
Web Scale Image Annotation: Learning to Rank with Joint Word-Image Embeddings

Abstract

Web scale image annotation datasets have tens of millions of images with tens of thousands of possible annotations. We propose a strongly performing method that scales to such datasets by simultaneously learning to optimize precision at k of the ranked list of annotations for a given image *and* learning a low-dimensional joint embedding space for both images and annotations. Our method both outperforms several baseline methods and, in comparison to them, provides a highly scalable architecture in terms of memory consumption and prediction time. We also demonstrate how our method learns an interpretable model, where annotations with alternate spellings or even languages are close in the embedding space. Hence, even when our model does not predict the exact annotation given by a human labeler, it often predicts similar annotations, a fact that we try to quantify by measuring the so-called “sibling” precision metric, where our method also obtains excellent results.

1. Introduction

The emergence of the web as a tool for sharing information has caused a massive increase in the size of potential datasets available for machines to learn from. Millions of images on web pages have tens of thousands of possible annotations in the form of HTML tags (which can be conveniently collected by querying search engines (Torralba et al., 2008a)), tags such as in www.flickr.com, or human-curated labels such as in www.image-net.org (Deng et al., 2009). We therefore need machine learning algorithms for image annotation that can scale to learn from such data. This includes: (i) scalable training and testing times, and (ii) scalable memory usage. In the ideal case we would

like a fast algorithm that fits on a laptop, at least at annotation time. For many recently proposed models tested on small datasets, e.g. (Makadia et al., 2008; Guillaumin et al., 2009), it is unclear if they satisfy these constraints.

In this work we study feasible methods for just such a goal. We consider models that learn to represent images and annotations jointly in a low dimension embedding space. Such embeddings are fast at testing time because the low dimension implies fast computations for ranking annotations. Simultaneously, the low dimension also implies small memory usage. To obtain good performance for such a model, we propose to train its parameters by learning to rank, optimizing for the top annotations in the list, e.g. optimizing precision at k . Unfortunately, such measures can be costly to train. To make training time efficient we propose the WARP loss (Weighted Approximate-Rank Pairwise loss) which can be seen as an efficient online version of the recently proposed OWPC loss (Usunier et al., 2009) which has been shown to be state-of-the-art on (small) text retrieval tasks.

The structure of the paper is as follows. Section 2 defines the embedding models that we employ. Section 3 defines the WARP loss and shows how to train our models with it. Section 4 describes how we perform our evaluation, including proposing a metric to measure the semantics of annotations in the case of tens of thousands of annotations, which we call the sibling precision. Section 5 details prior related work, Section 6 describes experiments conducted on web scale datasets, and Section 7 concludes.

2. Joint Word-Image Model

We propose to learn a mapping into a feature space where images and annotations are both represented. The mapping functions are therefore different, but are learnt jointly to optimize the supervised loss of interest for our final task, that of annotating images. We start with a representation of images $x \in \mathbb{R}^d$ and a representation of annotations $i \in \mathcal{Y} = \{1, \dots, Y\}$, indices into a dictionary of possible annotations. We then learn

a mapping from the image feature space to the joint space \mathbb{R}^D :

$$\Phi_I(x) : \mathbb{R}^d \rightarrow \mathbb{R}^D.$$

whilst jointly learning a mapping for annotations:

$$\Phi_W(i) : \{1, \dots, Y\} \rightarrow \mathbb{R}^D.$$

For simplicity these can be chosen to be linear maps, i.e. $\Phi_I(x) = Vx$ and $\Phi_W(i) = Wi$, where W_i indexes the i^{th} column of a $Y \times D$ matrix, but potentially any mapping could be used. In our work, we use sparse high dimensional feature vectors of bags-of-visual terms for image vectors x and each annotation has its own learnt representation (even if, for example, multi-word annotations share words). Hence, additional nonlinearities would not help for $\Phi_W(\cdot)$.

Our goal is, for a given image, to rank the possible annotations such that the highest ranked annotations best describe the semantic content of the image. We consider two possible models, the full-rank model:

$$f_i^{FULL}(x) = w_i x + \lambda_e s(\Phi_W(i), \Phi_I(x)) \quad (1)$$

and the low-rank model:

$$f_i^{LOW}(x) = s(\Phi_W(i), \Phi_I(x)) \quad (2)$$

where the possible annotations i are ranked according to the magnitude of $f_i(x)$, largest first, and $s(\cdot, \cdot)$ is a similarity function in the D -dimensional joint space, e.g. the inner product, or the negative Euclidean distance (in our experiments, we will employ the latter). In the full rank model each annotation i has an associated parameter vector $w_i \in \mathbb{R}^d$ to be learnt, as well as a shared parameter λ_e which indicates the amount of use of the shared low-rank embedding, and the parameters of the embedding themselves V and W . Essentially, the embedding term allows the model to learn to “multi-task” as these parameters are shared between labels. In the low-rank model only the parameters V and W are learnt. In the next section we describe the kind of loss functions we employ with our model, and thus subsequently the algorithm to train it.

3. Weighted Approximate-Rank Pairwise (WARP) Loss

We consider the task of ranking labels $i \in \mathcal{Y}$ given an example x . In our setting labeled data (x_i, y_i) will be provided for training where only a single annotation $y_i \in \mathcal{Y}$ is labeled correct¹. Let $f(x) \in \mathbb{R}^Y$ be a vector function providing a score for each of the labels, where $f_i(x)$ is the value for label i .

¹However, the methods described in this paper could easily be generalized to the multi-label case.

A class of ranking error functions was recently defined in (Usunier et al., 2009) as:

$$err(f(x), y) = L(rank_y(f(x))) \quad (3)$$

where $rank_y(f(x))$ is the rank of the true label y given by $f(x)$:

$$rank_y(f(x)) = \sum_{i \neq y} I(f_i(x) \geq f_y(x))$$

and $L(\cdot)$ transforms this rank into a loss:

$$L(k) = \sum_{j=1}^k \alpha_j, \text{ with } \alpha_1 \geq \alpha_2 \geq \dots \geq 0. \quad (4)$$

This class allows one to define different choices of $L(\cdot)$ with different minimizers. For example, minimizing L with $\alpha_j = \frac{1}{Y-1}$ would optimize the mean rank, $\alpha_1 = 1$ and $\alpha_{j>1} = 0$ the proportion of top-ranked correct labels, and larger values of α in the first few positions optimize the top k in the ranked list, which is of interest for optimizing precision at k or mean average precision (MAP). For example, given two images, if one choice of function ranks their true labels at position 1 and position 100 respectively, and another function both at position 50, then a choice of $\alpha_j = \frac{1}{Y-1}$ prefers these functions equally, whereas a choice of $\alpha_j = 1/j$ prefers the first function. Results on (small) text retrieval datasets in (Usunier et al., 2009) showed the latter choice of α yields state of the art results. We hence adopt the same choice but are interested in defining a method of optimizing such a loss function online, very efficiently, in order to train our image annotation models by stochastic gradient descent (SGD).

Online Learning to Rank The loss (3) is equal to:

$$err(f(x), y) = \sum_{i \neq y} L(rank_y(f(x))) \frac{I(f_i(x) \geq f_y(x))}{rank_y(f(x))}$$

with the convention $0/0 = 0$ when the correct label y is top-ranked. Using the hinge loss instead of the indicator function to add a margin and make the loss continuous, err can be approximated by:

$$\overline{err}(f(x), y) = \sum_{i \neq y} L(rank_y^1(f(x))) \frac{|1 - f_y(x) + f_i(x)|_+}{rank_y^1(f(x))} \quad (5)$$

where $|t|_+$ is the positive part of t and $rank_y^1(f(x))$ is the margin-penalized rank of y :

$$rank_y^1(f(x)) = \sum_{i \neq y} I(1 + f_i(x) > f_y(x)). \quad (6)$$

The overall risk we want to minimize is then:

$$Risk(f) = \int \overline{err}(f(x), y) dP(x, y). \quad (7)$$

An unbiased estimator of this risk can be obtained by stochastically sampling in the following way:

1. Sample a pair (x, y) according to $P(x, y)$;
2. For the chosen (x, y) sample a violating label \bar{y} such that $1 + f_{\bar{y}}(x) > f_y(x)$ and $\bar{y} \neq y$.

This chosen triplet (x, y, \bar{y}) has contribution:

$$\overline{err}(f(x), y, \bar{y}) = L(rank_y^1(f(x))) |1 - f_y(x) + f_{\bar{y}}(x)|_+ \quad (8)$$

to the total risk, i.e. taking the expectation of these contributions will approximate (7) because we have a probability of $1/rank_y^1(f(x))$ of drawing \bar{y} in step (2) which accounts for the denominator of (5).

This suggests for learning we can thus perform the following stochastic update procedure (Robbins & Monro, 1951) over the parameters β that define a family of possible functions $f \in \mathcal{F}$:

$$\beta(t+1) = \beta(t) - \gamma_t \frac{\partial \overline{err}(f(x), y, \bar{y})}{\partial \beta(t)}. \quad (9)$$

Weighted Approximate Ranking To perform the SGD described above we still have two problems that make this procedure inefficient:

- (i) In step (2), we need to compute the values $f_i(x)$ for $i = 1, \dots, Y$ to know which labels \bar{y} are violators, which is expensive for large Y .
- (ii) $rank_y^1(f(x))$ in (9) is also unknown without computing $f_i(x)$ for $i \in \mathcal{Y}$, which again is expensive.

We propose to solve both problems with the following approach: for step (2), we sample labels i with replacement until we find a violating label.

Now if there are $k = rank_y^1(f(x))$ violating pairs, the random variable N_k which counts the number of trials in our sampling step follows a geometric distribution of parameter $\frac{k}{Y-1}$ (i.e. $\Pr(N_k > q) = (1 - \frac{k}{Y-1})^q$). Thus $k = \frac{Y-1}{E[N_k]}$. This suggests that the value of $rank_y^1(f(x))$ in Equation (8) may be approximated by:

$$rank_y^1(f(x)) \approx \left\lfloor \frac{Y-1}{N} \right\rfloor$$

where $\lfloor \cdot \rfloor$ is the floor function and N the number of trials in the sampling step.

Algorithm 1 Online WARP Loss Optimization

Input: labeled data (x_i, y_i) , $y_i \in \{1, \dots, Y\}$.

repeat

 Pick a random labeled example (x_i, y_i)

 Set $N = 0$.

repeat

 Pick a random annotation $\bar{y} \in \{1, \dots, Y\} \setminus y_i$.

$N = N + 1$.

until $f_{\bar{y}}(x) > f_{y_i}(x) - 1$ or $N > Y - 1$

if $f_{\bar{y}}(x) > f_{y_i}(x) - 1$ **then**

 Make a gradient step to minimize:

$$L(\lfloor \frac{Y-1}{N} \rfloor) |1 - f_y(x) + f_{\bar{y}}(x)|_+$$

end if

until validation error does not improve.

Remark 1: We intuitively presented the sampling approach as an approximation of the SGD step of Equation (9). In fact, the sampling process gives an unbiased estimator of the risk (7) if we consider a new function \tilde{L} instead of L in Equation (5), with:

$$\tilde{L}(k) = E \left[L \left(\left\lfloor \frac{Y-1}{N_k} \right\rfloor \right) \right].$$

Elementary calculations show that \tilde{L} satisfies Equation (4). So our approach optimizes a new ranking error. One can also show that $\tilde{L}(k) \geq L(k)$ for all k , so \tilde{L} gives more weight to top-ranked labels than L .

Remark 2: The floor function in the approximation $rank_y^1(f(x)) \approx \lfloor \frac{Y-1}{N} \rfloor$ makes it useless to continue sampling after n unsuccessful trials. Thus, an SGD step requires less than $\min(\frac{Y-1}{rank_y^1(f(x))}, Y-1)$ computations of scores $f_i(x)$ on average. On difficult datasets, many correct labels will not have a rank close to 1, and even on easy datasets at the start of training the same will be true, so per-point updates will be much faster.

Training Our Models To summarize, our overall method which we call WSABIE (Web Scale Annotation by Image Embedding, pronounced “wasabi”) consists of the joint word-image embedding model of Section 2 trained with the WARP loss of Section 3. For the low-rank model (2) we are thus performing stochastic gradient descent directly as above. For the full-rank model (1) without additional regularization nothing prevents the embedding not being used at all, as one can already minimize the loss using the first term only. One could thus add to the objective function the term $\lambda_w \sum_i \|w_i\|^2$, where increasing λ_w gives preference to the embedding space (which is not regularized). As this would require the additional worry of choosing the hyperparameter λ_w , in our experiments we instead chose the following simpler (perhaps suboptimal) approach: we learn the two terms of (1) separately, and

then optimize the parameter λ_e (a line search) using the validation set. Pseudocode for training with WARP loss is given in Algorithm 1.

4. Evaluation and Sibling Precision

We measure in the experimental section the standard metrics of precision at the top k of the list ($p@k$) and mean average precision (MAP) for the algorithms we compare, which give more credit if the true annotation appears near the top of the list of possible annotations.

On our datasets, we have ten or a hundred thousand annotations, that can be semantically close to each other. In the extreme case, two different labels can be synonyms, translations or alternative spellings. Our model tries to capture this structure of the annotation set through the projection in the embedding space.

To evaluate the ability of our model to learn the semantic relations between labels from images, we propose a new metric called the sibling precision at k ($p_{sib}@k$). Suppose we have some ground-truth in the form of a matrix S , where $S_{ij} \in [0, 1]$ is a measure of semantic similarity between labels i and j ($S_{ij} = 1$ means that the words are semantically equivalent). Then, for a ranking $y^r = (y_1^r, \dots, y_Y^r)$, $p_{sib}@k$ is defined as:

$$p_{sib}@k(y^r, y) = \frac{\sum_{i=1}^k S_{y_i^r, y}}{k}.$$

When S is the identity matrix we recover the usual $p@k$ loss. Otherwise, $p_{sib}@k$ is a relaxation of $p@k$, as off-diagonal elements of S give credit when a prediction is semantically close to the true label. p_{sib} also measures the ability of the model to discover the whole semantic structure by considering the similarity of all the first k predicted labels. Notice that as a measure of “discovery” of the semantic relations, $p_{sib}@k$ is only meaningful when S is unknown during training.

In order to build S , we proceed as follows. We suppose we have a database of known relations between annotations of the form $\text{isa}(y_c, y_p)$ where y_p is a parent concept of y_c , e.g. $\text{isa}(\text{“toad”}, \text{“amphibian”})$. We then define two annotations as siblings if they share a “parent”:

$$S_{i,j} = \begin{cases} 1, & \text{if } i = j \vee \exists k : \text{isa}(i, k) \wedge \text{isa}(j, k) \\ 0, & \text{otherwise.} \end{cases}$$

In the databases we consider in Section 6, ImageNet already has reliable isa relations annotated in WordNet, and for Web we have obtained a similar but noisier proprietary set based on occurrences of patterns such as “X is a Y” on web pages. The median numbers of siblings per label are reported in Table 1.

5. Related Approaches

The problem of image annotation, including the related task of image classification, has been the subject of much research, mainly in the computer vision literature. However, mostly this research concentrates on tasks with a rather small number of classes, in part due to the availability of appropriate databases. Well known databases such as Caltech-256 (Griffin et al., 2007) and Pascal-VOC (Everingham et al., 2007) have a limited number of categories, ranging from 20 to 256. More recently, projects such as the TinyImage database (Torralba et al., 2008a) and ImageNet (Deng et al., 2009) have started proposing larger sets of annotated images with a larger set of categories, in the order of 10^4 different categories. Note that for now, even with these new large datasets, published results about image annotation or classification have concentrated on subsets pertaining to a few hundred different categories or less only, e.g. (Torralba et al., 2008a; Fergus et al., 2009). Much research in the literature has in fact concentrated on extracting better image features, then training independently simple classifiers such as linear or kernel SVMs for each category (for example (?)).

An alternative approach, championed by (Makadia et al., 2008; Torralba et al., 2008b; Guillaumin et al., 2009), is to use k -nearest neighbor in the image feature space. This has shown good annotation performance, in particular as the size of the training set grows. On the other hand, as the data grows, finding the exact neighbors becomes infeasible in terms of time and space requirements. Various approximate approaches have thus been proposed to alleviate this problem, ranging from trees to hashes, but can suffer from being fast but not precise, or precise but slow.

Embedding words in a low dimensional space to capture semantics is a classic (unsupervised) approach in text retrieval which has been adapted for image annotation before, for example PLSA has been used for images (Monay & Gatica-Perez, 2004) but has been shown to perform worse than (non-embedding based) supervised ranking models like PAMIR (Grangier & Bengio, 2008). Other related work includes learning embeddings for supervised document ranking (Bai et al., 2009) and for semi-supervised multi-task learning (Ando & Zhang, 2005; Loeff et al., 2009).

Several loss functions have also recently been proposed to optimize the top of the ranked list. Indeed the WARP loss we propose can be seen as an efficient online version of the OWPC loss of (Usunier et al., 2009). OWPC was compared to ListNet, AdaRank, SVM^{map} and RSVM for text retrieval, and was shown

to be state-of-the-art. However, to our knowledge none of these existing methods would scale to our setup as they either cannot be trained online, or do not avoid computing $f_i(x)$ for each $i \in \mathcal{Y}$ as the WARP loss does.

In terms of the sibling precision metric, WordNet has been used many times to calculate distances between concepts in the field of natural language processing, and has been used for image annotation to build voted classifiers (Torralba et al., 2008a; Torralba et al., 2008b). In our work we are concerned with measuring missing annotations and would consider using other similarity measures for that, not just from WordNet.

6. Experiments

6.1. Datasets

ImageNet Dataset ImageNet (Deng et al., 2009) is a new image dataset organized according to WordNet (Fellbaum, 1998). Concepts in WordNet, described by multiple words or word phrases, are hierarchically organized. ImageNet is a growing image dataset that attaches quality-controlled human-verified images to these concepts. We split the data into 2.5M images for training, 0.8M for validation and 0.8M for testing, removing duplicates between train, validation and test by throwing away test examples which had too close a nearest neighbor training or validation example in feature space. A baseline annotation method of assigning the most frequent annotation achieves a precision at 1 of $\sim 0.04\%$.

Web Image Dataset We had access to a very large proprietary database of images taken from the web, together with a very noisy annotation based on anonymized user click information, processed similarly to ImageNet. The most frequent annotation baseline achieves a precision at 1 of $\sim 0.01\%$.

Table 1 provides summary statistics of the number of images and labels for the ImageNet and Web datasets used in our experiments.

Table 1. Summary statistics of the datasets used in the experiments described in this paper.

Statistics	ImageNet	Web
Number of Training Images	2518604	9861293
Number of Test Images	839310	3286450
Number of Validation Images	837612	3287280
Number of Labels	15952	109444
Median Num Siblings per Label	12	143

6.2. Image Representation

In this work we focus on learning algorithms, not feature representations. Hence, for all methods we try,

we use the same sparse vector representation, following (Grangier & Bengio, 2008). This representation has been shown to perform very well on the related task of image ranking. Each image is first segmented into several overlapping square blocks at various scales. Each block is then represented by the concatenation of color and edge features. These are discretized into a dictionary of $d = 10,000$ blocks, by training KMeans on a large corpus of images. Each image can then be represented as a *bag of visual words*: a histogram of the number of times each visual word was present in the image, yielding vectors in \mathbb{R}^d with an average of $d_{\bar{v}} = 245$ non-zero values. It takes on average 0.5 seconds to extract these features per image.

6.3. Baselines

We compare our proposed approach to several baselines: k -nearest neighbors (k -NN), approximate k -NN, one-versus-rest large margin classifiers (One-Vs-Rest) of the form $f_y(x) = w_y x$ trained using the Passive Aggressive algorithm (Crammer et al., 2006), or the same models trained with a ranking loss instead, which we call PAMR^{IA} as it is like the PAMIR model used in (Grangier & Bengio, 2008) but applied to image annotation rather than ranking. For all methods, hyperparameters are chosen via the validation set.

We tested approximate k -NN (ANN) because k -NN is not scalable. There are many flavors of approximation (see, e.g (Torralba et al., 2008b)). We chose the following: a random projection at each node of the tree is chosen with a threshold to go left or right that is the median of the projected training data to make the tree balanced. After traversing p nodes we arrive at a leaf node containing $t \approx n/2^p$ of the original n training points from which we calculate the nearest neighbors. Choosing p trades off accuracy with speed.

6.4. Results

The results of comparing all the methods on ImageNet and Web are summarized in Tables 2 and 3. Further detailed plots of precision and sibling precision for ImageNet are given in Figure 1. WSABIE outperforms competing methods, apart from on ImageNet where k -NN is outperformed by WSABIE_{FULL} but not by WSABIE_{LOW} ($D = 300$), at least in terms of p@1. On the other hand, k -NN is not a realistically scalable method, while ANN ($p = 7$) performed much worse. k -NN fared less well on Web perhaps because of the increased noise or number of labels (i.e. there are less examples per class), hence we did not test ANN. Overall, WSABIE_{FULL} is marginally better than WSABIE_{LOW} while taking more resources. We give a deeper analysis

of the results, including time and space requirements in subsequent sections.

Table 2. Summary of Test Set Results on ImageNet. Precision at 1 and 10, Sibling Precision at 10, and Mean Average Precision (MAP) are given.

Algorithm	p@1	p@10	p _{sib} @10	MAP
k -NN	4.53%	1.50%	3.59%	6.47%
Approx. k -NN	1.55%	0.41%	1.69%	2.32%
One-vs-Rest	2.27%	1.02%	3.71%	5.17%
PAMIR ^{IA}	3.14%	1.26%	4.39%	6.43%
WSABIE _{FULL}	4.80%	1.67%	5.65%	8.97%
WSABIE _{LOW}	4.03%	1.48%	5.18%	7.75%

Table 3. Summary of Test Set Results on Web-data. Precision at 1 and 10, Sibling Precision at 10, and Mean Average Precision (MAP) are given.

Algorithm	p@1	p@10	p _{sib} @10	MAP
k -NN	0.30%	0.34%	5.97%	1.52%
One-vs-Rest	0.52%	0.29%	4.61%	1.45%
PAMIR ^{IA}	0.32%	0.16%	2.94%	0.83%
WSABIE _{FULL}	1.23%	0.49%	10.22%	2.60%
WSABIE _{LOW}	1.02%	0.43%	9.83%	2.26%

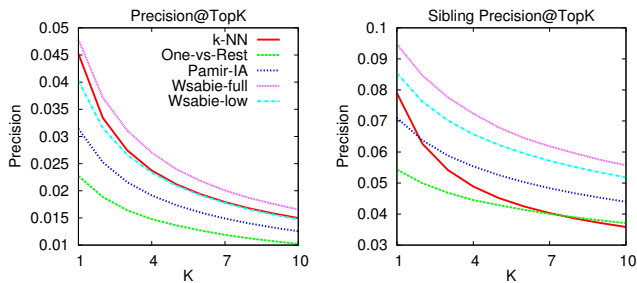


Figure 1. Precision@ k and Sibling Precision@ k on ImageNet for various methods.

Word-Image Embeddings Example word embeddings learnt by WSABIE_{LOW} for Web are given in Table 4 and some example image annotations are given in Table 9. One can see that the embeddings seem to learn the semantic structure of the annotation space (and images are also embedded in this space) and sibling annotations are close to each other. This explains why the sibling precision of WSABIE is far superior to competing methods, which do not attempt to learn the structure between annotations. One might ask how sensitive the results of WSABIE_{LOW} are to the choice of embedding dimension size. Table 5 shows the results are rather robust to choices of D .

WARP Loss We compared different models trained with either WARP or AUC minimization (via the margin ranking loss $|1 - f_y(x) + f_x(y)|_+$ as is used in PAMIR (Grangier & Bengio, 2008)). The results given in Table 6 show WARP consistently gives superior performance. We also compared training time using WARP (with eq. (2), $D = 100$), AUC or a naive implemen-

Table 5. Changing the Embedding Size on ImageNet. Test Error metrics when we change the dimension D of the embedding space used in WSABIE_{LOW}.

Embedding Dim.	p@1	p@10	p _{sib} @10	MAP
100	3.48%	1.39%	5.19%	7.12%
200	3.91%	1.47%	5.23%	7.66%
300	4.03%	1.48%	5.19%	7.75%
500	3.95%	1.44%	5.07%	7.58%

tation of SGD for (3) where the rank (6) is computed explicitly (note, in that case updates can be made for all violations for a given an image at once) which we call OWPC-SGD. For all methods we report their best learning rate γ . Figure 2 shows after 36 hours WARP and AUC are well trained, but AUC does not perform as well, and OWPC-SGD has hardly got anywhere. Full WARP models (eq. (1)) take several days to train.

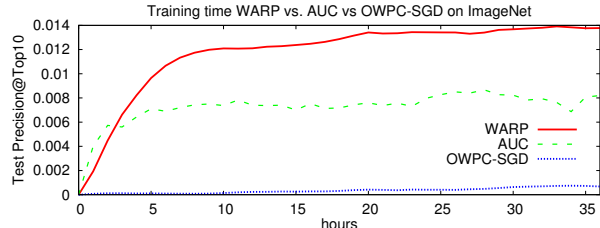


Figure 2. Training time: WARP vs. OWPC-SGD.

Table 6. WARP vs. AUC optimization. For each model choice, WARP consistently improves over AUC.

Model	Loss	p@1	p@10
Dataset: ImageNet			
$f_i(x) = s(\Phi_W(i), \Phi_I(x))$	AUC	1.65%	0.91%
$f_i(x) = s(\Phi_W(i), \Phi_I(x))$	WARP	4.03%	1.48%
$f_i(x) = w_i \cdot x$	AUC	3.14%	1.26%
$f_i(x) = w_i \cdot x$	WARP	4.25%	1.48%
Dataset: Web			
$f_i(x) = s(\Phi_W(i), \Phi_I(x))$	AUC	0.19%	0.13%
$f_i(x) = s(\Phi_W(i), \Phi_I(x))$	WARP	1.02%	0.43%
$f_i(x) = w_i \cdot x$	AUC	0.32%	0.16%
$f_i(x) = w_i \cdot x$	WARP	0.94%	0.40%

Computational Expense A summary of the test time and space complexity of the various algorithms we compare is given in Table 7 (not including cost of pre-processing of features) and concrete numbers on the particular datasets we use are given in Table 8 using a single computer, and assuming the data fits in memory (for WSABIE_{LOW} we give values for $D = 100$). In particular k -NN takes 255 days to compute the test error on ImageNet and 3913 days for Web, corresponding to 26 seconds and 103 seconds per image respectively, making it infeasible to use. In comparison, PAMIR^{IA} takes 0.07 and 0.5 seconds to compute and WSABIE_{LOW} takes 0.02 and 0.17 seconds, and also requires far less memory. In summary, WSABIE_{LOW} can be feasibly run on a laptop using limited resources whereas k -NN requires all the resources of an entire

Table 4. Nearest annotations in the embedding space learnt by WSABIE_{LOW} on Web-data. Translations (e.g. delphin, rosen) and alternative/misspellings and synonyms (beckam, mt fuji) have close embeddings. Other annotations are from similar visual images: e.g. Alessandro del Piero is a footballer, Bow Wow is a rapper who and an Appaloosa is an often white horse with black spots. Annotations in *blue+italics* are in our known sibling set.

Annotation	Neighboring Annotations
barack obama david beckham santa	<i>barak obama, obama</i> , barack, barrack obama, bow wow, george w bush, <i>berlusconi beckham</i> , david beckam, <i>alessandro del piero, del piero</i> , david becham, <i>fabio cannavaro santa claus</i> , papa noel, pere noel, <i>santa clause</i> , joyeux noel, tomte, santa klaus, <i>father christmas</i>
dolphin cows	delphin, dauphin, <i>whale</i> , delfin, delfini, baleine, <i>blue whale</i> , walvis, <i>bottlenose dolphin</i> , delphine <i>cattle</i> , shire, <i>dairy cows</i> , kuh, <i>horse, cow, shire horse</i> , kone, <i>holstein</i> , appaloosa, caballo, vache
rose pine tree	rosen, <i>hibiscus</i> , rose flower, <i>rosa</i> , roze, pink rose, <i>red rose, a rose</i> , fiori, <i>gerbera, white rose abies alba, abies, araucaria, pine</i> , neem tree, <i>oak tree, pinus sylvestris, western hemlock</i>
mount fuji eiffel tower	mt fuji, fuji, fujisan, fujiyama, <i>mountain</i> , zugspitze, fuji mountain, paysage, mount kinabalu <i>eiffel, tour eiffel</i> , la tour eiffel, <i>big ben</i> , paris, <i>blue mosque</i> , eifel tower, eiffel tour, paris france
ipod f18	i pod, <i>ipod nano, apple ipod</i> , ipod apple, new ipod, <i>ipod shuffle, ipod classic, apple iphone f 18</i> , eurofighter, f14, fighter jet, tomcat, mig 21, f 16, eurofighter typhoon, f22, fighter aircraft

cluster. Moreover as k -NN has time and space complexity $\mathcal{O}(n \cdot d_{\bar{o}})$, where n is the number of train examples and $d_{\bar{o}}$ is the number of non-zero features, as n increases its use of resources only gets worse, whereas the other algorithms do not depend on n at test time. WSABIE_{LOW} has a second advantage that it is hardly impacted if we were to choose a larger and denser set of features than the one we use, as it maps these features into a D dimensional space and the bulk of the computation is then in that space.

Table 7. Algorithm Time and Space Complexity. Time and space complexity needed to return the top ranked annotation on a single test set image, not including feature generation. Denote by Y the number of classes, n the number of train examples, d the image input dimension, $d_{\bar{o}}$ the average number of non-zero values per image, D the size of the embedding space, and p the depth of the tree for approximate k -NN.

Algorithm	Time Complexity	Space Complexity
k -NN	$\mathcal{O}(n \cdot d_{\bar{o}})$	$\mathcal{O}(n \cdot d_{\bar{o}})$
Approx. k -NN	$\mathcal{O}((p + n/2^p) \cdot d_{\bar{o}})$	$\mathcal{O}(n \cdot d)$
One-vs-Rest	$\mathcal{O}(Y \cdot d_{\bar{o}})$	$\mathcal{O}(Y \cdot d)$
PAMIR ^{IA}	$\mathcal{O}(Y \cdot d_{\bar{o}})$	$\mathcal{O}(Y \cdot d)$
WSABIE _{FULL}	$\mathcal{O}(Y \cdot d_{\bar{o}})$	$\mathcal{O}(c \cdot d)$
WSABIE _{LOW}	$\mathcal{O}((Y + d_{\bar{o}}) \cdot D)$	$\mathcal{O}((Y + d) \cdot D)$

Table 8. Test Time and Memory Constraints. Test time (d=days, h=hours) and memory requirement needed to return the top ranked annotation on the test set for ImageNet and Web, not including feature generation.

Algorithm	ImageNet		Web	
	Time	Space	Time	Space
k -NN	255 d	6.9 GB	3913 d	27.1 GB
Approx. k NN	2 d	7 GB	-	-
One-vs-Rest	17 h	1.2 GB	19 d	8.2 GB
PAMIR ^{IA}	17 h	1.2 GB	19 d	8.2 GB
WSABIE _{FULL}	17 h	1.2 GB	19 d	8.2 GB
WSABIE _{LOW}	5.6 h	12 MB	6.5 d	82 MB




Analysis of Nearest Neighbor Our version of approximate nearest neighbor (ANN) performed rather

poorly, whereas the true nearest neighbor (NN) performs reasonably well, but suffers from prohibitive computational expense and use of memory. These results seem deserving of further analysis. Indeed, positive results using NN have been observed before in smaller image annotation datasets (Makadia et al., 2008). ANN can be tuned depending on the test time vs accuracy tradeoff one wishes to achieve, i.e. with enough computational expense it can arbitrarily close to the accuracy of NN. We investigated how good ANN would have to be on ImageNet. Let’s say our algorithm only uses 1-nearest neighbor. If ANN retrieves the true closest neighbor 100% of the time we get a p@1 of 4.5% (note, we cannot beat this result by much by increasing k). If we retrieve it 50% of the time, and otherwise we get the second neighbor without fail 100% of the time the p@1 degrades to 3.4%. For 25% instead of 50%, we get 2.8%. If we only have a 25% chance of catching any particular neighbor (not just the first) we get 2.0%. For a 10% chance instead, we get 1.4%. Indeed in our implementation we get around 1.55% when we traverse a tree until we have around 20,000 points at the leaf. If we have around 2500 points at each leaf (so it is much faster) this goes down to 0.89%. If we have around 40,000 points at each leaf, we have around 1.94%. In conclusion, approximations of NN can be brittle because retrieving the first neighbor, at least on this dataset, is very important for achieving good accuracy. Indeed exactly the same phenomena has been observed before on other datasets, see e.g. Table 3 of (Makadia et al., 2008). As our feature space is high dimensional it is not surprising we do not retrieve the exact neighbor often, in line with previous literature, see e.g. Table 3 of (Torralba et al., 2008b).

7. Conclusions

We have introduced a scalable model for image annotation based upon learning a joint representation

Table 9. Examples of the top 10 annotations of three compared approaches: PAMIR^{IA}, One-vs-Rest and WSABIE_{LOW}, on the Web dataset. Annotations in **red+bold** are the true labels, and those in *blue+italics* are so-called siblings.

Image	PAMIR ^{IA}	One-vs-Rest	WSABIE _{LOW}
	bora, free willy, su, orka, world-wide, sunshine coast, bequia, tioman island, universal remote montagna, esperar, <i>bottlenose dolphin</i>	surf, bora, belize, sea world, balena, wale, tahiti, delfini, surfing, <i>mahi mahi</i>	delfini, <i>orca</i> , dolphin , mar, delfin, dauphin, <i>whale</i> , cancan, <i>killer whale</i> , sea world
	air show, st augustine, stade, concrete architecture, streetlight, doha qatar, skydiver, <i>tokyo tower</i> , sierra sinn, lazaro cardenas	eiffel tower , <i>tour eiffel</i> , snowboard, blue sky, <i>empire state building</i> , luxor, <i>eiffel</i> , <i>lighthouse</i> , jump, adventure	eiffel tower , <i>statue</i> , <i>eiffel</i> , mole antonelianna, la tour eiffel, london, cctv tower, <i>big ben</i> , calatrava, <i>tokyo tower</i>
	chris hanson, michael johns, ryan guettler, richard marx, depardieu, barack hussein obama, freddie vs jason, dragana, shocking, falco	falco, barack, daniel craig, obama , <i>barack obama</i> , kanye west, pharrell williams, 50 cent, barrack obama, bono	barrack obama, <i>barack obama</i> , hussein obama, <i>barack obama</i> , james marsden, <i>jay z</i> , obama , nelly, falco, barrack

of images and annotations that optimizes top-of-the-list ranking measures and shown how it improves over several baselines in both accuracy and efficiency on web scale datasets. Evaluation using the sibling precision metric shows that our embedding method learns the semantic structure of the annotation space, which helps lead to its good performance.

References

- Ando, R. K., & Zhang, T. (2005). A framework for learning predictive structures from multiple tasks and unlabeled data. *JMLR*, 6, 1817–1953.
- Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Cortes, C., & Mohri, M. (2009). Polynomial semantic indexing. *Advances in Neural Information Processing Systems (NIPS 2009)*.
- Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., & Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7, 551–585.
- Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A Large-Scale Hierarchical Image Database. *IEEE Conference on Computer Vision and Pattern Recognition*.
- Everingham, M., Gool, L. V., Williams, C. K. I., Winn, J., & Zisserman, A. (2007). The PASCAL Visual Object Classes Challenge 2007 (VOC2007).
- Fellbaum, C. (Ed.). (1998). *WordNet: An electronic lexical database*. MIT Press.
- Fergus, R., Fei-Fei, L., Perona, P., & Zisserman, A. (2005). Learning object categories from google’s image search. *Proceedings of the 10th International Conference on Computer Vision, Beijing, China* (pp. 1816–1823).
- Fergus, R., Weiss, Y., & Torralba, A. (2009). Semi-supervised learning in gigantic image collections. *Advances in Neural Information Processing Systems, 2009*.
- Grangier, D., & Bengio, S. (2008). A discriminative kernel-based model to rank images from text queries. *Transactions on Pattern Analysis and Machine Intelligence*, 30, 1371–1384.
- Griffin, G., Holub, A., & Perona, P. (2007). *Caltech-256 object category dataset* (Technical Report 7694). California Institute of Technology.
- Guillaumin, M., Mensink, T., Verbeek, J., Schmid, C., LEAR, I., & Kuntzmann, L. (2009). Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. *ICCV*.
- Loeff, N., Farhadi, A., Endres, I., & Forsyth, D. (2009). Unlabeled Data Improves Word Prediction. *ICCV ’09*.
- Makadia, A., Pavlovic, V., & Kumar, S. (2008). A new baseline for image annotation. *European conference on Computer Vision (ECCV)*.
- Monay, F., & Gatica-Perez, D. (2004). PLSA-based image auto-annotation: constraining the latent space. *Proceedings of the 12th annual ACM international conference on Multimedia* (pp. 348–351).
- Robbins, H., & Monro, S. (1951). A stochastic approximation method. *Annals of Mathematical Statistics*, 22, 400–407.
- Torralba, A., Fergus, R., & Freeman, W. T. (2008a). 80 million tiny images: a large dataset for non-parametric object and scene recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30, 1958–1970.
- Torralba, A. B., Fergus, R., & Weiss, Y. (2008b). Small codes and large image databases for recognition. *CVPR*. IEEE Computer Society.
- Usunier, N., Buffoni, D., & Gallinari, P. (2009). Ranking with ordered weighted pairwise classification. *Proceedings of the 26th International Conference on Machine Learning* (pp. 1057–1064). Montreal: Omnipress.