediction accuracies on these labels have a critical impact on the overall performance.

Though it is not straightforward to quantify incomplete-labelling, we try to analyse it from the number of labels per image (column 6). We posit that a large gap between mean (or median) and maximum number of labels per image indicates that many images are not labelled with all the relevant labels. Based on this, we can infer that both ESP-Game and IAPR-TC12 datasets suffer from incomplete-labelling. For the Corel-5K dataset, we examined the images and their corresponding annotations to realize incomplete-labelling. Unlike these three datasets, since the vocabulary size in the MIRFlickr-25K dataset is very small, and it was annotated under a strict set-up, the chances of incomplete-labelling are rare.

5.2 Features

Our first set of features is the TagProp-features released by Guillaumin et al. (2009). These are a combination of local and global features. The local features include the SIFT (Lowe 2004) and robust Hue (van de Weijer and Schmid 2006) descriptors obtained densely from multi-scale grid, and from Harris-Laplacian interest points. Each of these descriptors is used to form a histogram of bag-of-words representation. The global features comprise of the GIST descriptor (Oliva and Torralba 2001), and 3-D histograms (with 16-bins per channel) in each of the RGB, HSV and LAB colour spaces. To encode some information about the spatial-layout of an image, all but the GIST descriptor are also computed over three equal horizontal partitions for an image (denoted using "(V3H1)" as a suffix). In this case, the bin-size for colour histograms is reduced to 12-bins per channel to limit histogram sizes.

In addition to the above features, we extract deep learning based CNN-features using the pre-trained CNN model of Donahue et al. (2014). It is a generic CNN model, and has been found to work well in a variety of visual recognition tasks. In practice, we consider the output of the last three layers of the network as the features. We also compute representations based on two modern encoding techniques: Fisher vector (Perronnin et al. 2010) and VLAD (Jégou et al. 2010). For both, we consider the 128-dimensional SIFT features for learning a vocabulary of 256 clusters.

Table 2 shows the dimensionality of all the features, and the corresponding distance metrics used for computing pair-wise distances. For TagProp-features, we use the same distance metrics as in Guillaumin et al. (2009). For CNN-features, our choice of distance metrics was based on the annotation performance using JEC (Makadia et al. 2008, 2010) (since JEC does not involve any learning as such). For Encoding-features, we used L_2 distance following Perronnin et al. (2010). Also, in our preliminary evaluations using JEC, we empirically observed that before computing the pair-wise distances, taking a square-root of each element of some of the feature vectors[4] provided additional boost in the annotation performance. This has a motivation analogous to "power normalization" described in Perronnin et al. (2010).

5.3 Feature Embedding

In a recent work by Ballan et al. (2014), it was shown that learning cross-modal embedding of visual features can provide significant improvements in the performance of nearest

[4] Features indexed by (3), (4), and (9) to (15) in Table 2.

Table 2 Different features, their dimensionalities, and distance metrics used for computing pair-wise distances

Feature name	Dim.	Dist.
(1) Dense SIFT, **(2)** Harris SIFT	1000	χ^2
(3) Dense SIFT (V3H1), **(4)** Harris SIFT (V3H1)	3000	χ^2
(5) GIST	512	L_2
(6) Dense Hue, **(7)** Harris Hue	100	χ^2
(8) Dense Hue (V3H1), **(9)** Harris Hue (V3H1)	300	χ^2
(10) RGB, **(11)** HSV, **(12)** LAB	4096	L_1
(13) RGB (V3H1), **(14)** HSV (V3H1), **(15)** LAB (V3H1)	5184	L_1
(16) Layer-5	9216	L_1
(17) Layer-6, **(18)** Layer-7	4096	L_2
(19) Fisher	65536	L_2
(20) VLAD	32768	L_2

Top: the fifteen publicly available features provided by Guillaumin et al. (2009) ("TagProp-features")
Middle: features computed using the pre-trained CNN model of Donahue et al. (2014) ("CNN-features")
Bottom: features computed using advanced encoding techniques ("Encoding-features")

neighbour based annotation methods. As discussed in Sect. 1, the motivation behind cross-modal embedding is to reduce the semantic gap. Inspired from this, we learn a common subspace for both image features and labels. For this, we investigate both CCA and KCCA.

Let I_i and I_j be two images, both of which are represented using a set of feature vectors h_i^f and h_j^f for all features $f \in \mathcal{F}$. In case of CCA, we L_2-normalize each feature and use a linear kernel to compute similarity between two images:

$$K_v^{cca}(I_i, I_j) = \frac{1}{|\mathcal{F}|} \sum_{f \in \mathcal{F}} \langle h_i^f, h_j^f \rangle \tag{10}$$

And for KCCA, we use an exponential kernel:

$$K_v^{kcca}(I_i, I_j) = exp\left(\frac{-D(I_i, I_j)}{Z}\right) \tag{11}$$

where $D(I_i, I_j)$ is the distance between the two images computed using Eq. 7, and Z is the mean distance among all the training pairs.

For textual features, we represent the labels $Y \subseteq \mathcal{Y}$ that are associated with an image I using a binary vector $g \in \mathbb{R}^l$ (l is the vocabulary size). We keep $g(k) = 1$ if $y_k \in Y$, and 0 otherwise. With this, we use a linear kernel to compute similarity between two textual features, which is same as counting the number of common labels between two images:

$$K_t(g_i, g_j) = \langle g_i, g_j \rangle = \sum_{k=1}^{l} g_i(k) g_j(k) \tag{12}$$

We use the implementation of Hardoon et al. (2004) for learning the common subspace. It penalizes the norm of the projection vectors, and regularizes the learning to avoid trivial solutions. Similar to Ballan et al. (2014), we set the precision parameter for Gram-Schmidt decomposition to be $\eta = 30$, and the regularization parameter to be $\kappa = 0.1$. Also, the visual and textual spaces are swapped before computing the projection vectors. In practice, we consider the full-length feature vectors in the common subspace. After embedding the samples, we compute distance between two feature vectors using the L_2 distance metric, that was empirically chosen based on the annotation performance of the JEC method (Makadia et al. 2008, 2010).

Intuitively, the image representation in the learned subspace better captures the semantics of the data, and the neighbouring samples are semantically more meaningful than those obtained using raw features. As validated in our experiments, this in turn significantly improves the performance of the nearest neighbour based annotation methods such as Makadia et al. (2008, 2010), Guillaumin et al. (2009), including ours. Another practically useful advantage of feature embedding is that it provides a very compact yet effective representation for images. E.g., if we concatenate all the features as mentioned in Table 2, an image would be represented by a 152864-dimensional feature vector. Whereas, after feature embedding, this reduces to just a few hundreds of dimensions (or less).

6 Experiments

Here we empirically analyse and compare the performance of our approach on the above datasets.

6.1 Evaluation Measures

To analyse the annotation performance, we compute precision and recall of each label in a dataset. Suppose a label y_i is present in the ground-truth of m_1 images, and it is predicted for m_2 images during testing, out of which m_3 predictions are correct ($m_3 \leq m_2$ and $m_3 \leq m_1$). Then its precision will be $= m_3/m_2$, and recall will be $= m_3/m_1$. We average these

values over all the labels in a dataset and get (percentage) mean precision P and mean recall R. Using these two scores, we compute F1 score, which is the harmonic mean of P and R; i.e., $F1 = 2 \cdot P \cdot R/(P+R)$. This takes care of the trade-off between precision and recall. We also consider N+, the number of labels that are correctly assigned to at least one test image (in other words, the number of labels with recall greater than zero), as an evaluation metric. This measure is particularly useful in the case of class-imbalance, where frequent labels can suppress the recall of rare labels. Thus, different image annotation methods are compared using F1 and N+ scores.

Since image annotation has close parallels with the label ranking task, we additionally report mean average precision (mAP) scores for various methods.

6.2 Details

For the original features (without cross-modal embedding), we learn the distance metric using stochastic gradient descent on random batches of 1000 samples in leave-one-out manner. For 2PKNN without metric learning, the average distance using all the features is considered while determining the neighbours [similar to JEC (Makadia et al. 2008, 2010)]. Also, analogous to Guillaumin et al. (2009), this is scaled by a linear factor π that controls the decay of θ_{J,I_i} (Eq. 3). For each dataset, the K_1 parameter is set by doing cross-validation on training data in the range $\{1, 2, 3, 4, 5\}$. To evaluate JEC (Makadia et al. 2008, 2010), we implement this method following the steps outlined in the paper. For a given test image, in order to predict a ranking of labels rather than a fixed set of five labels, the number of nearest neighbours used is ensured to be sufficient to see enough unique labels [as followed in Makadia et al. (2008, 2010)]. To evaluate the variants of TagProp (Guillaumin et al. 2009), we use the publicly available code,[5] and cross-validate number of neighbours in the range $K = \{10, 20, \ldots, 200\}$. In case of common space learned using KCCA (Eq. 11), we use distance computed using both without and with the learned distance metric for TagProp and 2PKNN.

6.3 Comparisons and Discussion

Here, first we perform in-depth quantitative comparisons with two state-of-the-art nearest neighbour based image annotation methods JEC (Makadia et al. 2008, 2010) and the variants of TagProp (Guillaumin et al. 2009) under different settings. Then we compare our results with the reported results of the recent as well as some benchmark methods. Finally we discuss some qualitative results.

[5] The code is available at http://lear.inrialpes.fr/people/guillaumin/code.php.

6.3.1 Features and Feature Embeddings

In Table 3, we compare 2PKNN and 2PKNN with metric learning (2PKNN + ML) with JEC (Makadia et al. 2008, 2010), and all the four variants of TagProp (Guillaumin et al. 2009): (a) TagProp-SD that simply uses scaled average distance computed using different features, (b) TagProp-σSD that uses scaled average distance and label-specific sigmoid functions to boost the recall of rare labels, (c) TagProp-ML that learns a distance metric, and (d) TagProp-σML that learns both a distance metric as well as label-specific sigmoid functions. Using all the methods, we annotate each test image with five labels. We consider seven sets of features: (a) TagProp-features (denoted by T) that include fifteen features, (b) CNN-features (denoted by C) that include three features, (c) Encoding-features (denoted by E) that include two features, (d) Combined TagProp and CNN features (denoted by T + C) that include eighteen features, (e) Combined TagProp and Encoding features (denoted by T + E) that include seventeen features, (f) Combined CNN and Encoding features (denoted by by C + E) that include five features, and (g) Combined TagProp, CNN and Encoding features (denoted by T + C + E) that include twenty features.

From the table, we can observe that in case of TagProp, learning sigmoid functions usually improves the performance. Also, the performance improves with metric learning for both TagProp and 2PKNN. In general, we can notice that the best performing features usually vary for different methods, which indicates the importance of using different features for different methods. In most of the cases, we can observe that T + C + E features are more useful on the Corel-5K dataset, C + E on the ESP-Game dataset, T + C/T + C + E on the IAPR-TC12 dataset, and C + E on the MIRFlickr-25K dataset. From these results, we can conclude that combinations of both learned as well hand-crafted features can be useful in such settings.

In Table 4 and Table 5, we compare the performance of different methods by applying feature embedding using CCA and KCCA respectively. It should be noted that in case of CCA, we do not include metric learning since it involves a linear kernel. Also, in case of KCCA, we perform metric learning just once in the original space wherever applicable. From these results, we can make the following observations: (1) After CCA embedding, the performance generally reduces by a small amount compared to that using the original features. (2) Using KCCA embedding, the performance of all the methods generally improves in terms of F1 score (sometimes by a large margin) compared to both CCA embedding as well as original features, thus demonstrating the advantage of learning kernelized cross-modal embedding. (3) In some cases, the performance drops in terms of N+. This could be because the learned embedding does not efficiently capture the semantics of the labels with relatively low frequency.

Table 3 Performance comparison using different features

Dataset		Corel-5K				ESP-Game				IAPR-TC12				MIRFlickr-25K			
Method	Ftrs.	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+
JEC	T	31	36	33.3	148	24	19	21.2	220	31	20	24.3	219	27	24	25.4	36
	C	27	35	30.5	140	27	22	24.2	222	33	21	25.7	222	41	38	**39.4**	**37**
	E	22	26	23.8	126	23	18	20.2	215	24	17	19.9	203	25	25	25.0	36
	T+C	32	39	35.2	153	26	21	23.2	**224**	33	21	25.7	221	31	28	29.4	**37**
	T+E	32	36	33.9	150	25	20	22.2	223	31	21	25.0	220	28	25	26.4	**37**
	C+E	30	38	33.5	147	28	23	**25.3**	223	35	23	**27.8**	**228**	41	38	**39.4**	**37**
	T+C+E	34	41	**37.2**	**159**	26	21	23.2	222	33	22	26.4	222	32	29	30.4	**37**
TagProp-SD	T	31	31	31.0	122	38	21	27.1	218	49	20	28.4	201	47	29	35.9	34
	C	25	28	26.4	107	45	21	**28.6**	195	44	23	30.2	**206**	56	46	50.5	35
	E	20	24	21.8	115	39	17	23.7	202	43	18	25.4	196	41	29	34.0	33
	T+C	35	37	**36.0**	133	42	19	26.2	180	47	23	30.9	197	47	39	42.6	**36**
	T+E	32	34	33.0	133	40	20	26.7	**214**	50	20	28.6	202	46	30	36.3	33
	C+E	25	32	28.1	116	43	21	28.2	192	46	23	30.7	200	56	47	**51.1**	35
	T+C+E	34	38	35.9	**136**	42	18	25.2	178	48	23	**31.1**	202	51	41	45.5	35
TagProp-σSD	T	32	32	32.0	125	37	21	26.8	**220**	50	20	28.6	207	48	32	38.4	35
	C	25	28	26.4	109	45	21	28.6	195	44	24	31.1	**209**	57	46	50.9	36
	E	23	25	24.0	118	40	17	23.9	206	43	19	26.4	198	42	29	34.3	33
	T+C	35	37	36.0	138	46	21	**28.8**	203	49	23	**31.3**	207	54	48	50.8	**37**
	T+E	33	35	34.0	135	43	21	28.2	219	49	21	29.4	203	50	33	39.8	34
	C+E	26	32	28.7	119	44	21	28.4	199	47	23	30.9	203	56	47	51.1	35
	T+C+E	36	39	**37.4**	**142**	46	20	27.9	199	48	23	31.1	207	58	47	**51.9**	36
TagProp-ML	T	32	42	36.3	155	39	24	29.7	232	46	34	39.1	263	44	34	38.4	37
	C	30	42	35.0	156	37	32	34.3	**244**	42	34	37.6	262	59	50	**54.1**	**38**
	E	21	27	23.6	121	35	21	26.2	226	40	27	32.2	246	41	31	35.3	37
	T+C	36	48	**41.1**	**167**	38	30	33.5	230	49	37	42.2	268	51	41	45.5	**38**
	T+E	32	43	36.7	159	40	25	30.8	235	47	35	40.1	263	45	36	40.0	37
	C+E	31	43	36.0	154	38	32	**34.7**	243	45	37	40.6	263	59	50	**54.1**	**38**
	T+C+E	35	47	40.1	166	39	30	33.9	234	49	38	**42.8**	**270**	53	44	48.1	**38**
TagProp-σML	T	33	43	37.3	160	41	24	30.3	233	48	34	39.8	266	47	38	42.0	37
	C	31	43	36.0	157	38	33	35.3	245	43	35	38.6	266	59	50	54.1	**38**
	E	22	29	25.0	124	35	22	27.0	230	39	28	32.6	253	41	32	35.9	37
	T+C	36	50	41.9	**175**	39	31	34.5	244	48	39	**43.0**	**276**	59	50	54.1	**38**
	T+E	34	44	38.4	164	41	26	31.8	239	47	36	40.8	270	47	39	42.6	37
	C+E	32	44	37.1	160	39	33	**35.7**	**246**	45	38	41.2	267	59	51	**54.7**	**38**
	T+C+E	38	48	**42.4**	170	39	32	35.2	244	48	39	**43.0**	**276**	58	51	54.3	**38**
2PKNN	T	40	41	40.5	180	40	25	30.8	248	49	30	37.2	275	39	29	33.3	**38**
	C	39	45	41.8	189	50	29	**36.7**	**251**	54	31	39.4	271	57	45	50.3	**38**
	E	26	27	26.5	142	41	24	30.3	248	45	26	33.0	269	37	33	34.9	**38**
	T+C	42	42	42.0	187	56	19	28.4	243	56	29	38.2	275	46	33	38.4	**38**
	T+E	37	43	39.8	178	44	25	31.9	249	47	32	38.1	275	40	30	34.3	**38**
	C+E	40	43	41.4	**192**	53	27	35.8	246	53	33	**40.7**	**276**	56	46	**50.5**	**38**
	T+C+E	43	42	**42.5**	188	57	19	28.5	240	58	28	37.8	275	46	34	39.1	**38**

Table 3 continued

Dataset		Corel-5K				ESP-Game				IAPR-TC12				MIRFlickr-25K			
Method	Ftrs.	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+
2PKNN + ML	T	41	46	43.4	186	43	26	32.4	246	53	32	39.9	**277**	41	34	37.2	**38**
	C	38	47	42.0	189	47	31	37.4	253	51	33	40.1	273	58	46	51.3	**38**
	E	26	27	26.5	144	41	24	30.3	248	45	26	33.0	270	36	33	34.4	**38**
	T + C	40	51	**44.8**	**196**	49	30	37.2	254	49	36	**41.5**	**277**	58	46	51.3	**38**
	T + E	40	47	43.2	185	44	26	32.7	250	49	34	40.1	**277**	41	34	37.2	**38**
	C + E	39	47	42.6	192	49	31	**38.0**	**255**	48	36	41.1	276	57	47	**51.5**	**38**
	T + C + E	40	50	44.4	194	48	30	36.9	252	49	35	40.8	276	57	47	**51.5**	**38**

The best F1 and N+ scores for each method are highlighted in bold
T TagProp-features, *C* CNN-features, *E* Encoding-features, $T + C$ combined TagProp and CNN-features, $T + E$ combined TagProp and Encoding-features, $C + E$ combined CNN and Encoding-features, $T + C + E$ combined TagProp, CNN and Encoding-features

Table 4 Performance using feature embedding learned via CCA for different features

Dataset		Corel-5K				ESP-Game				IAPR-TC12				MIRFlickr-25K			
Method	Ftrs.	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+
JEC	T	29	31	30.0	131	26	18	21.3	216	29	12	17.0	188	34	28	30.7	**38**
	C	26	21	23.2	97	38	18	24.4	214	20	09	12.4	161	49	37	42.2	37
	E	22	24	23.0	114	34	18	23.5	212	39	16	22.7	188	37	29	32.5	37
	T + C	33	33	33.0	133	38	20	26.2	**223**	40	14	20.7	180	46	37	41.0	36
	T + E	31	32	31.5	133	34	19	24.4	219	38	19	25.3	200	38	31	34.1	**38**
	C + E	30	25	27.3	107	49	18	26.3	209	45	16	23.6	185	50	38	**43.2**	37
	T + C + E	35	34	**34.5**	**136**	43	20	**27.3**	**223**	44	19	**26.5**	**202**	46	38	41.6	36
TagProp-SD	T	28	25	26.4	110	37	22	27.6	230	40	25	30.8	244	34	32	33.0	**38**
	C	21	14	16.8	73	52	22	30.9	227	48	21	29.2	223	58	45	50.7	**38**
	E	21	20	20.5	93	34	22	26.7	231	46	23	30.7	227	40	36	37.9	37
	T + C	33	33	33.0	132	43	25	31.6	233	48	26	33.7	**245**	50	46	47.9	**38**
	T + E	30	32	31.0	130	38	25	30.2	232	48	27	34.6	239	38	37	37.5	**38**
	C + E	28	24	25.8	103	54	22	31.3	225	56	22	31.6	220	55	48	**51.3**	**38**
	T + C + E	33	35	**34.0**	**135**	46	26	**33.2**	**235**	52	27	**35.5**	241	49	48	48.5	**38**
TagProp-σSD	T	31	31	31.0	133	33	25	28.4	237	37	28	31.9	**258**	35	34	34.5	**38**
	C	27	22	24.2	102	47	25	32.6	237	46	25	32.4	245	57	48	52.1	**38**
	E	23	25	24.0	109	32	24	27.4	239	45	25	32.1	239	42	38	39.9	**38**
	T + C	33	34	33.5	133	41	27	32.6	**241**	47	30	36.6	256	51	48	49.5	**38**
	T + E	32	33	32.5	136	37	26	30.5	240	47	31	37.4	256	43	38	40.3	**38**
	C + E	29	25	26.9	108	53	23	32.1	231	58	24	34.0	235	58	49	**53.1**	**38**
	T + C + E	34	35	**34.5**	**139**	45	28	**34.5**	**241**	52	31	**38.8**	255	51	49	50.0	**38**
2PKNN	T	36	35	35.5	155	41	23	29.5	249	50	19	27.5	256	38	35	36.4	**38**
	C	37	32	34.3	154	50	25	33.3	243	50	14	21.9	244	57	47	51.5	**38**
	E	29	33	30.9	143	39	24	29.7	248	48	27	34.6	255	38	39	38.5	**38**
	T + C	41	41	41.0	170	44	27	33.5	**251**	57	22	31.7	250	48	47	47.5	**38**
	T + E	39	37	38.0	161	41	25	31.1	247	50	29	36.7	**266**	40	40	40.0	**38**
	C + E	40	37	38.4	159	58	25	34.9	247	60	26	36.3	256	53	51	**52.0**	**38**
	T + C + E	42	42	**42.0**	**176**	50	27	**35.1**	250	54	31	**39.4**	263	48	49	48.5	**38**

The best F1 and N+ scores for each method are highlighted in bold
T TagProp-features, *C* CNN-features, *E* Encoding-features, $T + C$ combined TagProp and CNN-features, $T + E$ combined TagProp and Encoding-features, $C + E$ combined CNN and Encoding-features, $T + C + E$ combined TagProp, CNN and Encoding-features)

Table 5 Performance of different methods using cross-modal embedding learned via KCCA for different feature combinations

Dataset		Corel-5K				ESP-Game				IAPR-TC12				MIRFlickr-25K			
Method	Ftrs.	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+
JEC	T	39	39	39.0	148	46	20	27.9	217	44	22	29.3	206	40	32	35.6	**38**
	C	34	34	34.0	132	48	23	31.1	219	46	22	29.8	211	53	41	**46.2**	**38**
	E	23	29	25.7	123	30	22	25.4	216	36	21	26.5	202	40	31	34.9	36
	T+C	41	43	**42.0**	**156**	47	23	30.9	219	47	24	**31.8**	**218**	47	40	43.2	**38**
	T+E	39	39	39.0	148	47	21	29.0	219	45	22	29.6	209	42	34	37.6	**38**
	C+E	32	35	33.4	131	50	23	31.5	218	46	23	30.7	210	51	42	46.1	37
	T+C+E	41	43	**42.0**	155	48	24	**32.0**	**224**	47	24	**31.8**	214	48	40	43.6	**38**
TagProp-SD	T	35	37	36.0	137	47	26	33.5	237	51	33	40.1	252	42	42	42.0	**38**
	C	32	34	33.0	130	50	30	37.5	**242**	53	32	39.9	250	51	53	**52.0**	**38**
	E	21	29	24.4	123	34	28	30.7	240	42	30	35.0	250	39	40	39.5	**38**
	T+C	27	22	24.2	96	47	29	35.9	239	53	36	42.9	**257**	48	51	49.5	**38**
	T+E	35	38	**36.4**	**140**	48	27	34.6	237	52	34	41.1	249	43	44	43.5	**38**
	C+E	31	32	31.5	116	52	30	**38.0**	238	54	34	41.7	252	50	53	51.5	**38**
	T+C+E	35	33	34.0	124	49	30	37.2	239	54	36	**43.2**	256	49	52	50.5	**38**
TagProp-σSD	T	37	40	38.4	145	46	28	34.8	241	50	37	42.5	261	41	42	41.5	**38**
	C	33	35	34.0	134	50	31	38.3	243	52	35	41.8	262	52	53	**52.5**	**38**
	E	22	30	25.4	125	30	31	30.5	242	39	34	36.3	261	38	40	39.0	**38**
	T+C	40	42	41.0	150	47	31	37.4	242	52	39	44.6	263	48	51	49.5	**38**
	T+E	35	36	35.5	136	48	28	35.4	243	52	37	43.2	260	41	43	42.0	**38**
	C+E	32	36	33.9	133	51	31	**38.6**	243	53	37	43.6	258	50	53	51.5	**38**
	T+C+E	41	43	**42.0**	**155**	48	31	37.7	**245**	53	40	**45.6**	**265**	49	51	50.0	**38**
TagProp-ML	T	34	35	34.5	**132**	50	25	33.3	235	50	35	41.2	256	43	45	44.0	**38**
	C	33	33	33.0	130	48	32	38.4	245	50	34	40.5	**262**	52	53	52.5	**38**
	E	21	28	24.0	121	31	28	29.4	237	45	29	35.3	241	40	40	40.0	37
	T+C	37	35	36.0	127	47	32	38.1	245	55	36	43.5	259	55	54	**54.5**	**38**
	T+E	33	34	33.5	130	53	24	33.0	233	51	34	40.8	250	44	45	44.5	**38**
	C+E	30	31	30.5	113	50	31	38.3	245	54	33	41.0	247	53	53	53.0	**38**
	T+C+E	37	36	**36.5**	131	49	32	**38.7**	**248**	56	36	**43.8**	254	55	54	**54.5**	**38**
TagProp-σML	T	37	39	38.0	144	50	26	34.2	240	48	39	43.0	263	40	44	41.9	37
	C	35	35	35.0	135	46	33	38.4	247	47	39	42.6	**274**	55	51	52.9	**38**
	E	21	29	24.4	123	28	31	29.4	242	44	33	37.7	256	40	40	40.0	**38**
	T+C	43	44	**43.5**	158	47	35	**40.1**	**249**	54	40	46.0	269	55	55	**55.0**	**38**
	T+E	34	37	35.4	140	53	25	34.0	237	50	38	43.2	262	42	43	42.5	**38**
	C+E	32	35	33.4	134	50	33	39.8	245	55	36	43.5	256	53	54	53.5	**38**
	T+C+E	42	45	43.4	**160**	48	34	39.8	**249**	56	40	**46.7**	268	55	55	**55.0**	**38**
2PKNN	T	45	44	44.5	176	45	29	35.3	251	50	37	42.5	273	43	44	43.5	**38**
	C	41	43	42.0	172	51	32	39.3	253	51	35	41.5	269	51	56	53.4	**38**
	E	32	34	33.0	155	39	30	33.9	251	45	32	37.4	272	41	43	42.0	**38**
	T+C	47	50	48.5	187	45	32	37.4	251	53	39	**44.9**	**279**	48	54	50.8	**38**
	T+E	45	46	45.5	179	45	30	36.0	252	51	38	43.6	272	43	46	44.4	**38**
	C+E	42	45	43.4	180	49	35	**40.8**	**255**	52	37	43.2	273	51	57	**53.8**	**38**
	T+C+E	48	50	**49.0**	**190**	43	33	37.3	250	53	38	44.3	275	48	55	51.3	**38**

Table 5 continued

Dataset		Corel-5K				ESP-Game				IAPR-TC12				MIRFlickr-25K			
Method	Ftrs.	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+
2PKNN + ML	T	45	47	46.0	186	45	29	35.3	249	52	37	43.2	272	39	42	40.4	**38**
	C	43	44	43.5	177	50	33	39.8	253	52	36	42.5	270	53	56	54.5	**38**
	E	32	34	33.0	155	39	30	33.9	251	45	32	37.4	272	41	44	42.4	**38**
	T + C	48	52	49.9	191	44	34	38.4	**255**	50	41	**45.1**	275	52	56	53.9	**38**
	T + E	45	48	46.5	185	46	31	37.0	251	53	38	44.3	272	41	43	42.0	**38**
	C + E	44	47	45.5	182	48	36	**41.1**	**255**	53	38	44.3	274	53	57	**54.9**	**38**
	T + C + E	48	52	**49.9**	**196**	46	33	38.4	254	50	40	44.4	**276**	52	56	53.9	**38**

The best F1 and N+ scores for each method are highlighted in bold
T TagProp-features, *C* CNN-features, *E* Encoding-features, $T + C$ combined TagProp and CNN-features, $T + E$ combined TagProp and Encoding-features, $C + E$ combined CNN and Encoding-features, $T + C + E$ combined TagProp, CNN and Encoding-features

(4) Using KCCA, the performance after metric learning usually improves for both TagProp and 2PKNN. This indicates that learned distance metric can benefit common embedding space. (5) The performance of 2PKNN + ML is consistently better than the other methods on the Corel-5K and ESP-Game datasets, comparable to TagProp-σML on the MIRFlickr-25K dataset, and inferior to TagProp-σSD and TagProp-σML on the IAPR-TC12 dataset by around 0.6 and 1.6 % respectively.

In all our subsequent analysis, we will use the features that give the best performance using the KCCA embedding, giving preference to F1 score over N+.

6.3.2 Tradeoff Between Precision and Recall

It is a popular and well-accepted practice among the image annotation methods (such as those listed in Table 6) to annotate each image with five labels for comparisons. However, it may not always be justifiable since the actual number of labels can vary significantly for different images. E.g., some (training) images in the ESP-Game and IAPR-TC12 datasets have up to 15 and 23 labels respectively. Assigning only 5 labels artificially restricts the number of labels that can be correctly recalled (N+), as well as the F1 score. Though this inspires us to increase the number of labels assigned per image, it remains ambiguous till what extent. Annotating each image with all the labels would ultimately result into the perfect recall and N+, however it would reduce the precision drastically and thus would not be practically useful. Hence it becomes important to analyse the tradeoff between precision and recall, as well as variation in N+ on increasing the number of labels assigned per image. We study these in Fig. 5 by increasing the number of labels assigned per image from one till the vocabulary size of a dataset. In Fig. 5a we compare methods that do not involve metric learning (JEC, TagProp-SD, TagProp-σSD and 2PKNN), and in Fig. 5b we compare methods that involve metric learning (TagProp-ML, TagProp-σML and 2PKNN + ML) along with 2PKNN. From the figure, we can make the following observations: (1) Generally both the precision and recall values increase up to a certain extent, and then precision starts to drop while recall continues to increase. This is expected since initially increasing the number of assigned labels also increases the number of correctly predicted labels. However, further increasing this results into increasing the number of incorrect predictions, and thus reduces the precision score. (2) In the beginning, 2PKNN remains above the three methods (JEC, TagProp-SD and TagProp-σSD) on the Corel-5K, IAPR-TC12 and MIRFlickr-25K datasets, and comparable on the ESP-Game dataset. On the Corel-5K dataset 2PKNN remains above the three methods for a very long range of recall. On the other three datasets, as precision starts to drop, the curve of 2PKNN gradually becomes comparable to that of TagProp-σSD and TagProp-SD, and remains so for some time. Then, as recall increases further, TagProp-σSD comes above both 2PKNN and TagProp-SD on the ESP-Game and IAPR-TC12 datasets, but remains comparable on the MIRFlickr-25K dataset. Also, by then JEC catches up with the other two methods 2PKNN and TagProp-SD. We try to explain these from the frequency of labels in a dataset (column 7 of Table 1). For the Corel-5K dataset, the average number of images per label is far less than the other three datasets. Thus, building small semantic neighbourhoods results into better performance of 2PKNN compared to the other methods. On the other hand, the average number of images per label for the MIRFlickr-25K dataset is quite large but the vocabulary size is too small. Due to this, the curves of all the methods become nearly comparable quite early. For the other two datasets, the average number of images per label is in between the Corel-5K and MIRFlickr-25K datasets. Thus, when the number of labels assigned per image is small, the semantic neighbourhoods of 2PKNN result into prediction of more diverse labels than the other three methods. On further increasing the number of labels assigned, the other three methods start predicting

Table 6 Comparison of our best results with the reported results of some of the benchmark and recent image annotation methods

Dataset	Corel-5K				ESP-Game				IAPR-TC12			
Method	P	R	F1	N+	P	R	F1	N+	P	R	F1	N+
CRM (Lavrenko et al. 2003)	16	19	17.4	107	–	–	–	–	–	–	–	–
MBRM (Feng et al. 2004)	24	25	24.5	122	18	19	18.5	209	24	23	23.5	223
InfNet (Metzler and Manmatha 2004)	17	24	19.9	112	–	–	–	–	–	–	–	–
NPDE (Yavlinsky et al. 2005)	18	21	19.4	114	–	–	–	–	–	–	–	–
SML (Carneiro et al. 2007)	23	29	25.7	137	–	–	–	–	–	–	–	–
TGLM (Liu et al. 2009)	25	29	26.9	131	–	–	–	–	–	–	–	–
JEC (Makadia et al. 2008, 2010)	27	32	29.3	139	23	19	20.8	227	25	16	19.5	196
MRFA (Xiang et al. 2009)	31	36	33.3	172	–	–	–	–	–	–	–	–
CCD (SVRMKL + KPCA) (Nakayama 2011)	36	41	38.3	159	36	24	28.8	232	44	29	35.0	251
GroupSparsity (Zhang et al. 2010)	30	33	31.4	146	–	–	–	–	32	29	30.4	252
BS-CRM (Moran and Lavrenko 2011)	22	27	24.2	130	–	–	–	–	24	22	23.0	250
RandomForest (Fu et al. 2012)	29	40	33.6	157	41	26	31.8	235	44	31	36.4	253
KSVM-VT (Verma and Jawahar 2013)	32	42	36.3	179	33	32	32.5	259	47	29	35.9	268
TagProp-SD (Guillaumin et al. 2009)	30	33	31.4	136	48	19	27.2	212	50	20	28.6	215
TagProp-σSD (Guillaumin et al. 2009)	28	35	31.1	145	39	24	29.7	232	41	30	34.6	259
TagProp-ML (Guillaumin et al. 2009)	31	37	33.7	146	49	20	28.4	213	48	25	32.9	227
TagProp-σML (Guillaumin et al. 2009)	33	42	37.0	160	39	27	31.9	239	46	35	**39.8**	266
FastTag (Chen et al. 2013)	32	43	36.7	166	46	22	29.8	247	47	26	33.9	280
SKL-CRM (Moran and Lavrenko 2014)	39	46	**42.2**	184	41	26	31.8	248	47	32	38.1	274
SVM-DMBRM (Murthy et al. 2014)	36	48	41.1	**197**	55	25	**34.4**	**259**	56	29	38.2	**283**
2PKNN (this work)	48	50	49.0	190	49	35	40.8	**255**	53	39	44.9	**279**
2PKNN + ML (this work)	48	52	**49.9**	**196**	48	36	**41.1**	**255**	50	41	**45.1**	275

The best F1 and N+ scores are highlighted in bold

the less frequent labels, that were earlier not assigned due to low prediction weight. This trend continues on increasing the number of labels assigned even further for all the methods except TagProp-σSD. This is because it uses label-specific sigmoid functions that boost the recall of rare labels and reduces that for frequent labels. This increase in recall also increases the precision of rare labels, and thus the average precision remains higher than that of the remaining methods. This implies that at higher recall values, TagProp-σSD can provide better performance than the other methods. However, assigning too many labels would reduce the practical utility of automatic prediction. (3) In Fig. 5b, we observe that 2PKNN + ML performs either comparable to or better than the metric learning based variants of TagProp (TagProp-ML and TagProp-σML) and 2PKNN, as also observed in Table 5.

The bottom rows in Fig. 5a, b show the variation in N+ on increasing the number of labels assigned per image. Compared to all the other methods, 2PKNN and 2PKNN + ML achieve the perfect N+ much earlier on all the four datasets. This is because they use semantic neighbourhoods for label propagation, that explicit the presence of all the labels among the selected neighbours. It can also be observed that though the sigmoid variants of TagProp (TagProp-σSD and TagProp-σML) achieve better N+ than the corresponding non-sigmoid variants towards the beginning, the latter outperform the former very soon. Moreover, it takes very long for the sigmoid variants to achieve the full N+. While the performance of TagProp-SD/ML is as expected, that of TagProp-σSD/σML is possibly because of the use of per-label sigmoid functions that boost the recall of rare labels, but at the cost of reducing that for the frequent ones.

6.3.3 Tradeoff Between Rare and Frequent Labels

In Fig. 6, we compare the annotation performance in terms of R and N+ on rare and frequent labels of each dataset, by assigning five labels to each test image. In Fig. 6a we compare methods that do not involve metric learning (JEC, TagProp-SD, TagProp-σSD and 2PKNN), and in Fig. 6b we compare methods that involve metric learning (TagProp-ML, TagProp-σML and 2PKNN + ML) along with 2PKNN. Recall that TagProp-σSD and TagProp-σML variants of TagProp (Guillaumin et al. 2009) learn a sigmoid function per label to boost the likelihood of rare labels. For comparison,

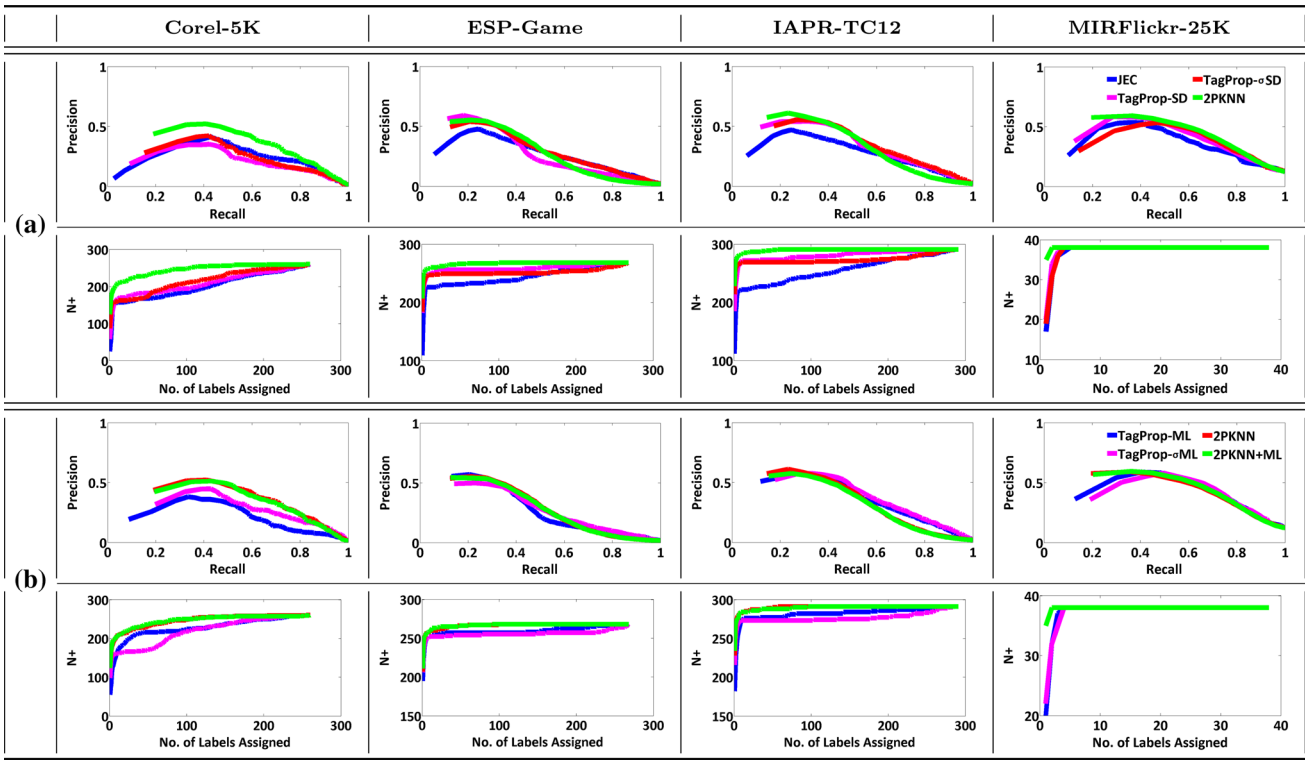

Fig. 5 Precision-versus-recall plots by varying the number of labels assigned to an image (*top row in each block*), and variation in N+ on varying the number of labels assigned to an image (*bottom row in each block*). (Best viewed in *colour*.) (Color figure online)

the labels are partitioned into two groups based on their frequency. The first partition consists of the 50 % least frequent (or rare) labels, and the second partition consists of the 50 % most frequent labels.

From the figure, we can make the following observations: (1) The JEC method (which is the simplest among all) provides very competitive performance on frequent labels. However, on rare labels it usually does not perform well. Compared to JEC, TagProp-SD usually performs better on all the datasets, except the Corel-5K where its performance is slightly inferior. This is possibly because TagProp-SD considers a larger neighbourhood for label prediction than JEC. However, in the Corel-5K dataset, the frequency of rare labels is too low compared to that of frequent labels (it has the lowest and the largest tail among all the four datasets, cf. Fig. 4). Due to this, using larger neighbourhoods results into giving more weight to frequent labels than rare labels, and thus reduces their prediction chances. (2) TagProp-σSD and TagProp-σML mostly provide better recall than TagProp-SD and TagProp-ML, thus demonstrating the advantage of learning label-specific sigmoid functions. (3) 2PKNN performs consistently better than TagProp-σSD on rare labels. This implies that building semantic neighbourhoods before label propagation can provide better boost for rare labels than learning label-specific sigmoid functions. (4) In general, the performance of each method on frequent labels is more than that on rare labels. However, the relative difference in the performance of 2PKNN and 2PKNN + ML on the two label-partitions is mostly less than the other three methods. This indicates that 2PKNN(+ML) addresses the class-imbalance problem better than the compared methods. (5) The performance of 2PKNN and 2PKNN + ML on rare labels is mostly better than the other methods, and that on frequent labels is either better than or comparable to the other methods. This shows that 2PKNN and 2PKNN + ML do not compromise much with the performance on frequent labels while gaining that on rare labels. (6) The performance of 2PKNN + ML is always either better than or comparable to 2PKNN on both the label-partitions for all the datasets. This confirms that our metric learning approach benefits both rare as well as frequent labels. From all the above observations, we can infer that 2PKNN along with metric learning can be a better option than either JEC or TagProp for the image annotation task.

6.3.4 Comparison with Previous Results

Table 6 summarizes our best results (from Table 5) as well as those reported by the previous methods. Similar to all other methods, we assign the top five labels (predicted using Eq. 2)

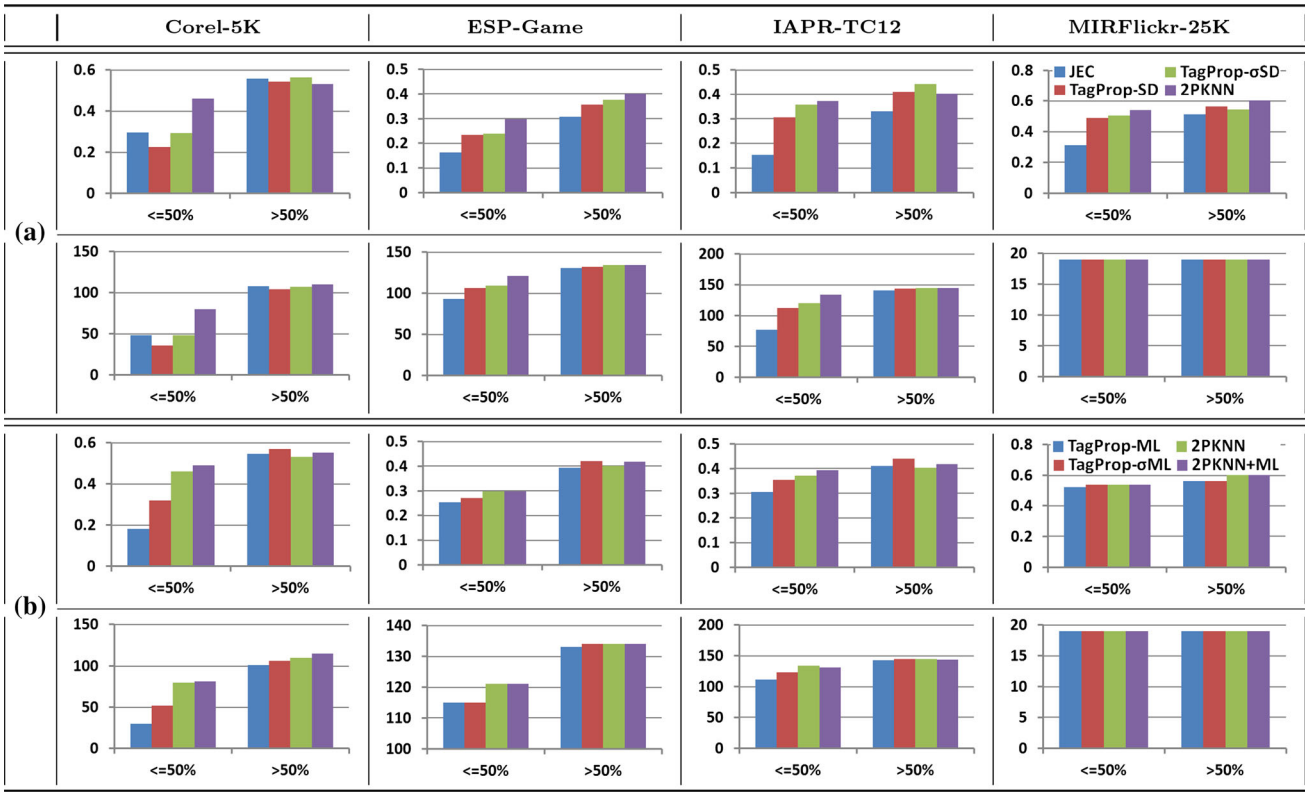

Fig. 6 Annotation performance in terms of R (*top row in each block*) and N+ (*bottom row in each block*) for different label partitions. The labels are grouped based on their frequency in a dataset (*horizontal axis*). This *first bin* corresponds to the subset of 50 % least frequent labels, and the *second bin* corresponds to the subset of 50 % most frequent labels. (Best viewed in *colour*.) (Color figure online)

to each test image, and the average precision and recall values are computed by averaging over all the labels in a dataset. Also, here we do not consider the MIRFlickr-25K dataset since the other three datasets are more popular and challenging, and none of the listed methods has evaluated on it. From these results, we can observe that on all the three datasets, the 2PKNN method significantly outperforms the previous methods in terms of F1 score. Precisely, for the Corel-5K, ESP-Game and IAPR- TC12 datasets, we achieve 7.7, 6.7 and 5.3 % of absolute improvements, and around 18.2, 19.5 and 13.3 % of relative improvements respectively over the previous best results. In terms of N+, our results are inferior only to the recent methods SVM-DMBRM (Murthy et al. 2014) on all the datasets, and FastTag (Chen et al. 2013) on the IAPR-TC12 dataset. However, since the F1 score of 2PKNN is much better than these two methods, this indicates that a higher N+ by these methods is probably because of correctly recalling only a small number of instances for several of the recalled labels. Whereas, though 2PKNN is able to recall a slightly less number of labels, the number of correctly recalled instances is much higher than the other two, thus resulting in better F1 score.

In Table 7, we compare our results with the recent work of Kalayeh et al. (2014). We compare with this work separately because their evaluation criteria is different from other methods compared in Table 6. Precisely, here also we annotate each test image with the top five labels. However, here we compute the average precision and recall values using only the labels with positive recall following Kalayeh et al. (2014). In terms of both F1 and N+ scores, our method significantly outperforms the NMF-KNN approach. Note that since the average precision and recall values are computed only for the correctly recalled labels, the N+ score acts as a divisive factor while computing the F1 score (from that computed for all the labels). Even with a big margin of 40/46

Table 7 Comparison of our best performance with the reported results of NMF-KNN (Kalayeh et al. 2014)

Dataset	Corel-5K				ESP-Game			
Method	P	R	F1	N+	P	R	F1	N+
NMF-KNN	38	56	45.3	150	33	26	29.1	238
2PKNN	65	68	66.5	190	51	37	42.9	255
2PKNN + ML	63	69	65.9	196	50	38	43.2	255

Similar to Kalayeh et al. (2014), here the average precision and recall values are computed using only the labels with positive recall, and not all the labels as followed by the methods in Table 6

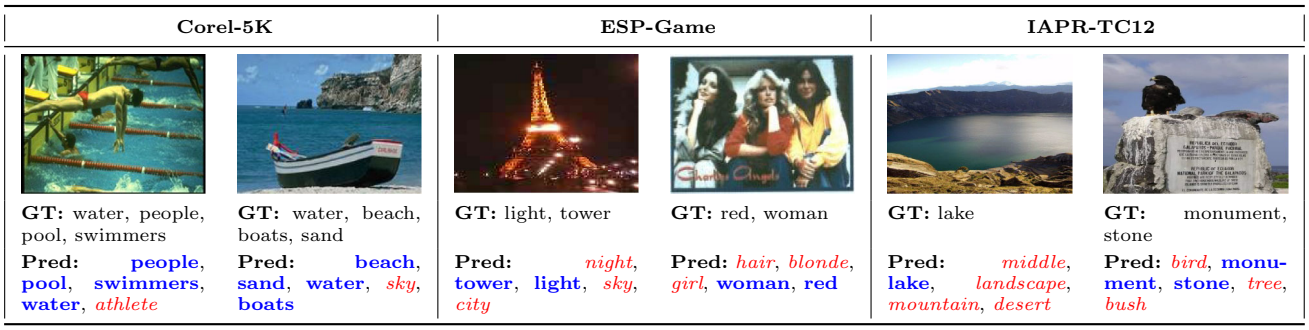

Fig. 7 Example images from the Corel-5K, ESP-Game and IAPR-TC12 datasets, along with the ground-truth set of labels (GT) and the top five labels predicted using 2PKNN + ML (Pred). The labels in **blue** (*bold*) are those that match with the ground-truth, while those in *red* (*italics*) are missing in the ground-truth though depicted in the corresponding images (Color figure online)

on the Corel-5K dataset and 17 on the ESP-Game dataset in terms of N+, our F1 scores are better than NMF-KNN by a factor of around 1.5.

From these comparisons, we can conclude that the 2PKNN and 2PKNN + ML methods along with the new image features achieve state-of-the-art results on all the three prevailing image annotation datasets in terms of F1 score, and competitive performance in terms of N+ score.

6.4 Qualitative Results

Though it is difficult to measure incomplete-labelling, we try to analyse it from qualitative results. Figure 7 shows a few examples of incompletely-labelled images from the Corel-5K, ESP-Game and IAPR-TC12 datasets, along with the top five labels predicted using 2PKNN + ML. It can be observed that for all these images, our method predicts all the ground-truth labels. Moreover, the additional labels predicted are actually depicted in the corresponding images, but missing in their ground-truth annotations. These results demonstrate that our method is capable of addressing the incomplete-labelling issue prevalent in the challenging real-world datasets.

6.5 Analysis

In this subsection, we analyse different aspects of the proposed approach, such as computational cost, diversity of labels among the neighbours obtained after the first pass of 2PKNN, effect of the parameter K_1 on performance, and label ranking performance. We also compare these aspects with appropriate methods wherever applicable.

6.5.1 Computational Cost

In Table 8, we compare the training and testing time of JEC, TagProp and 2PKNN. Our hardware configuration comprises Intel i7-4790K processor with 32 GB RAM. For the comparison, we consider $K = 200$ neighbours in TagProp, and $K_1 = 5$ in the first pass of 2PKNN, and use the combined TagProp, CNN and Encoding-features (T + C + E). For TagProp and 2PKNN, the training time denotes the time required for metric learning. Note that since there is no learning involved in JEC, the training time is not applicable.

Table 8 Computational cost of different methods using the combined TagProp, CNN and Encoding-features (in seconds)

	Training			Testing		
	JEC	TagProp	2PKNN	JEC	TagProp	2PKNN
Corel	NA	51.9	212.4	0.2	0.3	6.0
ESP	NA	1308.5	1397.2	0.8	2.5	53.8
IAPR	NA	1367.8	1336.6	0.7	2.3	54.4
Flickr	NA	609.3	360.4	2.9	15.1	76.7

For TagProp and 2PKNN, the training time denotes the time required for metric learning. For all the methods, the training and testing times do not take into consideration the time required for computing pair-wise distances. Since there is no learning involved in JEC, the training time is not applicable

Here, we can observe that the computational cost of 2PKNN is usually higher than JEC and TagProp. This is as expected since 2PKNN is a two-step process, and additionally requires semantic groups (sample-subsets based on label information) while identifying neighbours. Still, for the larger (ESP-Game and IAPR-TC12) datasets, its training cost is comparable to that of TagProp.

It should be noted that in all the comparisons in Table 8, we do not include the time required for computing pair-wise distances and performance evaluation, since these are the same for all, except JEC which is a training-free algorithm and does not require pair-wise distances among training samples. In practice, the operation of computing pair-wise distances can be highly parallelizable. However, we follow the simplest approach by computing pair-wise distances for each feature individually in parallel. Also, during distance computation, we do not perform any optimization. Due to this,

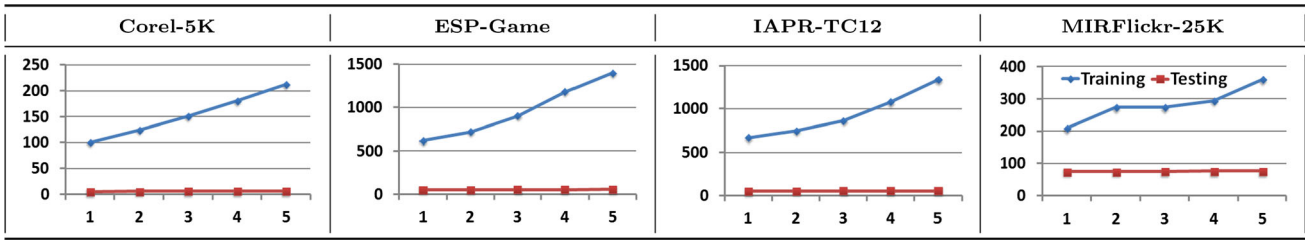

Fig. 8 Training and testing time (in seconds) for 2PKNN + ML (*vertical axis*) on varying the value of K_1 (*horizontal axis*), using the combined TagProp, CNN and Encoding-features

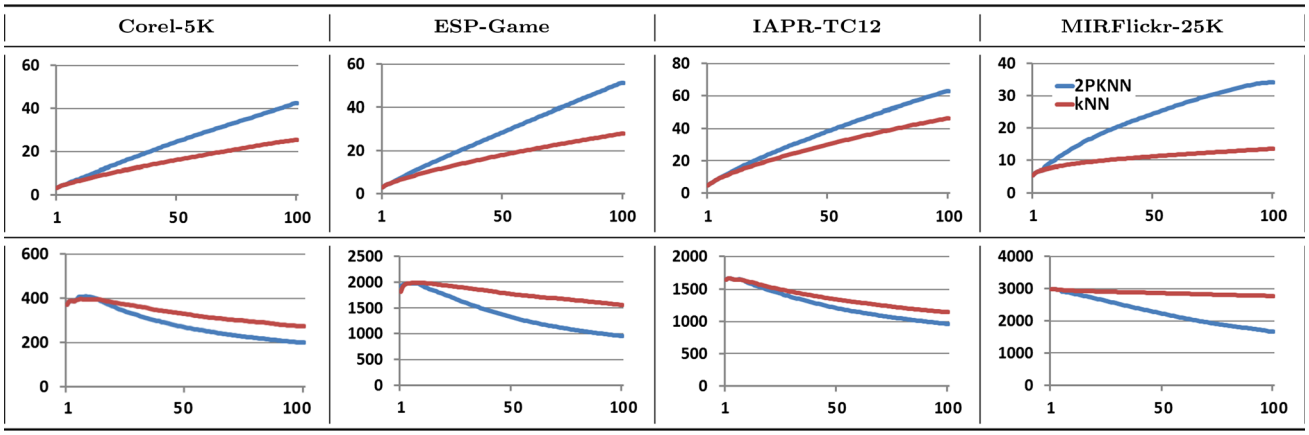

Fig. 9 Average number of distinct labels included in the neighbours (*top*), and average frequencies of these labels in the training set (*bottom*) on increasing the number of nearest neighbours (*horizontal axis*) after the first pass of 2PKNN and the conventional kNN algorithm. (Best viewed in *colour*.) (Color figure online)

the time required for computing pair-wise distances becomes quite significant. E.g., for the 65536-dimensional Fisher vector based features, it takes around 73 minutes for computing pair-wise distances among training samples, and around 14 minutes for computing pair-wise distances between training and testing samples on the Corel-5K dataset. As we can see, this time is comparatively much longer than training/testing time in Table 8, which makes the overall difference between the proposed and other methods relatively small.

In Fig. 8, we analyse the training and testing time of 2PKNN on varying the value of K_1 in $\{1, \ldots, 5\}$ using the same combination of features. Here, we can observe that while the increase in training time is almost linear for the Corel-5K, ESP-Game and IAPR-TC12 datasets (the datasets with large vocabularies), it is sub-linear for the MIRFlickr-25K which has a small vocabulary. Also, the increase in testing time as K_1 increases is very small, and is primarily affected by the number of test samples rather than the vocabulary size.

From these, we can conclude that the training (metric learning) of 2PKNN is quite scalable, and can be done in a reasonable time (in under 30 minutes) even for tens of thousands samples and hundreds of labels. The testing time, though substantially higher than the competing nearest neighbour based methods, is also fairly feasible for all practical applications.

6.5.2 Diversity of Labels in Neighbours

Now we try to empirically analyse the diversity of labels in the neighbours identified after the first pass of 2PKNN, and compare this with the conventional kNN algorithm. For 2PKNN, we vary the number of nearest neighbours from 1 to 100 on the subset of samples obtained after the first pass. For kNN, we vary the number of nearest neighbours in the same range but on the complete training set. In Fig. 9 (top), we calculate the number of distinct labels in the neighbours averaged over all test images. Here we can observe that using 2PKNN, we consistently get more diverse labels than kNN. In Fig. 9 (bottom), we calculate the frequency (in the full training set) of the distinct labels in the neighbours averaged over all test images. From this, we we can see that the average frequency using 2PKNN is consistently below kNN, thus indicating that using 2PKNN we can improve the occurrence of rare labels in the neighbours, whereas the influence of frequent labels is more than rare ones in kNN. From these, we can conclude that 2PKNN can help in retrieving neighbours

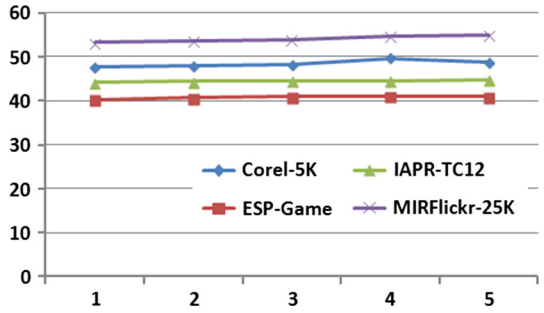

Fig. 10 Annotation performance of 2PKNN + ML in terms of F1 (*vertical axis*) on varying the value of the K_1 parameter (*horizontal axis*)

that contain more diverse and relatively less frequent labels than kNN.

6.5.3 Performance on Varying "K_1"

Analogous to kNN algorithm, K_1 is a hyper-parameter in 2PKNN that needs to be tuned by doing cross-validation. Figure 10 shows the influence of K_1 on the annotation performance on all the four datasets (in terms of F1 score). We can observe that though the performance varies on changing the value of K_1, it is not very sensitive to its choice and remains fairly stable.

6.5.4 Label Ranking Performance

Finally, in Table 9, we compare the label ranking performance of different methods using mAP as the evaluation metric. Here, we observe that the performance of 2PKNN(+ML) is far inferior than the TagProp variants. However, it is worth noticing that there are fundamental distinctions between the label ranking and image annotation tasks. First, in image annotation, there is no discrimination among the predicted labels based on their order/rank of prediction, whereas label ranking does make such a discrimination. Second, in image annotation, the performance (P, R, F1 and N+) is evaluated on a per-label basis, thus giving more importance to rare labels than the frequent ones. While in label ranking, the performance (mAP) is evaluated on a per-image basis, thus treating all the labels of an image equally. Since it is easier to predict frequent labels than the rare ones, this in turn contributes in increasing the overall mAP. This is also supported from the consistent drop in mAP on using the sigmoid variants of TagProp compared to non-sigmoid variants, that in fact gave better annotation performance (Table 5). Hence, mAP may not be as appropriate as F1 and N+ measures for comparing image annotation methods.

Table 9 Performance comparison in terms of mAP

Dataset	Corel	ESP	IAPR	Flickr
JEC	52.2	33.6	38.8	62.2
TagProp-SD	60.3	46.1	54.9	74.3
TagProp-σSD	59.1	45.5	53.7	68.1
TagProp-ML	65.1	49.2	55.4	73.6
TagProp-σML	61.1	47.2	54.7	71.6
2PKNN	53.4	46.0	54.0	69.4
2PKNN + ML	55.7	47.9	54.7	70.4

7 Conclusion

We have shown that the proposed 2PKNN method along with metric learning achieves either comparable or state-of-the-art results on the challenging image annotation datasets. Extensive analyses demonstrate that our method can be useful for annotation of natural image databases where frequencies of labels follow the long-tail phenomenon. We also showed that advanced feature extraction, encoding and embedding techniques can be useful in improving the performance of existing methods. We believe our work will provide a new platform for evaluating and comparing the future techniques in this domain.

Acknowledgements We thank Prof. Raghavan Manmatha for sharing the Corel-5K dataset, and the anonymous reviewers for their helpful comments. Yashaswi Verma is partially supported by Microsoft Research India PhD fellowship 2013.

Appendix

In our conference paper Verma and Jawahar (2012), we proposed to learn a combined distance metric in both distance as well as feature spaces, denoted by **w** and **v** respectively. While the dimensionality of **w** is equal to the number of different features used to represent a sample, that of **v** is equal to the combined dimensionality of all the individual features. As a result, we found it was not easy to learn **v** using limited examples as the number of features increased, and the computational cost was also quite high. Hence, in this paper we have considered only the **w** metric. Empirically, we found that the performance obtained using only the **w** metric was usually comparable to that using both **w** and **v**. E.g., in Table 10, we compare their performance using three feature combinations on the Corel-5K dataset. Here, we can observe that though we compromise a bit on the performance, the difference is not significant. More importantly, the training time improves by several orders of magnitude.

Table 10 Performance comparison in terms of F1, N+ and training time (in hours) between our previous metric learning approach (Verma and Jawahar 2012) that learns a combined metric in distance and feature spaces (2PKNN+ML (w+v)) with our current approach that learns a metric only in the distance space (2PKNN+ML(w)) on the Corel-5K dataset

Features	2PKNN+ML (w+v)			2PKNN+ML (w)		
	F1	N+	Time	F1	N+	Time
T	43.9	188	5.89	43.4	186	0.06
C	42.0	189	2.57	42.0	189	0.03
T+C	45.8	194	8.28	44.8	196	0.08

T TagProp-features, *C* CNN-features, *T*+*C* combined TagProp and CNN features)

References

Anderson, C. (2006). *The long tail: Why the future of business is selling less of more*. Hyperion.

Ballan, L., Uricchio, T., Seidenari, L., & Bimbo, A. D. (2014). A cross-media model for automatic image annotation. In *Proceedings of the ICMR*.

Carneiro, G., Chan, A. B., Moreno, P. J., & Vasconcelos, N. (2007). Supervised learning of semantic classes for image annotation and retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *29*(3), 394–410.

Chen, M., Zheng, A., & Weinberger, K. Q. (2013). Fast image tagging. In *Proceedings of the ICML*.

Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., et al. (2014). DeCAF: A deep convolutional activation feature for generic visual recognition. In *Proceedings of the ICML*.

Duygulu, P., Barnard, K., de Freitas, J. F., & Forsyth, D. A. (2002). Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In *Proceedings of the ECCV* (pp. 97–112).

Feng, S. L., Manmatha, R., & Lavrenko, V. (2004). Multiple bernoulli relevance models for image and video annotation. In *Proceedings of the CVPR* (pp. 1002–1009).

Fu, H., Zhang, Q., & Qiu, G. (2012). Random forest for image annotation. In *Proceedings of the ECCV* (pp. 86–99).

Grubinger, M. (2007). *Analysis and evaluation of visual information systems performance*. PhD thesis, Victoria University, Melbourne, Australia.

Guillaumin, M., Mensink, T., Verbeek, J. J., & Schmid, C. (2009). Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *Proceedings of the ICCV* (pp. 309–316).

Gupta, A., Verma, Y., & Jawahar, C. V. (2012). *Choosing linguistics over vision to describe images*. In *Proceedings of the AAAI*.

Hardoon, D. R., Szedmak, S., & Shawe-Taylor, J. (2004). Canonical correlation analysis: An overview with application to learning methods. *Neural Computation*, *16*(12), 2639–2664.

Hotelling, H. (1936). Relations between two sets of variates. *Biometrika*, *28*, 321–377.

Huiskes, M. J., & Lew, M. S. (2008). The MIR Flickr retrieval evaluation. In *MIR*.

Jégou, H., Douze, M., Schmid, C., & Pérez, P. (2010). Aggregating local descriptors into a compact image representation. In *Proceedings of the CVPR* (pp. 3304–3311).

Jeon, J., Lavrenko, V., & Manmatha, R. (2003). Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the ACM SIGIR* (pp. 119–126).

Jin, R., Wang, S., & Zhou, Z. H. (2009). Learning a distance metric from multi-instance multi-label data. In *Proceedings of the CVPR* (pp. 896–902).

Kalayeh, M. M., Idrees, H., & Shah, M. (2014). NMF-KNN: Image annotation using weighted multi-view non-negative matrix factorization. In *Proceedings of the CVPR*.

Lavrenko, V., Manmatha, R., & Jeon, J. (2003). A model for learning the semantics of pictures. In *NIPS*.

Li, X., Snoek, C. G. M., & Worring, M. (2009). Learning social tag relevance by neighbor voting. *IEEE Transactions on Multimedia*, *11*(7), 1310–1322.

Liu, J., Li, M., Liu, Q., Lu, H., & Ma, S. (2009). Image annotation via graph learning. *Pattern Recognition*, *42*(2), 218–228.

Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, *60*(2), 91–110.

Makadia, A., Pavlovic, V., & Kumar, S. (2008). A new baseline for image annotation. In *Proceedings of the ECCV* (pp. 316–329).

Makadia, A., Pavlovic, V., & Kumar, S. (2010). Baselines for image annotation. *International Journal of Computer Vision*, *90*(1), 88–105.

Metzler, D., & Manmatha, R. (2004). An inference network approach to image retrieval. In *Proceedings of the CIVR* (pp. 42–50).

Moran, S., & Lavrenko, V. (2011). Optimal tag sets for automatic image annotation. In *Proceedings of the BMVC* (pp. 1.1–1.11).

Moran, S., & Lavrenko, V. (2014). A sparse kernel relevance model for automatic image annotation. *International Journal of Multimedia Information Retrieval, 3*(4), 209–229.

Mori, Y., Takahashi, H., & Oka, R. (1999). Image-to-word transformation based on dividing and vector quantizing images with words. In *MISRM'99 first international workshop on multimedia intelligent storage and retrieval management*.

Murthy, V. N., Can, E. F., & Manmatha, R. (2014). A hybrid model for automatic image annotation. In *Proceedings of the ICMR*.

Nakayama, H. (2011). *Linear distance metric learning for large-scale generic image recognition*. PhD thesis, The University of Tokyo, Japan.

Oliva, A., & Torralba, A. (2001). Modeling the shape of the scene: A holistic representation of the spatial envelope. *International Journal of Computer Vision*, *42*(3), 145–175.

Perronnin, F., Sánchez, J., & Mensink, T. (2010). Improving the fisher kernel for large-scale image classification. In *Proceedings of the ECCV* (pp. 143–156).

Shalev-Shwartz, S., Singer, Y., & Srebro, N. (2007). Pegasos: Primal estimated sub-gradient solver for svm. In *Proceedings of the ICML* (pp. 807–814).

van de Weijer, J., & Schmid, C. (2006). Coloring local feature extraction. In *Proceedings of the ECCV* (pp. 334–348).

Verbeek, J., Guillaumin, M., Mensink, T., & Schmid, C. (2010). Image Annotation with TagProp on the MIRFLICKR set. In *MIR*.

Verma, Y., & Jawahar, C. V. (2012). Image annotation using metric learning in semantic neighbourhoods. In *Proceedings of the ECCV* (pp. 836–849).

Verma, Y., & Jawahar, C. V. (2013). Exploring SVM for image annotation in presence of confusing labels. In *Proceedings of the BMVC*.

von Ahn, L., & Dabbish, L. (2004). Labeling images with a computer game. In *SIGCHI conference on human factors in computing systems* (pp. 319–326).

Wang, C., Blei, D., & Fei-Fei, L. (2009). Simultaneous image classification and annotation. In *Proceedings of the CVPR*.

Wang, H., Huang, H., & Ding, C. H. Q. (2011). Image annotation using bi-relational graph of images and semantic labels. In *Proceedings of the CVPR* (pp. 793–800).

Weinberger, K. Q., & Saul, L. K. (2009). Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research*, *10*, 207–244.

Xiang, Y., Zhou, X., Chua, T. S., & Ngo, C. W. (2009). A revisit of generative model for automatic image annotation using markov random fields. In *Proceedings of the CVPR* (pp. 1153–1160).

Yavlinsky, A., Schofield, E., & Rüger, S. (2005). Automated image annotation using global features and robust nonparametric density estimation. In *Proceedings of the CIVR* (pp. 507–517).

Zhang, S., Huang, J., Huang, Y., Yu, Y., Li, H., & Metaxas, D. N. (2010). Automatic image annotation using group sparsity. In *Proceedings of the CVPR* (pp. 3312–3319).