WSABIE: Scaling Up To Large Vocabulary Image Annotation

Jason Weston[1] and **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 the top 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, called WSABIE, both outperforms several baseline methods and is faster and consumes less memory.

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 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. [Makadia *et al.*, 2008], 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 (p@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 [Usunier *et al.*, 2009] 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.

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 and fast computation time;

(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 use. Section 3 defines the WARP loss and how to train our models with it. Section 4 details prior work, Section 5 describes experiments on large scale datasets, and Section 6 concludes. Note that parts of this paper were published previously in [Weston *et al.*, 2010].

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 to rank the possible annotations of a given image such that the highest ranked ones 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 [Tibshirani, 1996]. 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 [Usunier *et al.*, 2009] as:

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

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. 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 [Usunier *et al.*, 2009] 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:

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

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

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

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).

[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.

Algorithm 1 Online WARP Loss Optimization

Input: labeled data $(x_i, y_i), y_i \in \{1, \ldots, Y\}$.
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 $\bar{y} \in \{1, \ldots, Y\} \setminus y_i$.
    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.

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

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

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.

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.

4 Related Approaches

The problem of image annotation, including the related task of image classification, has been the subject of much research 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 [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/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 (e.g. [Grauman and Darrell, 2007a]).

An alternative approach, championed by [Makadia *et al.*, 2008; Torralba *et al.*, 2008b], and others, is to use k-nearest neighbors 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 and Gatica-Perez, 2004] but has been shown to perform worse than (non-embedding based) supervised ranking models like PAMIR [Grangier and Bengio, 2008]. Embedding for image retrieval (rather than annotation) using KCCA was also explored in [Zhou *et al.*, 2007].

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 [Usunier *et al.*, 2009], which is similar to Eq. (6) except that the weight given to

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

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 [Usunier *et al.*, 2009] 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 [Xia *et al.*, 2008] or SVM^{map} [Yue *et al.*, 2007] 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.

5 Experiments

5.1 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. 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.**

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

5.2 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 [Grangier and Bengio, 2008], 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 $d_{\bar{\varnothing}} = 245$ 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 are given.

	ImageNet		Web-data	
Algorithm	p@1	p@10	p@1	p@10
Approx. k-NN	1.55%	0.41%	0.30%	0.34%
One-vs-Rest	2.27%	1.02%	0.52%	0.29%
PAMIR^{IA}	3.14%	1.26%	0.32%	0.16%
WSABIE	4.03%	1.48%	1.03%	0.44%

5.3 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 [Crammer *et al.*, 2006], or the same models trained with a ranking loss instead, which we call PAMIR^{IA} as it is like the PAMIR model used in [Grangier and 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 feasible. 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.

5.4 Results

The results of comparing all methods on ImageNet and Web-data are summarized in Table 2. WSABIE outperforms all competing methods. We give a deeper analysis of the results, including time/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).

Table 3: Nearest annotations in the embedding space learnt by WSABIE on Web-data. Translations (e.g. delphin) and synonyms or misspellings (beckam, mt fuji) have close embeddings.

Annotation	**Neighboring Annotations**
barack obama	barak obama, obama, barack, barrack obama
david beckham	beckham, david beckam, alessandro del piero
santa	santa claus, papa noel, pere noel, santa clause
dolphin	delphin, dauphin, whale, delfin, delfini, baleine
cows	cattle, shire, dairy cows, kuh, horse, cow
rose	rosen, hibiscus, rose flower, rosa, roze
pine tree	abies alba, abies, araucaria, pine, neem tree
mount fuji	mt fuji, fuji, fujisan, fujiyama, mountain
eiffel tower	eiffel, tour eiffel, la tour eiffel, big ben, paris
ipod	i pod, ipod nano, apple ipod, ipod apple
f18	f 18, eurofighter, f14, fighter jet, tomcat, mig 21

Table 4: **WARP vs. AUC optimization.** For each model, WARP consistently improves over AUC.

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

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 [Grangier and Bengio, 2008]). The results given in Table 4 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 1 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.

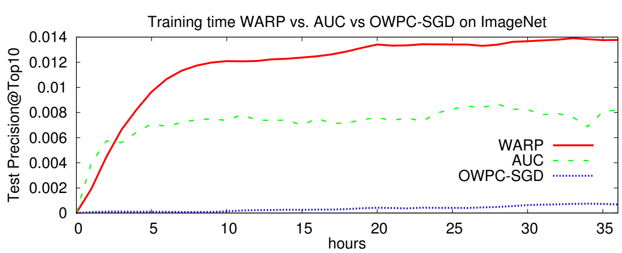

Figure 1: Training time: WARP vs. OWPC-SGD & AUC.

Table 5: **Algorithm 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 the Web-data dataset. We denote by Y the number of classes, n the number of training examples, d the image input dimension, $d_{\bar{\varnothing}}$ 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	Test Time and Memory Usage for Web-data	
		Time	Space
k-NN	$\mathcal{O}(n \cdot d_{\bar{\varnothing}})$	3913 d	27 GB
Approx. k-NN	$\mathcal{O}((p + n/2^p) \cdot d_{\bar{\varnothing}})$	*variable*	27 GB
One-vs-Rest	$\mathcal{O}(Y \cdot d_{\bar{\varnothing}})$	19 d	8.2 GB
PAMIR^{IA}	$\mathcal{O}(Y \cdot d_{\bar{\varnothing}})$	19 d	8.2 GB
WSABIE	$\mathcal{O}((Y + d_{\bar{\varnothing}}) \cdot D)$	6.5 d	82 MB

Computational Expense

A summary of the test time and space complexity of the various algorithms we compare is given in Table 5 (not including cost of pre-processing of features) as well as concrete numbers on the Web-data 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 3913 days to compute the test error on the Web data, corresponding to 103 seconds per image, making it infeasible to use. In comparison, PAMIR^{IA} takes 0.5 seconds to compute per image and requires 8.2GB. WSABIE takes 0.17 seconds, and requires far less memory, only 82MB. 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 $\mathcal{O}(n \cdot d_{\bar{\varnothing}})$, where n is the number of training examples and $d_{\bar{\varnothing}}$ 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.

Ensemble Models

Ensembles of models are known to provide a performance boost [Wolpert, 1992], 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 6 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 6: Ensemble models combining different annotation approaches on ImageNet. W-300 means WSABIE 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
WSABIE-300	4.03%	1.48%
λ_1W-100 + λ_2W-200	5.74%	1.97%
λ_1W-100 + λ_2W-300	5.38%	1.99%
λ_1W-100 + λ_2W-200 + λ_3W-300	6.14%	2.09%

Ensembles of Features and Models

In this paper we have concentrated on learning algorithms, not feature representations, but of course good feature representations can improve results a lot. So far we have only used one type of feature representation, a bag of visual terms model, but many other types of feature representation appear in the literature. Here, we explore the possibility that an ensemble of feature representations can improve performance as has been shown before [Makadia *et al.*, 2008]. We thus combined multiple feature representations which are the concatenation of various spatial [Grauman and Darrell, 2007b] and multiscale color and texton histograms [Leung and Malik, 1999] for a total of about 5×10^5 dimensions. The descriptors are somewhat sparse, with about 50000 non-zero weights per image. Some of the constituent histograms are normalized and some are not. We then perform Kernel PCA [Schoelkopf *et al.*, 1999] on the combined feature representation using the intersection kernel [Barla *et al.*, 2003] to produce a 1024 dimensional input vector for training WSABIE. We then train WSABIE as before. We consider both a single WSABIE model and an ensemble of trained models as in the previous section, this time with either 3 or 10 models all of dimension 100. We also compare to exact nearest-neighbor using the KPCA features as the representation for finding neighbors, so we have a comparable input to WSABIE. The results are given in Table 7. Our results show an ensemble of features and of trained models gives our best performance of around 10% p@1.

Table 7: Ensembles of features and models on ImageNet. These results should be compared to single models or ensembles of models using only a single feature representation which are given in Table 6. We also include the result of exact nearest neighbor, computed over the entire test set using the KPCA representation of the ensemble of features.

Algorithm	p@1	p@10
Exact Nearest Neighbor	7.73%	-
WSABIE KPCA features	8.83%	2.71%
WSABIE KPCA (Ensemble 3 models)	9.82%	2.88%
WSABIE KPCA (Ensemble 10 models)	10.03%	3.02%

Note that although the numerical performance in terms of p@k still might appear relatively low (the correct annotation is top-ranked "only" 10% of the time), this is actually much better than it seems for several reasons. Firstly, in the ImageNet task there are ∼16,000 possible labels so it is of course hard to get the right label top-ranked. Secondly, the p@k metric is in fact a worst case result. The reason for this is that there are many labels that are actually either correct or almost correct, which is not captured by this metric. This is especially true in the Web-data where annotations include synonyms or near-synonyms such as "barack obama" and "obama" (see Table 3). Similarly, predicting "toad" instead of "frog" is simply labeled as wrong in our system, which does not capture the semantics that the system got it *almost* right. To this end in another work we have developed a metric called sibling-precision which attempts to capture these semantics, which is detailed in [Weston *et al.*, 2010], but we do not have space to describe it here. Overall, we believe the performance of our method actually gives a practically useful system.

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.

Image	One-vs-Rest	WSABIE
	surf, bora, belize, sea world, balena, wale, tahiti, delfini, surfing, mahi mahi	delfini, orca, **dolphin**, mar, delfin, dauphin, whale, cancun, killer whale, sea world
	**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
	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

6 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).

Acknowledgements

We thank N. Petrovic for his help with the k-NN experiments.

References

[Barla *et al.*, 2003] A. Barla, F. Odone, and A. Verri. Histogram intersection kernel for image classification. *International Conference on Image Processing (ICIP)*, 3:III–513–16 vol.2, 2003.

[Crammer *et al.*, 2006] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585, 2006.

[Deng *et al.*, 2009] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[Everingham *et al.*, 2007] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2007 (VOC2007), 2007.

[Fellbaum, 1998] Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.

[Fergus *et al.*, 2009] R. Fergus, Y. Weiss, and A. Torralba. Semi-supervised learning in gigantic image collections. In *Advances in Neural Information Processing Systems, 2009*, 2009.

[Grangier and Bengio, 2008] David Grangier and Samy Bengio. A discriminative kernel-based model to rank images from text queries. *Transactions on Pattern Analysis and Machine Intelligence*, 30(8):1371–1384, 2008.

[Grauman and Darrell, 2007a] K. Grauman and T. Darrell. The pyramid match kernel: Efficient learning with sets of features. *Journal of Machine Learning Research*, 8(725-760):7–8, 2007.

[Grauman and Darrell, 2007b] Kristen Grauman and Trevor Darrell. The Pyramid Match Kernel: Efficient Learning with Sets of Features. *Journal of Machine Learning Research*, 8:725–760, April 2007.

[Griffin *et al.*, 2007] G. Griffin, A. Holub, and P. Perona. Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology, 2007.

[Leung and Malik, 1999] T. Leung and J. Malik. Recognizing surface using three-dimensional textons. *Proc. of 7th Int'l Conf. on Computer Vision, Corfu, Greece*, 1999.

[Makadia *et al.*, 2008] A. Makadia, V. Pavlovic, and S. Kumar. A new baseline for image annotation. In *European conference on Computer Vision (ECCV)*, 2008.

[Monay and Gatica-Perez, 2004] F. Monay and D. Gatica-Perez. PLSA-based image auto-annotation: constraining the latent space. In *Proceedings of the 12th annual ACM international conference on Multimedia*, pages 348–351. ACM New York, NY, USA, 2004.

[Robbins and Monro, 1951] Herbert Robbins and Sutton Monro. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407, 1951.

[Schoelkopf *et al.*, 1999] Bernhard Schoelkopf, Alexander J. Smola, and Klaus R. Müller. Kernel principal component analysis. *Advances in kernel methods: support vector learning*, pages 327–352, 1999.

[Tibshirani, 1996] R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[Torralba *et al.*, 2008a] A. Torralba, R. Fergus, and W. T. Freeman. 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, 2008.

[Torralba *et al.*, 2008b] Antonio B. Torralba, Robert Fergus, and Yair Weiss. Small codes and large image databases for recognition. In *CVPR*. IEEE Computer Society, 2008.

[Usunier *et al.*, 2009] Nicolas Usunier, David Buffoni, and Patrick Gallinari. Ranking with ordered weighted pairwise classification. In Léon Bottou and Michael Littman, editors, *Proceedings of the 26th International Conference on Machine Learning*, pages 1057–1064, Montreal, June 2009. Omnipress.

[Weston *et al.*, 2010] Jason Weston, Samy Bengio, and Nicolas Usunier. Large scale image annotation: learning to rank with joint word-image embeddings. *Machine Learning*, 81(1):21–35, 2010.

[Wolpert, 1992] D. Wolpert. Stacked generalization. *Neural Networks*, 5:241–259, 1992.

[Xia *et al.*, 2008] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the 25th International Conference on Machine Learning*, 2008.

[Yue *et al.*, 2007] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 271–278, 2007.

[Zhou *et al.*, 2007] Z.H. Zhou, D.C. Zhan, and Q. Yang. Semi-supervised learning with very few labeled training examples. In *Proceedings of the National Conference on Artificial Intelligence*, volume 22, page 675. AAAI Press; MIT Press; 1999, 2007.