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

Jason Weston[1], Samy Bengio[1], and Nicolas Usunier[2]

1 Google, USA
2 Université Paris 6, LIP6, France
{jweston,bengio}@google.com nicolas.usunier@lip6.fr

Abstract. Image annotation datasets are becoming larger and larger, with tens of millions of images and 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, is faster and consumes less memory. 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 newly introduced "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 [1]), tags such as in `www.flickr.com`, or human-curated labels such as in `www.image-net.org` [2]. We therefore need machine learning algorithms for image annotation that can scale to learn from and annotate 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. [3, 4], 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). The WARP loss is related to the recently proposed Ordered Weighted Pairwise Classification (OWPC) loss [5] which has been shown to be state-of-the-art on (small) text retrieval tasks. WARP uses stochastic gradient descent and a novel sampling trick to approximate ranks resulting in an efficient online optimization strategy which we show is superior to standard stochastic gradient descent applied to the same loss, enabling us to train on datasets that do not even fit in memory. Moreover, WARP can be applied to our embedding models (in fact, to arbitrary differentiable models) whereas the OWPC loss, which relies on SVM_{struct} cannot.

The model we propose learns the semantic structure of both word sequences (annotations) and images, and we observe semantically consistent rankings due to semantically similar annotations (and hence also images) appearing close in the embedding space. Given tens of thousands of possible labels it is unlikely that a human annotator will have assigned all the correct possible labels to an image, hence we cannot always be sure if a prediction is wrong. Moreover, when a prediction is wrong it would be nice to know if it is completely wrong ("motorcar" instead of "tuna") or semantically almost right ("tuna" instead of "salmon"). To quantify this we propose a novel metric, the sibling precision, which gives credit for semantically similar predictions to the correct one by using a collected dataset of similarities between annotations.

Overall the novelty of the paper is:

(i) reporting image annotation results on a larger scale than ever previously reported (10 million training examples and 100 thousand annotations);

(ii) showing for the first time the utility of optimizing precision at k for image annotation;

(iii) proposing a large scale algorithm for (approximately) optimizing precision at k (WARP loss);

(iv) showing that our embedding model yields low memory usage, fast computation time, and learns the semantic structure of similarly annotated images as measured by the sibling precision;

(v) showing that using an embedding model trained with the WARP loss yields better performance than any known competing approach for this task.

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 the newly proposed sibling precision metric. Section 5 details prior related work, Section 6 describes experiments conducted on large 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 $i \in \mathcal{Y} = \{1, \ldots, Y\}$, indices into a dictionary of possible annotations. We then learn a mapping from the image feature space to the joint space $\mathbb{R}^D$:

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

while jointly learning a mapping for annotations:

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

These are chosen to be linear maps, i.e. $\Phi_I(x) = Vx$ and $\Phi_W(i) = W_i$, where W_i indexes the i^{th} column of a $D \times Y$ 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).

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 the following model:

$$f_i(x) = \Phi_W(i)^\top \Phi_I(x) = W_i^\top V x \tag{1}$$

where the possible annotations i are ranked according to the magnitude of $f_i(x)$, largest first, and our family of models have constrained norm:

$$||V_i||_2 \leq C, \quad i = 1, \ldots, d, \tag{2}$$

$$||W_i||_2 \leq C, \quad i = 1, \ldots, Y. \tag{3}$$

which acts as a regularizer in the same way as is used in lasso [6]. In the next section we describe the kind of loss function 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 pairs (x, y) will be provided for training where only a single annotation $y_i \in \mathcal{Y}$ is labeled correct[1]. 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.

A class of ranking error functions was recently defined in [5] as:

$$err(f(x), y) = L(rank_y(f(x))) \tag{4}$$

[1] However, the methods described in this paper could be generalized to the multi-label case, naively by averaging the loss over all positive labels.

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

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

where I is the indicator function, and $L(\cdot)$ transforms this rank into a loss:

$$L(k) = \sum_{j=1}^{k} \alpha_j, \text{ with } \alpha_1 \geq \alpha_2 \geq \cdots \geq 0. \tag{5}$$

This class of functions allows one to define different choices of $L(\cdot)$ with different minimizers. 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, which gives superior precision at 1.

The authors of [5] used this loss (calling their method OWPC) in an SVM$_{struct}$ formalism to train on (small) text retrieval datasets, showing experimentally that the choice of $\alpha_j = 1/j$ yields state-of-the-art results measuring precision at k. We hence adopt the same choice but are interested in a method that can: (i) train embedding models of the form (1) which cannot be trained using SVM$_{struct}$; and (ii) can be trained efficiently online when the data will not even fit into memory. In the following section we show how this can be done by employing a novel sampling trick to make stochastic gradient descent (SGD) feasible for optimizing precision at k for arbitrary differentiable models, including our embedding model formulation.

Online Learning to Rank The loss (4) is equal to:

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

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:

$$\overline{err}(f(x), y) = \sum_{i \neq y} L\left(rank^1_y(f(x))\right) \frac{|1 - f_y(x) + f_i(x)|_+}{rank^1_y(f(x))} \tag{6}$$

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

$$rank^1_y(f(x)) = \sum_{i \neq y} I(1 + f_i(x) > f_y(x)). \tag{7}$$

The overall risk we want to minimize is then:

$$Risk(f) = \int \overline{err}(f(x), y) dP(x, y). \tag{8}$$

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 $1 + f_{\bar{y}}(x) > f_y(x)$.

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

$$\overline{err}_{\bar{y}}(f(x), y, \bar{y}) = L(rank_y^1(f(x))) \, |1 - f_y(x) + f_{\bar{y}}(x)|_+ \tag{9}$$

to the total risk, i.e. taking the expectation of these contributions approximates (8) because we have probability $1/rank_y^1(f(x))$ of drawing $\bar{y}$ in step (2) (or a contribution of 0 if $rank_y^1(f(x)) = 0$) which accounts for the denominator of (6).

This suggests for learning we can thus perform the following stochastic update procedure [7] over the parameters β that define a family of possible functions $f \in \mathcal{F}$:

$$\beta_{t+1} = \beta_t - \gamma_t \frac{\partial \overline{err}(f(x), y, \bar{y})}{\partial \beta_t}. \tag{10}$$

where γ_t is the learning rate.

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 $i = 1, \ldots, Y$ to know which labels $\bar{y}$ are violators, which is expensive for large Y.
(ii) $rank_y^1(f(x))$ in (10) 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 uniformly with replacement until we find a violating label.

Now if there are $k = rank_y^1(f(x))$ violating labels, 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 (9) may be approximated by:

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

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

Remark 1: We intuitively presented the sampling approach as an approximation of the SGD step of Equation (10). In fact, the sampling process gives an unbiased estimator of the risk (8) if we consider a new function $\tilde{L}$ instead of L in Equation (6), with:

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

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)
  Let $f_{y_i}(x_i) = \Phi_W(y_i)^\top \Phi_I(x_i)$
  Set $N = 0$.
  repeat
    Pick a random annotation $\bar{y} \in \{1, \dots, Y\} \setminus y_i$.
    Let $f_{\bar{y}}(x_i) = \Phi_W(\bar{y})^\top \Phi_I(x_i)$
    $N = N + 1$.
  until $f_{\bar{y}}(x_i) > f_{y_i}(x_i) - 1$ or $N \geq Y - 1$
  if $f_{\bar{y}}(x_i) > f_{y_i}(x_i) - 1$ **then**
    Make a gradient step to minimize:
      $L(\lfloor \frac{Y-1}{N} \rfloor)|1 - f_y(x_i) + f_{\bar{y}}(x_i)|_+$
    Project weights to enforce constraints (2)-(3).
  end if
until validation error does not improve.

Elementary calculations show that $\tilde{L}$ satisfies Equation (5). So our approach optimizes a new ranking error.

Remark 2: The floor function in the approximation $rank^1_y(f(x)) \approx \lfloor \frac{Y-1}{N} \rfloor$ makes it useless to continue sampling after $Y-1$ unsuccessful trials. Thus, an SGD step requires less than $1 + \min(\frac{Y-1}{rank^1_y(f(x))}, Y-1)$ computations of scores $f_i(x)$ on average. If the true annotation happens to be at the top of the list for that particular image our method is as slow as standard SGD (i.e. computing the rank (7) explicitly). However at the start of training when there are many errors this is rarely the case. Moreover, on difficult datasets (as in our case), many correct labels will never have a rank close to one. Hence, our method results in the huge speedups observed later in experiments (see Figure 2).

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. The mapping matrices V and W are initialized at random with mean 0, standard deviation $\frac{1}{\sqrt{d}}$, which is a common choice, e.g. as implemented in the Torch Machine Learning library[2] (which is the software we used for our experiments). Note, the initial weights are rescaled if they violate the constraints (2)-(3). Pseudocode for training with WARP loss is given in Algorithm 1. We use a fixed learning rate γ, chosen using a validation set (a decaying schedule over time t is also possible, but we did not implement that approach). The validation error in the last line of Algorithm 1 is in practice only evaluated after every hour on a subset of the validation set for computational efficiency.

[2] `http://torch5.sourceforge.net/`

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 between ten or a hundred thousand annotations, some of which can be semantically rather 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 $y^r = (y_1^r, \ldots, y_Y^r)$, $p_{sib}@k$ 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 semantic structure by considering the similarity of all the first k predicted labels.

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. isa("toad","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-data we have obtained a similar but noisier proprietary set based on occurences 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, this research mostly 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 [8] and Pascal-VOC [9] have a limited

number of categories, ranging from 20 to 256. More recently, projects such as the TinyImage database [1] and ImageNet [2] 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. [1, 10]. 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 [11]).

An alternative approach, championed by [3, 12, 4, 13], and others, 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 [14] but has been shown to perform worse than (non-embedding based) supervised ranking models like Pamir [15]. Embedding for image retrieval (rather than annotation) using KCCA was also explored in [16]. Other related work includes learning embeddings for supervised document ranking [17] and for semi-supervised multi-task learning [18, 19].

Several loss functions have also recently been proposed to optimize the top of the ranked list. The most similar to our work is the so-called OWPC loss of [5], which is similar to Eq. (6) except that the weight given to each pair of labels (y, i) depends on the rank of the incorrect label i rather than the rank of y. In its original formulation, the algorithm of [5] relies on SVM_{struct} for the optimization, which cannot be used to train our embedding models. Moreover, even if one tries to train our models with SGD on the OWPC loss, each step would necessarily be more costly as it would require additional sampling steps to compute or approximate the rank of the incorrect label. This argument also applies to the loss functions proposed for other algorithms such as ListNet [20] or SVM^{map} [21] because the contribution to these losses of a single annotation is tightly bound to the scores of all other annotations. Thus, 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 widely used to calculate distances between concepts in the field of natural language processing, and has been used for image annotation to build voted classifiers [1, 12]. 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 [2] is a new image dataset organized according to WordNet [22]. 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. The most frequent annotation only appears 0.04% of the time[3].

Web-data 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 appears 0.01% of the time.

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

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

Statistics	ImageNet	Web-data
Number of Training Images	2,518,604	9,861,293
Number of Test Images	839,310	3,286,450
Number of Validation Images	837,612	3,287,280
Number of Labels	15,952	109,444
Median Number of 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 a standard bag-of-visual-terms type representation, which has a sparse vector representation. In particular, we use the bag-of-terms feature setup of [15], which was 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 k-means 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 $d_{\bar{\emptyset}} = 245$ non-zero values. It takes on average 0.5 seconds to extract these features per image (and via sampling the pixels this is invariant to the resolution).

[3] This is a commonly measured sanity check in case there is an annotation that occurs rather often, artificially inflating precision.

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

	ImageNet				Web-data			
Algorithm	p@1	p@10	p_{sib}@10	MAP	p@1	p@10	p_{sib}@10	MAP
Approx. k-NN	1.55%	0.41%	1.69%	2.32%	0.30%	0.34%	5.97%	1.52%
One-vs-Rest	2.27%	1.02%	3.71%	5.17%	0.52%	0.29%	4.61%	1.45%
PAMIR^{IA}	3.14%	1.26%	4.39%	6.43%	0.32%	0.16%	2.94%	0.83%
WSABIE	4.03%	1.48%	5.18%	7.75%	1.03%	0.44%	9.84%	2.27%

6.3 Baselines

We compare our proposed approach to several baselines: approximate k-nearest neighbors (k-NN), one-versus-rest large margin classifiers (One-Vs-Rest) of the form $f_i(x) = w_i^\top x$ trained using the Passive Aggressive algorithm [23], or the same models trained with a ranking loss instead, which we call PAMIR^{IA} as it is like the PAMIR model used in [15] 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 feasible. There are many flavors of approximation (see, e.g [12]). 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-data are summarized in Table 2. Further detailed plots of precision and sibling precision for ImageNet are given in Figure 1. WSABIE outperforms all competing methods. We give a deeper analysis of the results, including time and space requirements in subsequent sections.

Word-Image Embeddings Example word embeddings learnt by WSABIE for Web-data are given in Table 3 and some example image annotations are given in Table 8. Overall, we observe that the embeddings capture the semantic structure of the annotations (and images are also embedded in this space). This explains why the sibling precision of WSABIE is superior to competing methods, which do not attempt to learn the structure between annotations. We also observe that WSABIE is relatively insensitive to the embedding dimension size D as shown in Table 4. Note that although the numerical performance in terms of p@k appears low, this is actually a worst case result as there are many labels that are actually either correct or almost correct, which is not captured by p@k. (However, this is captured to some degree by p_{sib}@k.) Overall, we believe the performance of our method actually gives a practically useful system.

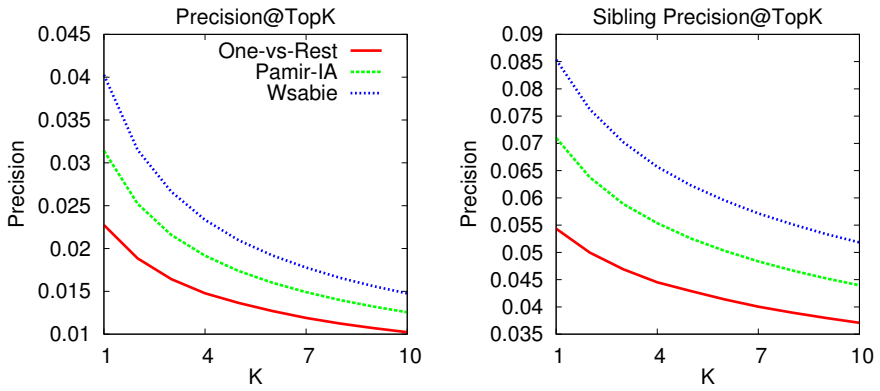

Fig. 1. Precision@k and Sibling Precision@k on ImageNet for various methods.

Table 3. Nearest annotations in the embedding space learnt by WSABIE on Web-data. Translations (e.g. delphin, rosen) and synonyms or misspellings (beckam, mt fuji) have close embeddings. Other annotations are from similar visual images, e.g. Alessandro Del Piero is a soccer player. Annotations in *blue+italics* are in our known sibling set.

Annotation	**Neighboring Annotations**
barack obama	*barak obama*, *obama*, barack, barrack obama, bow wow, george bush
david beckham	*beckham*, david beckam, *alessandro del piero*, *del piero*, david becham
santa	*santa claus*, papa noel, pere noel, *santa clause*, joyeux noel, tomte
dolphin	delphin, dauphin, *whale*, delfin, delfini, baleine, *blue whale*, walvis
cows	*cattle*, shire, *dairy cows*, kuh, *horse*, *cow*, *shire horse*, kone, *holstein*
rose	rosen, *hibiscus*, rose flower, *rosa*, roze, pink rose, *red rose*, *a rose*
pine tree	*abies alba*, *abies*, *araucaria*, *pine*, neem tree, *oak tree*, *pinus sylvestris*
mount fuji	mt fuji, fuji, fujisan, fujiyama, *mountain*, zugspitze, fuji mountain
eiffel tower	*eiffel*, *tour eiffel*, la tour eiffel, *big ben*, paris, *blue mosque*, eifel tower
ipod	i pod, *ipod nano*, *apple ipod*, ipod apple, new ipod, *ipod shuffle*
f18	f 18, eurofighter, f14, fighter jet, tomcat, mig 21, f 16, eurofighter

WARP Loss We compared different models trained with either WARP or AUC optimization (via the margin ranking loss $|1 - f_y(x) + f_{\bar{x}}(y)|_+$ as is used in PAMIR [15]). The results given in Table 5 show WARP consistently gives superior performance. We also compared training time using WARP (with eq. (1), $D = 100$), AUC or a standard implementation of SGD for (4) where the rank (7) is computed explicitly rather than using our approximation method (note, in that case updates can be made for all violations for a given image at once) which we call OWPC-SGD. For all methods we report results using their best learning rate γ as measured on the validation set. 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. Hence, the trick of approximating the rank introduced in Section 3 is very important for our task.

Table 4. Changing the Embedding Size on ImageNet. Test Error metrics when we change the dimension D of the embedding space used in WSABIE.

Embedding Dimension	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%

Table 5. WARP vs. AUC optimization. For each model, WARP consistently improves over AUC.

		ImageNet		Web-data	
Model	Loss	p@1	p@10	p@1	p@10
$f_i(x) = \Phi_W(i)^\top \Phi_I(x)$	AUC	1.65%	0.91%	0.19%	0.13%
$f_i(x) = \Phi_W(i)^\top \Phi_I(x)$	WARP	**4.03%**	**1.48%**	**1.03%**	**0.44%**
$f_i(x) = w_i^\top x$	AUC	3.14%	1.26%	0.32%	0.16%
$f_i(x) = w_i^\top x$	WARP	**4.25%**	**1.48%**	**0.94%**	**0.39%**

Computational Expense A summary of the test time and space complexity of the various algorithms we compare is given in Table 6 (not including cost of pre-processing of features) as well as concrete numbers on the particular datasets we use using a single computer, and assuming the data fits in memory (for WSABIE we give values for $D = 100$). In particular k-NN would take 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 per image and requires 1.2GB and 8.2GB respectively. WSABIE takes 0.02 and 0.17 seconds, and requires far less memory, only 12MB and 82MB respectively. In summary, WSABIE can be feasibly run on a laptop using limited resources whereas k-NN requires all the resources of an entire cluster. Moreover as k-NN has time and space complexity $\mathcal{O}(n \cdot d_{\bar{\emptyset}})$, where n is the number of training examples and $d_{\bar{\emptyset}}$ 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 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.

Approximate Nearest Neighbor and Nearest Neighbor Approximate nearest neighbor (ANN) performed rather poorly, and computing true nearest neighbor is infeasible for our problems due to prohibitive computational expense and use of memory. ANN, however, can be tuned depending on the test time vs accuracy tradeoff one wishes to achieve. We investigated some parameter choices for ImageNet. In our implementation in the experiments reported in Table 2 we get 1.55% precision at 1 when we traverse a tree until we have ≈20,000 points at the leaf. Note, this takes around two days to compute which should be compared

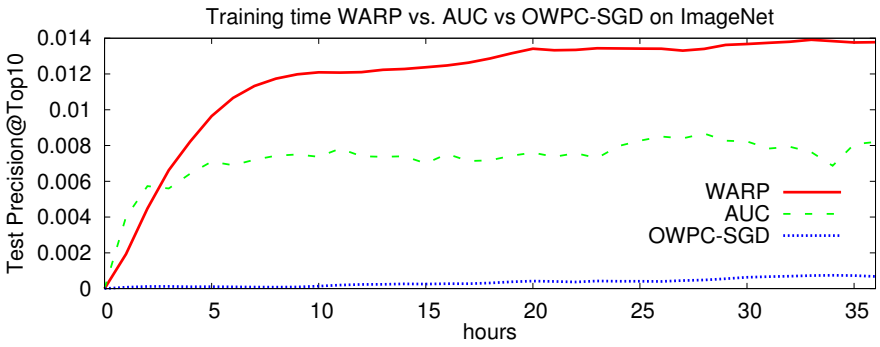

Fig. 2. Training time: WARP vs. OWPC-SGD & AUC.

Table 6. 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. Prediction times (d=days, h=hours) and memory requirements for the whole test sets are given for both ImageNet and Web-data datasets. We denote by Y the number of classes, n the number of training examples, d the image input dimension, $d_{\bar{\emptyset}}$ 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	Test Time and Memory Usage			
			ImageNet		Web-data	
			Time	Space	Time	Space
k-NN	$\mathcal{O}(n \cdot d_{\bar{\emptyset}})$	$\mathcal{O}(n \cdot d_{\bar{\emptyset}})$	255 d	6.9 GB	3913 d	27 GB
Approx. k-NN	$\mathcal{O}((p + n/2^p) \cdot d_{\bar{\emptyset}})$	$\mathcal{O}(n \cdot d)$	*variable*	7 GB	*variable*	27 GB
One-vs-Rest	$\mathcal{O}(Y \cdot d_{\bar{\emptyset}})$	$\mathcal{O}(Y \cdot d)$	17 h	1.2 GB	19 d	8.2 GB
PAMIR^{IA}	$\mathcal{O}(Y \cdot d_{\bar{\emptyset}})$	$\mathcal{O}(Y \cdot d)$	17 h	1.2 GB	19 d	8.2 GB
WSABIE	$\mathcal{O}((Y + d_{\bar{\emptyset}}) \cdot D)$	$\mathcal{O}((Y + d) \cdot D)$	5.6 h	12 MB	6.5 d	82 MB

to other methods in Table 6. If we have ≈2,500 points at each leaf (so it is much faster) this goes down to 0.89%. If we have ≈40,000 points at each leaf, we still have only 1.94%. In summary, we cannot get ANN to perform well on our large scale datasets.

Despite exact nearest neighbor being prohibitively expensive, we used a very large cluster of machines to compute it on our test set to evaluate its accuracy. For ImageNet we get 4.5% p@1 and for Web-data we get 0.3% p@1 (the latter is quoted in Table 2). For Web-data this is still poor, but for ImageNet this result is good (the difference might be due to Web-data being noisier and having less points per class). It can thus be understood that ANN is hard to get to work well for ImageNet via the following analysis: if we retrieve the first nearest neighbor correctly only 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%. Retrieving the first neighbor, at least on this dataset, is very important for achieving good

accuracy, a phenomena that has been observed on other datasets, see e.g. Table 3 of [3]. 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 [12].

Ensemble Models Ensembles of models are known to provide a performance boost [24], so we linearly combined various pre-trained WSABIE annotation models with different embedding sizes. We estimated the weights of the linear combination using the validation set in order to minimize the overall cost. Table 7 shows a summary of the results. In fact, ensembles of our model do give an impressive boost in performance. Combining 100, 200 and 300 dimensional models provides the best overall performance, while still yielding a scalable solution, both in terms of memory and time.

Table 7. Ensemble models combining different annotation approaches, on the ImageNet data. WSABIE-300 means it was trained with an embedding of size 300. Hyperparameters λ_1, λ_2, λ_3 are chosen on the validation set using grid search.

Ensemble Model	p@1	p@10	p_{sib}@10	MAP
WSABIE-300	4.03%	1.48%	5.18%	7.75%
λ_1WSABIE-100 + λ_2WSABIE-200	5.74%	1.97%	6.29%	10.17%
λ_1WSABIE-100 + λ_2WSABIE-300	5.38%	1.99%	6.27%	10.36%
λ_1WSABIE-100 + λ_2WSABIE-200 + λ_3WSABIE-300	6.14%	2.09%	6.42%	11.23%

7 Conclusions

We have introduced a scalable model for image annotation based upon learning a joint representation of images and annotations that optimizes top-of-the-list ranking measures.

To our knowledge this is the first data analysis of image annotation (considering all classes at once, with millions of data points) at such a scale. We have shown that our embedding model trained with the WARP loss is faster, takes less memory, and yields better performance than any known competing approach for this task. Stochastic gradient descent is the standard method for optimizing non-convex models such as our embedding model, but SGD applied to the loss function of interest is too slow, and the novelty of our training algorithm is to use an approximation of the rank, which is otherwise slow to compute, via a sampling trick that makes such optimization feasible for the first time. In fact, to the best of our knowledge this is the largest scale optimization of a rank-dependent measure attempting to maximize the precision at the top positions of the ranking reported on any dataset (not just for image annotation).

Finally, evaluation using the novel sibling precision metric shows that our embedding method learns more of the semantic structure of the annotation space

Table 8. Examples of the top 10 annotations of three compared approaches: PAMIR^{IA} , One-vs-Rest and WSABIE, on the Web-data 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
	bora, free willy, su, orka, worldwide, sunshine coast, bequia, tioman island, universal remote montagna, esperar, *bottlenose dolphin*	surf, bora, belize, sea world, balena, wale, tahiti, delfini, surfing, *mahi mahi*	delfini, *orca*, **dolphin**, mar, delfin, dauphin, *whale*, cancun, *killer whale*, sea world
	air show, st augustine, stade, concrete architecture, streetlight, doha qatar, skydiver, *tokyo tower*, sierra sinn, lazaro cardenas	**eiffel tower**, *tour eiffel*, snowboard, blue sky, *empire state building*, luxor, *eiffel*, *lighthouse*, jump, adventure	**eiffel tower**, *statue*, *eiffel*, mole antoneliana, la tour eiffel, londra, cctv tower, *big ben*, calatrava, *tokyo tower*
	chris hanson, michael johns, ryan guettler, richard marx, depardieu, barack hussein obama, freddie vs jason, dragana, shocking, falco	falco, barack, daniel craig, **obama**, *barack obama*, kanye west, pharrell williams, 50 cent, barrack obama, bono	barrack obama, *barack obama*, barack hussein obama, *barack obama*, james marsden, *jay z*, **obama**, nelly, falco, barack

than competing methods, which we believe helps lead to its good performance in other metrics.

Acknowledgements

We especially thank Nemanja Petrovic for helping us with the nearest neighbor experiments.

References

1. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: a large dataset for non-parametric object and scene recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence **30** (2008) 1958–1970
2. Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: ImageNet: A Large-Scale Hierarchical Image Database. In: IEEE Conference on Computer Vision and Pattern Recognition. (2009)
3. Makadia, A., Pavlovic, V., Kumar, S.: A new baseline for image annotation. In: European conference on Computer Vision (ECCV). (2008)
4. Guillaumin, M., Mensink, T., Verbeek, J., Schmid, C., LEAR, I., Kuntzmann, L.: Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In: ICCV. (2009)

5. Usunier, N., Buffoni, D., Gallinari, P.: Ranking with ordered weighted pairwise classification. In Bottou, L., Littman, M., eds.: Proceedings of the 26th International Conference on Machine Learning, Montreal, Omnipress (June 2009) 1057–1064
6. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) **58**(1) (1996) 267–288
7. Robbins, H., Monro, S.: A stochastic approximation method. Annals of Mathematical Statistics **22** (1951) 400–407
8. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology (2007)
9. Everingham, M., Gool, L.V., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2007 (VOC2007) (2007)
10. Fergus, R., Weiss, Y., Torralba, A.: Semi-supervised learning in gigantic image collections. In: Advances in Neural Information Processing Systems, 2009. (2009)
11. Grauman, K., Darrell, T.: The pyramid match kernel: Efficient learning with sets of features. Journal of Machine Learning Research **8**(725-760) (2007) 7–8
12. Torralba, A.B., Fergus, R., Weiss, Y.: Small codes and large image databases for recognition. In: CVPR, IEEE Computer Society (2008)
13. Wang, J., Li, J., Wiederholdy, G.: SIMPLIcity: Semantics-sensitive integrated matching for picture libraries. Advances in Visual Information Systems 171–193
14. Monay, F., Gatica-Perez, D.: PLSA-based image auto-annotation: constraining the latent space. In: Proceedings of the 12th annual ACM international conference on Multimedia, ACM New York, NY, USA (2004) 348–351
15. Grangier, D., Bengio, S.: A discriminative kernel-based model to rank images from text queries. Transactions on Pattern Analysis and Machine Intelligence **30**(8) (2008) 1371–1384
16. Zhou, Z., Zhan, D., Yang, Q.: Semi-supervised learning with very few labeled training examples. In: Proceedings of the National Conference on Artificial Intelligence. Volume 22., AAAI Press; MIT Press; 1999 (2007) 675
17. Bai, B., Weston, J., Grangier, D., Collobert, R., Sadamasa, K., Qi, Y., Cortes, C., Mohri, M.: Polynomial semantic indexing. In: Advances in Neural Information Processing Systems (NIPS 2009). (2009)
18. Ando, R.K., Zhang, T.: A framework for learning predictive structures from multiple tasks and unlabeled data. JMLR **6** (11 2005) 1817–1953
19. Loeff, N., Farhadi, A., Endres, I., Forsyth, D.: Unlabeled Data Improves Word Prediction. ICCV '09 (2009)
20. Xia, F., Liu, T.Y., Wang, J., Zhang, W., Li, H.: Listwise approach to learning to rank: theory and algorithm. In: Proceedings of the 25th International Conference on Machine Learning. (2008)
21. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval. (2007) 271–278
22. Fellbaum, C., ed.: WordNet: An Electronic Lexical Database. MIT Press (1998)
23. Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., Singer, Y.: Online passive-aggressive algorithms. Journal of Machine Learning Research **7** (2006) 551–585
24. Wolpert, D.: Stacked generalization. Neural Networks **5** (1992) 241–259