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

Jason Weston*, Samy Bengio* and Nicolas Usunier†

* Google, USA † Universite de Paris, LIP6, France

BTAS, September 2010

Image Annotation: What is it?

Goal: Predict text given an image: 100,000s+ of possible annotations.

→ **obama**

→ **eiffel tower**

What do we need?

Scalable+good features → we have not focused on this, use *bag-of-terms*
Scalable(memory,speed)+good classifier → One-Vs-Rest, PAMIR, k-NN?

In this work we propose a scalable+good classifier.

Datasets (to grasp the scale)

Statistics	ImageNet	Web
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

Our Proposed Solution

Model choice: Jointly embed annotations and images.

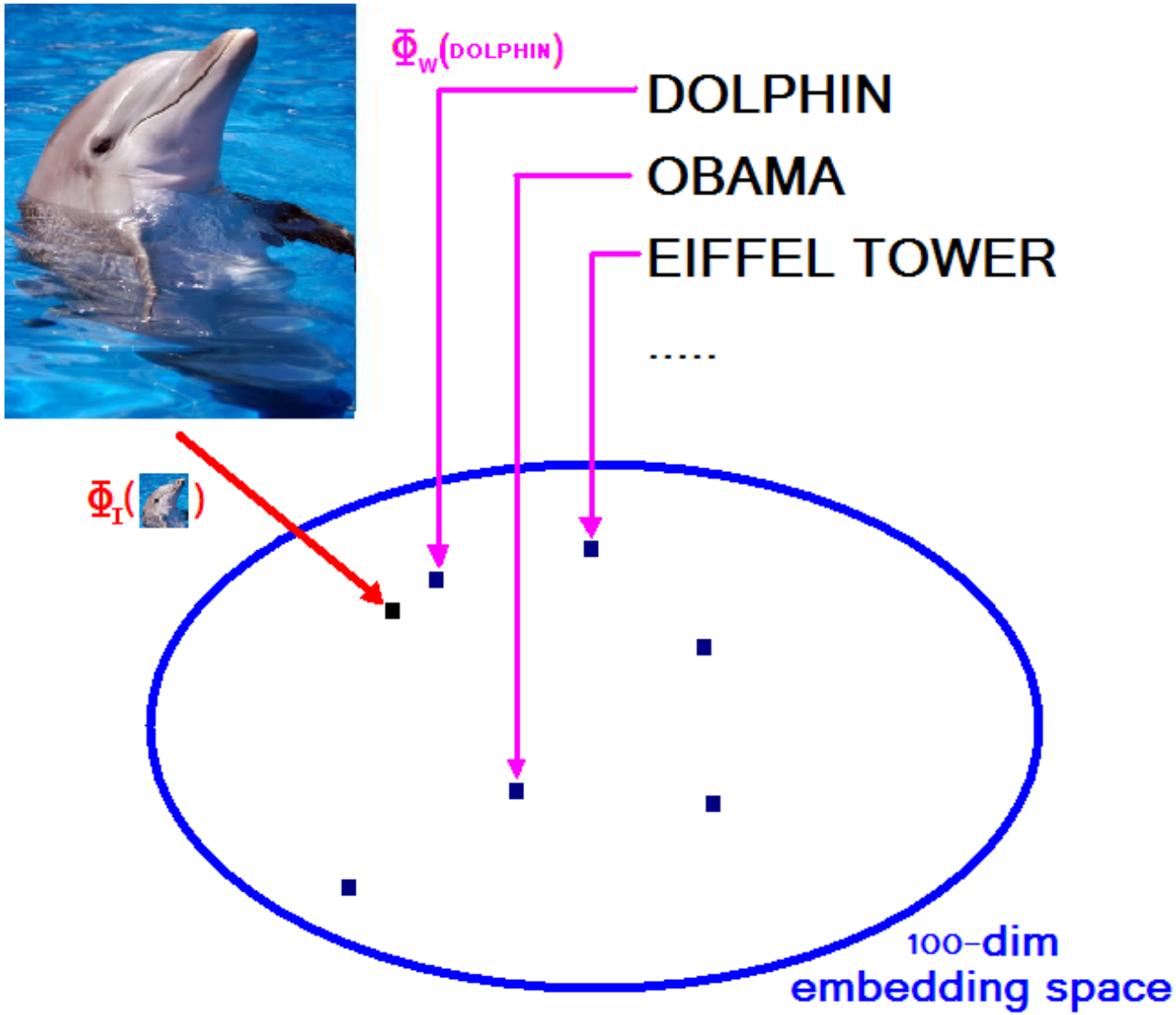
- Learns structure by multi-tasking (One-Vs-Rest doesn't)
- Small memory usage (82MB vs. 8.2GB for One-Vs-Rest)
- Fast Test Time (0.17s vs 0.5s for One-vs-Rest)

(not including feature extraction time: 0.4s)

New loss function: WARP Loss.

- Directly optimizes precision@k.
- Efficient to train on millions of examples.

Our Proposed Solution



Learn $\Phi_I(\cdot)$ and $\Phi_W(\cdot)$ to optimize precision@k.

Joint Word-Image Embedding Model

Images: $d = 10,000$ dimensional sparse "visterms". Learn map:

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

Annotations: Y possible annotations, indexed by i. Learn map:

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

Our model compares the degree of match between the image and annotations in the embedding space:

$$f_i(x) = sim(\Phi_W(i), \Phi_I(x)) = W_i^\top Vx$$

We also constrain the weights (regularize):

$$||V_i||_2 \leq C, \quad i = 1, \ldots, d, \qquad ||W_i||_2 \leq C, \quad i = 1, \ldots, Y.$$

Ranking Annotations: AUC is Suboptimal

Classical approach to learning to rank is maximize AUC by minimizing:

$$\sum_x \sum_y \sum_{\bar{y} \neq y} |1 + f_{\bar{y}}(x) - f_y(x)|_+$$

A scalable version of this is via stochastic gradient descent (SGD): sample triplets $(x, y, \bar{y})$ and make a gradient step on the hinge loss.

Problem: All pairwise errors are considered the same.

Example:

Function 1: true annotations ranked 1st and 101st.

Function 2: true annotations ranked 50st and 52st.

AUC prefers these *equally* as both have 100 "violations".

We want to optimize the top of the ranked list!

Ordered Weighted Pairwise Classification (OWPC) Loss

A class of ranking error functions recently defined in [Usunier et al. '09]:

$$err(f(x), y) = L(rank_y(f(x))),$$

where

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

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

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

Different choices of $L(\cdot)$ have different minimizers:

- $\alpha_j = \frac{1}{Y-1} \rightarrow$ minimize mean rank
- $\alpha_j = \frac{1}{j} \rightarrow$ more weight on optimizing the top of list.
  - Example from before: $\alpha_j = \frac{1}{j} \rightarrow$ err(func1)=5.18, err(func2)=8.99.

SVM_{struct} with OWPC = State-of-art on small text retrieval tasks.

Weighted Approximate-Rank Pairwise (WARP) Loss

Problem: we would like to apply SGD:

$$err(f(x), y) = L(rank_y^1(f(x))), \qquad rank_y^1(f(x)) = \sum_{\bar{y} \neq y} I(f_{\bar{y}}(x) + 1 \geq f_y(x))$$

. . . but this is expensive to compute per (x, y) sample when Y is large.

Solution: approximate by sampling $f_i(x)$ until we find a violating label $\bar{y}$

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

where N is the number of trials in the sampling step.

Online WARP Loss

Input: labeled data (x_i, y_i), $y_i \in \{1, \ldots, Y\}$.
repeat
  Pick a random labeled example (x_i, y_i)
  Set $N = 0$.
  repeat
    Pick a random annotation $\bar{y} \in \{1, \ldots, Y\} \setminus y_i$.
    $N = N + 1$.
  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:
$$L(\lfloor \tfrac{Y-1}{N} \rfloor)|1 - f_y(x) + f_{\bar{y}}(x)|_+$$
  end if
until validation error does not improve.

Evaluation: Sibling Precision

We measure the standard measures precision@k and MAP.
Two labels can be synonyms, translations or at least similar (toad instead of frog). We want to give credit for predicting similar labels.
To evaluate this we measure for a ranking $y^r = (y_1^r, \ldots, y_Y^r)$:

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

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

When S is the identity we recover the usual $p@k$ loss.
For ImageNet we have isa relations annotated in WordNet.
For Web we use a set collected via "X is a Y" patterns on web pages.
Median number of siblings: ImageNet = 12, Web = 143.

Other Approaches

Methods Compared:

- One-Vs-Rest: $f_i(x) = w_i \cdot x$ - trained with Hinge loss.
- PAMIR^{IA}: $f_i(x) = w_i \cdot x$ - trained with AUC [Grangier & Bengio, '08].
- Approximate k-NN - speed/accuracy trade-off:
  we tried: bal. tree of depth p, calc distance of all $\frac{n}{2^p}$ points.

Other Related Work

- Unsupervised text embedding, e.g. LSI, pLSI, LDA, etc.
- Supervised text embedding: e.g. [Bai et al. '09]
- Optimizing Precision@k/MAP for text: e.g. ListNet [Cao et al. '07], SVM$_{map}$ [Yu et al., '07], LambdaRank [Burges et al., '07] and more.

Test Set Performance Results

On ImageNet

Algorithm	p@1	p@10	p_{sib}@10	MAP
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	4.03%	1.48%	5.18%	7.75%

On Web Images

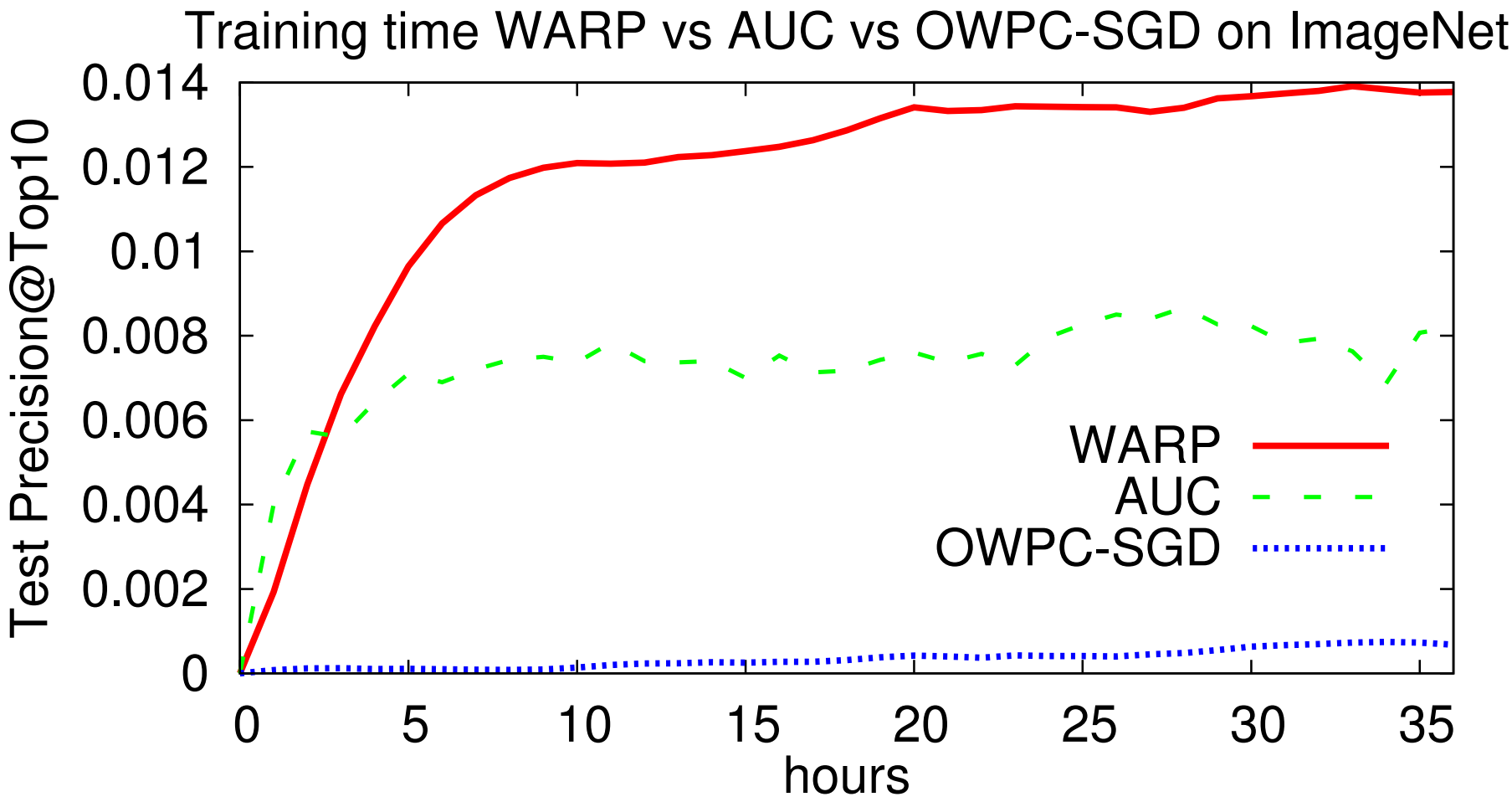
Algorithm	p@1	p@10	p_{sib}@10	MAP
Approx. k-NN	0.30%	0.34%	5.97%	1.52%
One-vs-Rest	0.52%	0.29%	4.61%	1.45%
PAMIR^{IA}	0.32%	0.16%	2.94%	0.83%
WSABIE	1.03%	0.44%	9.84%	2.27%

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%
	WARP	**4.03%**	**1.48%**
$f_i(x) = w_i \cdot x$	AUC	3.14%	1.26%
	WARP	**4.25%**	**1.48%**
Dataset: Web			
$f_i(x) = s(\Phi_W(i), \Phi_I(x))$	AUC	0.19%	0.13%
	WARP	**1.03%**	**0.44%**
$f_i(x) = w_i \cdot x$	AUC	0.32%	0.16%
	WARP	**0.94%**	**0.39%**

Training time: WARP vs. OWPC-SGD & AUC



Test Time and Memory Constraints

Test Time and Memory requirement needed to return the top ranked annotation on the test set of Imagenet and Web, not including feature generation.

Algorithm	ImageNet		Web	
	Time	Space	Time	Space
k-NN	255 days (26.2s)	6.9 GB	3913 days (103s)	27.1 GB
Approx. kNN	2 days	7 GB	-	-
One-vs-Rest	17 h (0.07s)	1.2 GB	19 days (0.5s)	8.2 GB
PAMIR	17 h	1.2 GB	19 days	8.2 GB
WSABIE	5.6 h (0.02s)	12 MB	6.5 days (0.17s)	82 MB

Changing the Embedding Size on ImageNet

Test error metrics when we change the dimension D of the embedding space used in WSABIE.

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%

Training an Ensemble of WSABIEs

Ensemble learning is known to improve performance.

Several WSABIEs can be trained and combined, giving improved performance, but still give a reasonably low memory usage + fast model.

Model	p@1	p@10	p_{sib}@10	MAP
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	4.03%	1.48%	5.18%	7.75%
WSABIE Ensemble (2 models)	5.74%	1.97%	6.29%	10.17%
WSABIE Ensemble (3 models)	6.14%	2.09%	6.42%	11.23%

Using better features..

This paper is not about feature representations.
But, clearly better features lead to better performance.

ImageNet: bag-of-words features

Algorithm	p@1	p@10	p_{sib}@10	MAP
WSABIE	4.03%	1.48%	5.18%	7.75%
WSABIE Ensemble (3 models)	6.14%	2.09%	6.42%	11.23%

ImageNet: words + position + color features

Algorithm	p@1	p@10	p_{sib}@10	MAP
Exact Nearest Neighbor	7.73%			
WSABIE	8.83%	2.71%	9.48%	14.97%
WSABIE Ensemble (3 models)	9.82%	2.88%	9.91%	16.24%
WSABIE Ensemble (10 models)	10.03%	3.02%	10.4%	17.02%

Learned Annotation Embedding (on Web Data)

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

Image Annotation Examples: Dolphin

WSABIE: delfini, orca, dolphin, mar, delfin, dauphin, whale, cancun, killer whale, sea world

One-Vs-Rest: surf, bora, belize, sea world, balena, wale, tahiti, delfini, surfing, mahi mahi

WSABIE: blue whale, whale shark, great white shark, underwater, white shark, shark, manta ray, dolphin, requin, blue shark, diving

One-Vs-Rest: freediving, blau, deep sea, azul, caretta caretta, manta ray, leopard seal, taucher, dolphin, underwater scene, business background

Image Annotation Examples: Obama & Eiffel Tower

WSABIE: barrack obama, barak obama, barack hussein obama, barack obama, james marsden, jay z, obama, nelly, falco, barack

One-Vs-Rest: falco, barack, daniel craig, obama, barack obama, kanye west, pharrell williams, 50 cent, barrack obama, bono, smoking

WSABIE: eiffel, paris by night, la tour eiffel, tour eiffel, eiffel tower, las vegas strip, eifel, tokyo tower, eifel tower

One-Vs-Rest: tour eiffel, eiffel tower, eiffel, la tour eiffel, paris by night, paris france, advent, paris, warhammer

Image Annotation Examples: Ipod

WSABIE: ipod, ipod nano, nokia, i pod, nintendo ds, nintendo, lg, pc, nokia 7610, vino

One-Vs-Rest: wine, ipod, i pod, zippo, brochure, moleskine, nintendo ds, book, nokia, ipod classic

WSABIE: radioactive, ipod ad, post it, smiley, yellow, smiley face, smile, iowa hawkeyes, a style, caution, soda stereo, kill bill, idance

One-Vs-Rest: pacman, pac man, a style, amarillo, smiley face, smile, enjoi, gelb, radioactive, be happy, yellow caution, soda stereo

Conclusion and Future Work

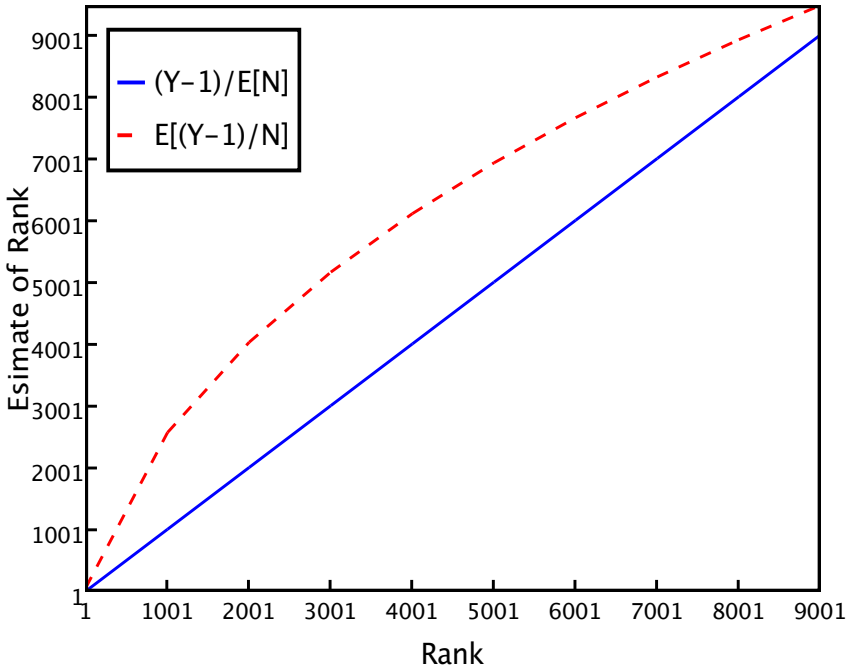
Conclusion

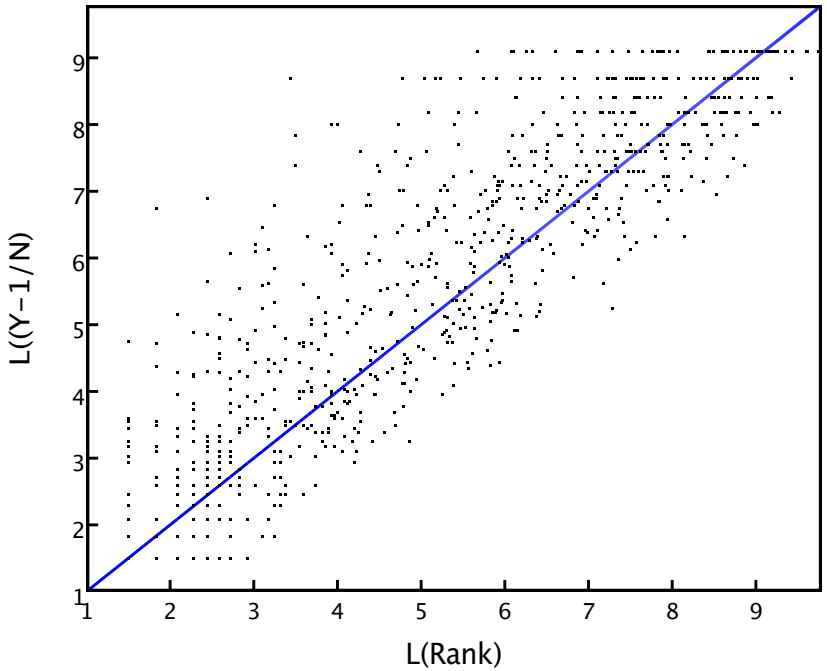
- Embedding model is scalable + performs quite well.
- WARP loss applicable to many large scale retrieval/ranking tasks.

Future Work

- Learn image features that also optimize prec@k ?
- Incorporate pixel predictions into the model.. ?
- Multi-task across datasets, e.g. with NLP tasks..?

WARP Loss: Approximation Accuracy





$\frac{(Y-1)}{E[N]} = (Y-1)/\sum_{i=1}^{\infty} i(1-p)^{i-1}p,$

$E[\frac{(Y-1)}{N}] = \sum_{i=1}^{\infty} \frac{Y-1}{i}(1-p)^{i-1}p$

$p = Pr(violation) = \frac{rank}{Y-1}$

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, $d_{\bar{\emptyset}}$ 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
k-NN	$\mathcal{O}(n \cdot d_{\bar{\emptyset}})$	$\mathcal{O}(n \cdot d_{\bar{\emptyset}})$
Approx. k-NN	$\mathcal{O}((p + n/2^p) \cdot d_{\bar{\emptyset}})$	$\mathcal{O}(n \cdot d)$
One-vs-Rest	$\mathcal{O}(Y \cdot d_{\bar{\emptyset}})$	$\mathcal{O}(Y \cdot d)$
PAMIR^{IA}	$\mathcal{O}(Y \cdot d_{\bar{\emptyset}})$	$\mathcal{O}(Y \cdot d)$
WSABIE	$\mathcal{O}((Y + d_{\bar{\emptyset}}) \cdot D)$	$\mathcal{O}((Y + d) \cdot D)$

Feature Representation

We use the sparse vector representation of [Grangier & Bengio '08]:

- Each image segmented into overlapping blocks at various scales.
- Each block represented by color+edge features.
- Discretized by training kmeans (10,000 "visterms").

Each image represented as a *bag of visual words*: a histogram of the number of times each visual word was present in the image.
10k dim sparse vectors an average of $d_{\bar{\emptyset}} = 245$ non-zero values.
It takes on average 0.4 seconds to extract these features per image.

Bag of Visterms Representation

input	image
block segmentation	set of overlapping blocks
block descriptors	each block is described with color and edge (LBP) histograms
block quantization	each block is mapped to a discrete index, through kmeans learned over the training blocks.
bag of visterms	set of block indexes = set of visual words
output	*tf idf* weighted vector