Sampled Image Tagging and Retrieval Methods on User Generated Content

BMVC 2012 Submission # 956

Abstract

Traditional image tagging and retrieval algorithms have limited generalizability as a result of being trained with heavily curated datasets. These limitations are most evident when arbitrary search words are used that do not intersect with training set labels. Weak labels from user-generated content (UGC) found in the wild (e.g., Google Photos, FlickR, etc.) have an almost unlimited number of unique words in the metadata tags. Prior work on word embeddings successfully leveraged unstructured text with large vocabularies, and our proposed method seeks to apply similar cost functions to open source imagery. Specifically, we train a deep learning image tagging and retrieval system on large-scale UGC using sampling methods and joint optimization of word embeddings. By using the Yahoo! FlickR Creative Commons (YFCC100M) dataset, such an approach builds robustness to common unstructured data issues that include but are not limited to irrelevant tags, misspellings, multiple languages, polysemy, and tag imbalance. The final proposed algorithm not only yields comparable results to state of the art in conventional image tagging, but enables a new capability to train algorithms on large-scale unstructured text in the YFCC100M dataset and outperform cited work in zero-shot capability.

1 Introduction

Automated approaches to image tagging and retrieval have benefited from some of the techniques developed for detection and localization competitions [4, 14, 21] such as the rise of convolutional neural networks (CNNs) [12, 22, 24]. Such algorithms work well on curated datasets but are unfortunately limited in the number of keywords they support due to small training label sets. To be useful, deep learning approaches need to accommodate open vocabulary search, meaning that their implementations would require training label sets to be orders of magnitude larger.

Rather than manually extending curated datasets, one idea is to use open source imagery datasets that are created with user-generated content (UGC) like the Visual Genome project [11] or Yahoo! FlickR Creative Commons (YFCC100M) [25]. These datasets have the advantage of containing an almost unlimited number and variety of unique tags that cover much of the vocabulary of the English language. The issue with training on their metadata tags is that they also include noise in the form of misspellings, unevenly distributed numbers of tags, different languages, and irrelevant and unduly specific tags. Our approach to remedy the so-called "weak labels" in these UGC datasets is to statistically overcome the inherent noise with the sheer scale of the data.

Leveraging the scale of UGC data is non-trivial for a variety of reasons. While there is an abundance of work addressing *image* scale in YFCC100M [17], the focus of our work is on the scale of the *labels*, which take the form of noisy metadata tags. The challenge with this is negotiating matrix operations on deep learning architectures that require a final layer that is proportional to the number of words. The number of weights in this final layer is (#hidden units $\times$ #unique tags), about 43M parameters in our case. As reported in [10], the forward and backpropagation of a single batch on their final layer alone ($\frac{1}{4}$ the size of ours) takes 1.6 seconds, and requires simplifying heuristics. Considering only the last layer, a single epoch through the YFCC100M dataset would take over two weeks!

Fortunately, unstructured text and large vocabulary is well-studied with word embeddings [16, 19]. Several works [13, 18, 27] have sought to exploit word embeddings by projecting image features into the resulting semantic space. Such efforts primarily focus on zero-shot learning by targeting static word vectors as labels. By maintaining static word vectors, these approaches assume that semantic and syntactic similarity equate to visual similarity, which is often not the case (word co-occurrence and parts of speech often have little to do with visual appearance). As a result, projecting into word embedding space merely *uses* the word vectors and does not scale to, nor truly train on, unstructured UGC.

Our proposed method sidesteps this issue by jointly optimizing *both* image and word embeddings, while simultaneously addressing the scale issue through the use of negative sampling and noise contrastive estimation [7]. Specifically, we use the traditional cross-entropy cost function and provide an analytical comparison to ranking cost functions [2, 27], to which we also apply the proposed sampling methods for fair comparison. In doing so, we train against the largest UGC corpus currently available in open source [25], and demonstrate that despite the issues, automated tagging using a sampled cost function can produce considerably more useful information than the original user-generated tags. More importantly, we allow users to search for relevant images using an almost unlimited vocabulary.

2 Approach

Our proposed approach makes use of un-normalized cost functions from the natural language processing domain and uses optimization strategies rooted in unsupervised and embedding approaches. Most notably, Restricted Boltzmann Machines and word embeddings like *word2vec* rely on some variant of noise contrastive estimation, where the distribution of foreground (e.g., the surrounding context of a word) is separated from the distribution of the background (e.g., the probability distribution over all words in the corpora). In our case, the context is the set of tags for each image, and negative samples can be obtained by sampling from the tag distribution. It is then straightforward use the skip-gram approach (Fig. 1) with the following cost function:

$$\begin{aligned}\mathcal{L}(\{W_i\}, v_{p,n}) = & \sum_{p}^{P} \log E_p \left[\sigma(f_{\{W_i\}}^T v_p)\right] + \sum_{n}^{N} \log E_n \left[\sigma(-f_{\{W_i\}}^T v_n)\right] \\ & + \alpha \sum_{p,p'} \sigma(v_p^T v_{p'}) + \sum_{p,n} \sigma(-v_p^T v_n)\end{aligned} \tag{1}$$

where v_p, are positively sampled vectors coming from words the image has been tagged with, v_n are the negatively sampled vectors from the probability distribution over all possible tags, P is the number of positive samples, N is the number of negative samples, and the feature vector f is parameterized by the set of weights $\{W_i\}$ from a neural network $h(\cdot, \{W_i\})$:

$$f_{\{W_i\}} = h(x, \{W_i\}) \tag{2}$$

with inputs being ImageNet Large Scale Visual Recognition Competition [21] features x.

The first term in (1) positively correlates the feature vector with the metadata tags, pulling the image closer to the context through backpropagation over $\{W_i\}$. The second term pushes the them away from the background distribution. The final two terms, with α taken to be small (we use 0.01), serves to promote similarity in co-occurring tags. The architecture of our neural network is:

$$\text{imfeats} \rightarrow 4096 \rightarrow 8192 \rightarrow 2048 \rightarrow 300 \rightarrow \text{wordvecs}$$

We try both *Inception* and *VGG* features in combination with word vectors (*word2vec* and *Glove*). Importantly, (1) not only takes the set of weights $\{W_i\}$ from the neural network $h(\cdot)$ for parameters, but is also, notably different than most cited work [13, 18, 27], a function of $v_{p,n}$. In other words, our method also optimizes the word vectors.

2.1 Out of Vocabulary Updates

While the tag set in a UGC corpus has an extensive coverage of the English language, it still does not include all possible words. We address this by seeding our model with a pre-trained word embedding [15], since the set difference between many pre-trained word embeddings and the metadata in the image corpus is non-trivial. However this introduces a new problem: as we only train on samples in the image corpus, words in the set difference will never be updated when jointly optimizing in Sec. 3. The relationship between optimized and unoptimized words quickly devolves, and the dot product between them becomes meaningless.

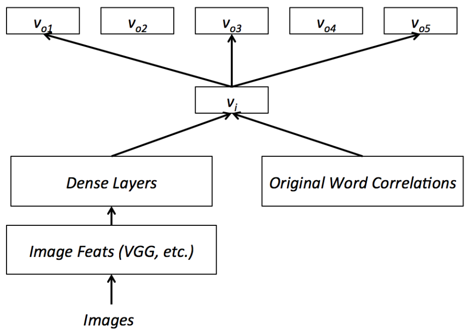


Figure 1: Jointly optimizing word vectors and image weights.

Our solution to this, illustrated in Fig. 1, is to make a final optimization pass after training the neural network on the image corpus. This step makes the most sense when performed *offline*, simply snapping out-of-vocabulary words into place. This allows us to apply what we know of the semantic relationship between the out-of-vocabulary words and the in-vocabulary words, i.e., words that have been optimized when training the neural network.

Let V_W be word vectors from the text corpus, and V_I be word vectors from the image corpus. The set difference is $\{V_D\} = \{V_W\} - \{V_I\}$. Without loss of generality, let us assume that all words in the image corpus exist in the word corpus, i.e. $\{V_I\} \subset \{V_W\}$ and furthermore, we can organize $V_\cup$ to be ordered in the following manner: $V_\cup = [V_I | V_D]$.

Before training in Sec. 2.2, let us preserve the initial word vectors V_I and V_D. We can write a nonlinear correlation matrix[1] for every word vector to every other word vector as:

$$C^{(i)} = \sigma \left(\begin{bmatrix} V_I^T V_I & V_I^T V_D \\ V_D^T V_I & V_D^T V_D \end{bmatrix} \right), \tag{3}$$

where the superscript i denotes the initial semantic relationships of each word to one another. After training with our image corpus, all the vectors in V_I will have changed while none of the vectors in V_D will have been updated. To update V_D in the absence of any image information, we can only rely on semantic information, which are specified by the relationships in $C^{(i)}$ the lower and right-hand submatrices of $C^{(i)}$. Specifically, we wish to match initial and final correlations from seen words to unseen words and between the unseen words themselves:

$$C_{d,m}^{(i)} \log \sigma(v_d^T v_m) + (1 - C^{(i)}) \log \left(1 - \sigma(v_d^T v_m)\right) \tag{4}$$

for $v_d \in \{V_D\}$ and $v_m \in \{V_\cup\}$.

2.2 On Sampling

Inherent in (1) is the idea that positive and negative sampling can converge to a meaningful result in expectation. A similar approach was initially attempted in [27] using single samples, but was quickly abandoned due to updates being too sparse. In practice, with proper initalization of the final layer (i.e., using pre-trained word vectors) with a large corpus (we used New York Times and Wikipedia [15]), reasonable image retrieval results begin appearing and converging with a single pass through the YFCC100M data, at least for frequently occurring words. To assess the efficiency of sampling, see Fig. 2 which was created using a smaller corpora where it is possible to use traditional optimization in the cross-entropy loss function as a comparison point with respect to sampled loss.

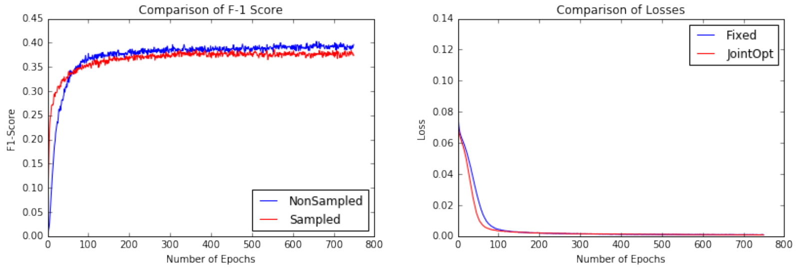
Since our variables are tensors, we require the number of samples to be consistent between images for batch purposes. That is, we must fix P and N. We choose a fixed number of positive samples, chosen uniformly since we make no assumptions on tag ordering. For any image with fewer than P tags, we sample with replacement to reach our threshold.

	291 Labels 17665 Images		925 Labels 102709 Images	
	Full	Sampled	Full	Sampled
Fast0Tag [27]	2.12s	0.39s	43.6s	3.22s
Fixed WV	1.21s	0.32s	9.3s	1.94s
Proposed Approach	1.30s	0.43s	10.9s	2.31s

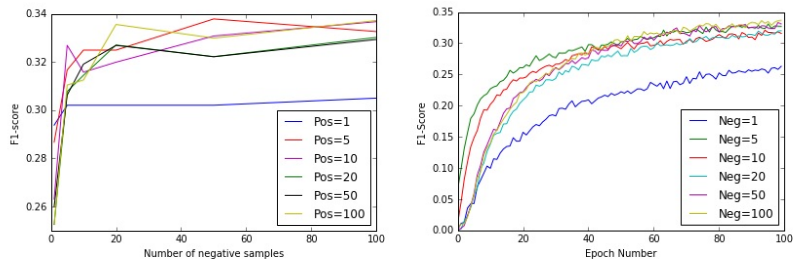
Table 1: Timing Per Epoch.

We choose $P = 5$ and $N = 10$ because the knee in the curve appears in Fig. 2(c) at under 10 for positive and negative samples. Another of the more concrete conclusions in Fig. 2(d) is that single samples in both positive and negative universally performs more poorly across 100 epochs. In Fig. 2(c), we observe that performance approaches near-parity when $P \geq 5$ for larger numbers of samples. The number of negative samples seems to have less of an effect than previously thought as we observe similar performance when $N \leq 100$ neg samples in Fig. 2(d). This has implications in how neural networks are trained in general when it is

[1] Note that for computational purposes, we do not explicitly construct $C^{(i)}$, but rather store a subset of the original word vectors and compute dot products online.



(a) The effect of sampling (b) The effect of joint optimization



(c) On varying the number of positive samples (d) On varying the number of negative samples

Figure 2: Sampling loss plots

noted that training and optimizing for all the zeros in one/multi-hot encodings is common. The plots seem to suggest that this is unnecessary.

The computational complexity of (1) is $\mathcal{O}(\max(P,N))$ per image. For comparison, Fast0Tag [27] has complexity that scales according to $\mathcal{O}(PN)$ per image because of its double sum over p and n. Without sampling, if there are more than a few labels per image, this balloons quite significantly, particularly if there are large numbers of positive labels for a single image because it ranks every tag to every other tag for every single image in a minibatch. Table 1 shows the unsurprising effect of sampling versus a full optimization. Transitioning from 291 tags (IAPR TC-12 [5] 2s/epoch) to 925 tags (NUS-WIDE [3], 43s/epoch) was significant, while going from 925 tags to 13980 (Visual Genome [11]) was untenable on our TitanX GeForce NVIDIA card due to both memory/time.

2.3 On the Cost Function

The proposed objective (1) stands in contrast to prior work based on projections of images into semantic space based on ranking. Most notably, the Fast0Tag objective which originates from RankNet [2] can be rewritten to take the form:

$$\mathcal{L} = \beta \sum_p \sum_n \log \sigma \left(f^T (v_p - v_n) \right) \tag{5}$$

The form of (5) appears different from (1), but contrasts can be conceptualized with some linear algebra and simplifying assumptions. As discussed, we are sampling P and N, but for sake of explanation, let us assume that $P = N$ (it does not), remove the expectations, and for clarity, let's assign $L = P = N$. Then the first portion of (1) can be written:

$$\begin{aligned}
\mathcal{L} &= \sum_{p}^{L} \log \sigma(v_p^T f_n) + \sum_{n}^{L} \log \sigma(-v_n^T f) \\
&= \sum_{p}^{L}\sum_{n}^{L} \frac{1}{L} \log\left(\sigma(f^T v_p)\sigma(-f^T v_n)\right) \\
&= \beta' \sum_{p}^{L}\sum_{n}^{L} \log\left(\sigma\left(f^T(v_p - v_n)\right) + e^{v_p^T f} + e^{-v_n^T f}\right)
\end{aligned} \tag{6}$$

Indeed, while (5) ranks the *difference* between every positive example to every other negative example, the extra terms in (6) are dot products between images and words to maximize or minimize. Thus, rather than operate only no *directionality*, the nonlinear correlations between individual images and relevant individual tags are also weighted.

3 Implementation

To replicate our work (code at `http://released-upon-acceptance`), nontrivial details remain in the execution of the ideas in Sec. 2.2 due to the scale of the data. Such considerations include pre-training the final layer of our neural network, the partition of the architecture onto GPU/CPU memory, and additional tricks that are necessary for sampling.

3.1 Pre-training the Final Layer

There are two issues with initializing the final layer. First, rare words will have few examples in the tag set and hence have poor quality vectors. Second, the layer has so many parameters that the tags alone do not provide enough data to train it. Pre-training using a very large word corpus alleviates these concerns. We pre-train the final layer using *word2vec* or *Glove*[2]. We tested several corpora and settled on Wikipedia8B [15]. The dimensionality of the hidden layer before the final word embedding is 300 and thus each word vector is 300 dimensions. These vectors are stored in a $432,213 \times 300$ weight matrix.

3.2 GPU/CPU Split

Keeping parameters in memory decreases training time, but work in deep learning, which mostly focuses on using GPUs, has shown that negotiating GPU DRAM (memory) hampers the flexibility that a researcher needs to fully explore a domain [20]. GPU memory capacity and sharing have made remarkable advances in hardware, but, at least for now, servers with NVLink and multi-GPU nodes are expensive. Onboard server memory is abundant and so we look to exploit this memory instead by performing CPU based calculations where possible.

Mikolov's original *word2vec* is trained using multi-threaded CPUs. Since his work deals primarily with unstructured text, his neural network is wide rather than deep, using a single hidden layer. For this reason, our algorithm makes use of both CPU and GPU. We place the word embeddings (final layer of the neural network) off GPU, and the corresponding gradient calculations in (1) are done with the CPU. The backpropagation through the remainder of

[2] We tried both; *Glove* tended to provide better results, though we did not explore many text corpora.

	NUS-Wide Dataset [3]						Multi-Corpora								
	925→81			925→1006			IA→ESP			VG→ESP			YFCC→ESP		
	P	R	F1	P	R	F1	P	R	F1	P	R	F1	P	R	F1
Fast0Tag	16.2	**39.3**	22.9	15.6	14.9	15.2	15.7	6.9	9.6	Will Not Scale			Will Not Scale		
Fixed WV	**21.2**	36.3	26.9	17.1	14.8	16.3	**16.2**	6.1	9.4	14.1	17.9	15.7			
Full XEntropy	21.1	37.3	**27.0**	**17.3**	**15.9**	**16.1**	16.3	**7.0**	**9.8**	**14.8**	**18.1**	**16.3**			
S+Fast0Tag	15.0	43.4	22.3	12.4	10.3	11.6	5.9	8.3	6.9	**17.6**	18.4	18.0	5.1	3.9	4.4
S Fixed WV	**15.9**	44.2	**22.9**	13.0	11.3	12.1	10.0	5.2	7.1	16.6	**19.8**	18.1	17.4	**17.6**	17.5
Proposed Algorithm	15.4	**44.6**	22.9	**13.1**	**11.6**	**12.3**	**13.3**	**10.2**	**12.1**	17.3	19.0	**18.1**	**21.9**	15.1	**17.9**

Table 2: Zero-shot and multi-corpus tagging top 5 results for precision/recall/F1. Due to space constraints, we report at limited precision, but the bolded results are the hightest results at full precision.

the neural network (all the dense layers prior) are still done on the GPU using Tensorflow. Such a split makes training times quicker and feasible.

3.3 Scaled Sampling

We use words that occur with the highest frequencies, which limits the number to 423k. To ensure consistent tensor dimensions, for each image, we sampled 5-10 positive tags uniformly.[3] Unfortunately, sampling from a 432k dimensional distribution is time consuming and our original code profile assessed that 60% of time was spent in sampling.

To side-step this issue, we explored two options: (1) pre-sampling at each epoch and (2) using the metadata from an adjacent image. The first option is done by taking a random subset from the distribution of tags in the corpus at each epoch. During backpropagation, we sample again from this random subset. This fast sampling may create inherent bias issues as the chance of re-sampling frequently used words is high. Using the metadata from the adjacent image in the batch avoids this problem because the images are randomly selected. Using this method reduced computation time to negligible rates. Overall, by sampling, we can iterate through an entire epoch of YFCC100M in under 3 hours. The bulk of our models converged to meaningful results within a *single* epoch. What follows in Sec. 4 were rapidly prototyped for 40 epochs through the YFCC dataset.

4 Results

While the primary objective is to train on UGC, we perform quantitative metrics on curated, traditional corpora in order to compare against state of the art. These include IAPR TC-12 (IA - 291 unique tags) [5], ESP-game (ESP - 288 tags, INRIA-LEAR's version) [26], NUS-WIDE (using splits from [27], 81/925/1006 tags) [3], Visual Genome (VG - 13980 unique tags) [11], and YFCC100M [25] (millions, but we pruned to 432k tags) with both InceptionNet [24] and downloaded YFCC-VGG features [23].

To assess image tagging capability, we train, validate, and test against proper splits from a single corpus. To assess generalization capability for a variety of content and word tags, we perform cross-corpus evaluation: train on a single dataset, then test on a different dataset. For example, IA→ESP is trained and validated on the IAPR TC-12 training/validation splits and tested on the ESP-game test split. For YFCC100M, we only test on ESP-Game since YFCC100M→YFCC100M evaluation is not meaningful due to noisy truth data. Along with cited algorithms in Table 3, neural network approaches (both sampled and unsampled) using the cross-entropy function (with joint optimization) have run for a total of 750 epochs of the

[3] In future implementations, we imagine this would be some form of an inverse distribution of the tag frequency.

Figure 3: YFCC Image Retrieval and Truth Tags. Each pair of columns is a search term, and the top six images are retrieved. To demonstrate how much we rely on the statistical properties of the dataset, the metadata tags are also shown next to each image retrieved. In many cases, the metadata is in a different language (which we had to convert from URL encoded strings in UTF-8.) On a side note, it is apparent that a significant portion of weddings in the YFCC100M occur in Taiwan.

data. The effect of sampling on accuracy can be shown to be negligible during training while vastly improving running times as previously shown in Table 1.

The proposed methods are denoted in Table 2 and Table 3 as X-Ent, Opt+X-Ent, and S+Opt+X-Ent. The "Opt" stands for an optimized final word vector layer, the "S" stands for the sampled version, and X-Ent means a cross-entropy cost function with a word vector final layer. We also include "AvgWV" as a benchmark in Table 3, referring to an approach using the average of an image's tag vectors as a target for the deep neural network.

Image Annotation						
	IA [5]			ESP [26]		
Top 5	P	R	F1	P	R	F1
least-squares	40	19	26	35	19	25
Avg WV	21	13	16	38	20	26
TagProp [6]	45	34	39	39	27	32
FastTag [1]	**47**	26	34	**46**	22	30
Fast0Tag [27]	41	33	36	38	35	36
Fixed WV	44	34	39	38	36	37
Full Cross-entropy	43	36	**39**	38	36	37
Sampled Fast0Tag	37	38	37	37	38	38
Sampled Fixed WV	**38**	39	38	37	37	37
Proposed Algorithm	38	**39**	38	37	**39**	**38**

Table 3: On the effect of using Fast0Tag and other methods versus the cross-entropy cost function for a single corpus evaluation. We selected the higher number at precision.

We use a provided implementation of Fast0Tag [27] from the author as well as an original implementation using sampling, denoted in the tables as S+Fast0Tag. Such an implementation was able to achieve reasonable runtimes (on par with cross-entropy) that plagued the original implementation, which is evident in Table 1 in Sec. 2.2. We used the sampled code in place of the provided code for larger datasets, where the provided code was incapable of performing due to memory and complexity issues. It is important to note that many of the algorithms compared against [1, 6], especially the nearest neighbor ones, simply cannot address the large scale vocabulary in UGC nor the quantity of images in a reasonable amount of time, either at inference time or during training time.

Across both Table 2 and Table 3, optimized cross-entropy outperforms Fast0Tag [27], whether sampled or non-sampled. Although the goal in sampling was not to achieve the highest accuracy but to deal with scale, in many cases, adding sampling to the approach boosted recall for both conventional and cross-corpora evaluation. Our explanation for this phenomenon is that sampling helped to deal with noisy tagging.

Zero-shot capability is baked into the NUS-WIDE dataset with splits developed by [27] and varying numbers of tags. The 925 tag split does not intersect with the 81 tag splits, and the 1006 tag split is the union. We also include *cross-corpora* results where we train on one dataset and test on another dataset. However, the goal of the proposed algorithm is generalization using a large vocabulary. This was achieved through two large datasets: the Visual Genome object dataset (a subset of annotations distinct from the localization and captioning set) and to a larger extent, the YFCC100M dataset. We perform some pre-processing, removing images with no tags and cutting off low-frequency words to achieve a tag count of approximately 14k and 432k words, respectively.

MS COCO [14]			
	Top 5 Performance		
Top 5	P	R	F1
Average WordVec	**43.2**	61.8	44.0
Correlated WV [8]	38.0	70.6	48.4
Fast0Tag	40.3	68.7	50.8
Fixed WV	40.0	68.6	50.5
X-Entropy	42.1	68.6	52.1
Sampled Fast0Tag	38.1	69.6	49.2
Sampled Fixed WV	38.0	69.6	49.2
Proposed Algorithm	42.0	**72.5**	**53.2**

Visual Genome			
	Top 5 Perforamnce		
Top 5	P	R	F1
Average WordVec	14.2	4.4	6.7
DenseCap-Objects	13.3	**14.9**	**14.0**
Sampled Fast0Tag	14.2	5.6	8.1
Sampled Fixed WV	14.5	5.3	7.7
Proposed Algorithm	**17.6**	10.3	13.0

Table 4: Results without training on localization information using the Microsoft COCO objects dataset [14] and Visual Genome dataset.

Finally, for comparison, we also retrained DenseCap [9] to identify objects rather than phrases, replacing its recurrent network with a multi-hot encoding as the final layer after localization. Densecap has an unfair advantage in that it is designed specifically for localized object detection, but it is apparent that the proposed algorithm still has higher precision and comparable retrieval and F1 scores.

5 Conclusions

We have proposed a method that can tag and retrieve images with UGC scale vocabulary through joint image and word vector optimization and sampling methods. We have demonstrated that our tagging mechanism can yield considerably more useful information than the original tags themselves.

References

[1] M. Chen, A. Zheng, and K.Q. Weinberger. Fast image tagging. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2013)*, 2013.

[2] Wei Chen, Tie yan Liu, Yanyan Lan, Zhi ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, pages 315–323. Curran Associates, Inc., 2009. URL http://papers.nips.cc/paper/3708-ranking-measures-and-loss-functions-in-learning-to-rank.pdf.

[3] Tat-Seng Chua, Jinhui Tang, Richang Hong, Haojie Li, Zhiping Luo, and Yan-Tao. Zheng. Nus-wide: A real-world web image database from national university of singapore. In *Proc. of ACM Conf. on Image and Video Retrieval (CIVR'09)*, Santorini, Greece., July 8-10, 2009.

[4] Mark Everingham, Luc Gool, Christopher K. Williams, John Winn, and Andrew Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vision*, 88(2):303–338, June 2010. ISSN 0920-5691. doi: 10.1007/s11263-009-0275-4. URL http://dx.doi.org/10.1007/s11263-009-0275-4.

[5] M. Grubinger, P.D. Clough, H. Muller, and T. Deselaers. The iapr benchmark: A new evaluation resource for visual information systems. In *Proceedings of the International Conference on Language Resources and Evaluation*, 2006.

[6] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *IEEE 12th International Conference on Computer Vision (ICCV 2009)*, 2009.

[7] Michael U. Gutmann and Aapo Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13(1):307–361, February 2012. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2503308.2188396.

[8] Yuexian Hou, Jian-Yun Nie, Le Sun, Bo Wang, and Peng Zhang, editors. *Information Retrieval Technology, 8th Asia Information Retrieval Societies Conference, AIRS 2012, Tianjin, China, December 17-19, 2012. Proceedings*, volume 7675 of *Lecture Notes in Computer Science*, 2012. Springer. ISBN 978-3-642-35340-6. doi: 10.1007/978-3-642-35341-3. URL http://dx.doi.org/10.1007/978-3-642-35341-3.

[9] Justin Johnson, Andrej Karpathy, and Li Fei-Fei. Densecap: Fully convolutional localization networks for dense captioning. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[10] Armand Joulin, Laurens van der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. *CoRR*, abs/1511.02251, 2015. URL http://arxiv.org/abs/1511.02251.

[11] Ranjay Krishna, Yuke Zhu, Oliver Groth, Justin Johnson, Kenji Hata, Joshua Kravitz, Stephanie Chen, Yannis Kalanditis, Li-Jia Li, David A Shamma, Michael Bernstein, and Li Fei-Fei. Visual genome: Connecting language and vision using crowdsourced dense image annotations. 2016.

[12] A. Krizhevsky, I. Sutskever, and G.E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25 (NIPS 2012)*, 2012.

[13] Xirong Li, Shuai Liao, Weiyu Lan, Xiaoyong Du, and Gang Yang. Zero-shot image tagging by hierarchical semantic embedding. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 879–882, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3621-5. doi: 10.1145/2766462.2767773. URL http://doi.acm.org/10.1145/2766462.2767773.

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. URL http://arxiv.org/abs/1405.0312.

[15] Matt Mahoney. Wikipedia 8 Billion Character Dataset. http://mattmahoney.net/dc/text.html, 2006.

[16] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26 (NIPS 2013)*, 2013.

[17] Karl Ni, Roger A. Pearce, Kofi Boakye, Brian Van Essen, Damian Borth, Barry Chen, and Eric Wang. Large-scale deep learning on the YFCC100M dataset. *CoRR*, abs/1502.03409, 2015. URL http://arxiv.org/abs/1502.03409.

[18] Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *CoRR*, abs/1312.5650, 2013. URL http://arxiv.org/abs/1312.5650.

[19] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

[20] Minsoo Rhu, Natalia Gimelshein, Jason Clemons, Arslan Zulfiqar, and Stephen W. Keckler. Virtualizing deep neural networks for memory-efficient neural network design. *CoRR*, abs/1602.08124, 2016. URL http://arxiv.org/abs/1602.08124.

[21] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge, 2014.

[22] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014.

[23] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *CoRR*, abs/1409.1556, 2014. URL http://arxiv.org/abs/1409.1556.

[24] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014. URL http://arxiv.org/abs/1409.4842.

[25] Bart Thomee, David A. Shamma, Gerald Friedland, Benjamin Elizalde, Karl Ni, Douglas Poland, Damian Borth, and Li-Jia Li. The new data and new challenges in multimedia research. *arXiv preprint arXiv:1503.01817*, 2015.

[26] Ingmar Weber, Stephen Robertson, and Milan VojnoviÄĞ. Rethinking the esp game. In *In CHI*, pages 3937–3942, 2009.

[27] Yang Zhang, Boqing Gong, and Mubarak Shah. Fast zero-shot image tagging. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2016.