Love Thy Neighbors: Image Annotation by Exploiting Image Metadata

Justin Johnson*   Lamberto Ballan*   Li Fei-Fei
Computer Science Department, Stanford University
{jcjohns,lballan,feifeili}@cs.stanford.edu

Abstract

Some images that are difficult to recognize on their own may become more clear in the context of a neighborhood *of related images with similar social-network metadata. We build on this intuition to improve multilabel image annotation. Our model uses image metadata nonparametrically to generate neighborhoods of related images using Jaccard similarities, then uses a deep neural network to blend visual information from the image and its neighbors. Prior work typically models image metadata parametrically; in contrast, our nonparametric treatment allows our model to perform well even when the vocabulary of metadata changes between training and testing. We perform comprehensive experiments on the NUS-WIDE dataset, where we show that our model outperforms state-of-the-art methods for multilabel image annotation even when our model is forced to generalize to new types of metadata.*

1. Introduction

Take a look at the image in Figure 1a. Might it be a flower petal, or a piece of fruit, or perhaps even an octopus tentacle? The image on its own is ambiguous. Take another look, but this time consider that the images in Figure 1b share social-network metadata with Figure 1a. Now the answer is clear: all of these images show flowers. The context of additional unannotated images disambiguates the visual classification task. We build on this intuition, showing improvements in multilabel image annotation by exploiting image metadata to augment each image with a *neighborhood* of related images.

Most images on the web carry metadata; the idea of using it to improve visual classification is not new. Prior work takes advantage of user tags for image classification and retrieval [19, 5, 23, 38], uses GPS data [20, 35, 48] to improve image classification, and utilizes timestamps [26] to both improve recognition and study topical evolution over time. The motivation behind much of this work is the notion that images with similar metadata tend to depict similar scenes.

One class of image metadata where this notion is par-

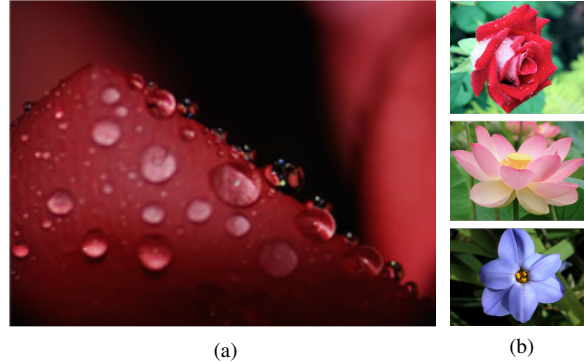(a)   (b)

Figure 1: On its own, the image in (a) is ambiguous - it might be a flower petal, but it could also be a piece of fruit or possibly an octopus tentacle. In the context of a *neighborhood* (b) of images with similar metadata, it is more clear that (a) shows a flower. Our model utilizes image neighborhoods to improve multilabel image annotation.

ticularly relevant is *social-network metadata*, which can be harvested for images embedded in social networks such as Flickr. These metadata, such as user-generated tags and community-curated groups to which an image belongs, are applied to images by people as a means to communicate with other people; as such, they can be highly informative as to the semantic contents of images. McAuley and Leskovec [37] pioneered the study of multilabel image annotation using metadata, and demonstrated impressive results using only metadata and no visual features whatsoever.

Despite its significance, the applicability of McAuley and Leskovec's method to real-world scenarios is limited due to the parametric method by which image metadata is modeled. In practice, the vocabulary of metadata may shift over time: new tags may become popular, new image groups may be created, etc. An ideal method should be able to handle such changes, but their method assumes identical vocabularies during training and testing.

In this paper we revisit the problem of multilabel image annotation, taking advantage of both metadata and strong visual models. Our key technical contribution is to generate *neighborhoods* of images (as in Figure 1) nonparametrically using image metadata, then to operate on these neighbor-

---
*Indicates equal contribution.

hoods with a novel parametric model that learns the degree to which visual information from an image and its neighbors should be trusted.

In addition to giving state-of-the-art performance on multilabel image annotation (Section 5.1), this approach allows our model to perform tasks that are difficult or impossible using existing methods. Specifically, we show that our model can do the following:

- **Handle different types of metadata.** We show that the same model can give state-of-the-art performance using three different types of metadata (image tags, image sets, and image groups). We also show that our model gives strong results when different metadata are available at training time and testing time.

- **Adapt to changing vocabularies.** Our nonparametric approach to handling metadata allows our model to handle different vocabularies at train and test time. We show that our model gives strong performance even when the training and testing vocabulary of user tags are completely disjoint.

2. Related Work

Automatic image annotation and image search. Our work falls in the broad area of image annotation and search [34]. Harvesting images from the web to train visual classifiers without human annotation is an idea that have been explored many times in the past decade [14, 45, 32, 3, 43, 7, 10, 6]. Early work on image annotation used voting to transfer labels between visually similar images, often using simple nonparametric models [36, 33]. This strategy is well suited for multimodal data and large vocabularies of weak labels, but is very sensitive to the metric used to find visual neighbors. Extensions use learnable metrics and weighted voting schemes [18, 44], or more carefully select the training images used for voting [47]. Our method differs from this work because we do not transfer labels from the training set; instead we compute nearest-neighbors between *test-set* images using metadata.

These approaches have shown good results, but are limited because they treat tags and visual features separately, and may be biased towards common labels. Some authors instead tackle multilabel image annotation by learning parametric models over visual features that can make predictions [17, 45, 49, 15] or rank tags [29]. Gong *et al.* [15] recently showed state of the art results on NUS-WIDE [8] using CNNs with multilabel ranking losses. These methods typically do not take advantage of image metadata.

Multimodal representation learning: images and tags. A common approach for utilizing image metadata is to learn a joint representation of image and tags. To this end,

prior work generatively models the association between visual data and tags or labels [30, 2, 4, 40] or applies non-negative matrix factorization to model this latent structure [50, 13, 25]. Similarly, Niu *et al.* [38] encode the text tags as relations among the images, and define a semi-supervised relational topic model for image classification. Another popular approach maps images and tags to a common semantic space, using CCA or kCCA [46, 23, 16, 1]. This line of work is closely related to our task, however these approaches only model user tags and assume static vocabularies; in contrast we show that our model can generalize to new types of metadata.

Beyond images and tags. Besides user tags, previous work uses GPS and timestamps [20, 35, 26, 48] to improve classification performance in specific tasks such as landmark classification. Some authors model the relations between images using multiple metadata [41, 37, 11, 28, 12]. Duan *et al.* [11] present a latent CRF model in which tags, visual features and GPS-tags are used jointly for image clustering. McAuley and Leskovec model pairwise social relations between images and then apply a structural learning approach for image classification and labeling [37]. They use this model to analyze the utility of different types of metadata for image labeling. Our work is similarly motivated, but their method does not use any visual representation. In contrast, we use a deep neural network to blend the visual information of images that share similar metadata.

3. Model

We design a system that incorporates both visual features of images and the neighborhoods in which they are embedded. An ideal system should be able to handle different types of signals, and should be able to generalize to new types of image metadata and adapt to their changes over time (e.g. users add new tags or add images to photo-sets). To this end we use metadata nonparametrically to generate image neighborhoods, then operate on images together with their neighborhoods using a parametric model. The entire model is summarized in Figure 2.

Let X be a set of images, Y a set of possible labels, and $\mathcal{D} = \{(x, y) \mid x \in X, y \subseteq Y\}$ a dataset associating each image with a set of labels. Let Z be a set of possible neighborhoods for images; in our case a neighborhood is a set of related images, so Z is the power set $Z = 2^X$.

We use metadata to associate images with neighborhoods. A simple approach would assign each image $x \in X$ to a single neighborhood $z \in Z$; however there may be more than one useful neighborhood for each image. As such, we instead use image metadata to generate a set of *candidate neighborhoods* $Z_x \subseteq Z$ for each image x.

At training time, each element of Z_x is a set of training images, and is computed using training image metadata. At
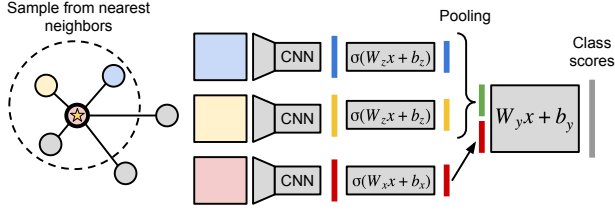
Figure 2: Schematic of our model. To make predictions for an image, we sample several of its nearest neighbors to form a *neighborhood* and we use a CNN to extract visual features. We compute hidden state representations for the image and its neighbors, then operate on the concatenation of these two representations to compute class scores.

test time, test image metadata is used to build Z_x from test images; note that we do not use the training set at test time.

For an image $x \in X$ and neighborhood $z \in Z_x$, we use a function f parameterized by weights w to predict label scores $f(x, z; w) \in \mathbb{R}^{|Y|}$ for the image x. We average these scores over all candidate neighborhoods for x, giving

$$s(x; w) = \frac{1}{|Z_x|} \sum_{z \in Z_x} f(x, z; w). \qquad (1)$$

To train the model, we choose a loss ℓ and optimize:

$$w^* = \arg\min_w \sum_{(x,y) \in \mathcal{D}} \ell(s(x; w), y). \qquad (2)$$

The set Z_x may be large, so for computational efficiency we approximate $s(x; w)$ by sampling from Z_x. During training, we draw a single sample during each forward pass and at test time we use ten samples.

3.1. Candidate Neighborhoods

We generate candidate neighborhoods using a nearest-neighbor approach. We use image metadata to compute a distance between each pair of images. We fix a *neighborhood size* $m > 0$ and a *max rank* $M \geq m$; the candidate neighborhoods Z_x for an image x then consist of all subsets of size m of the M-nearest neighbors to x.

The types of image metadata that we consider are user tags, image photo-sets, and image groups. Sets are galleries of images collected by the same user (e.g. pictures from the same event such as a wedding). Image groups are community-curated; images belonging to the same concept, scene or event are uploaded by the social network users. Each type of metadata has a vocabulary T of possible values, and associates each image $x \in X$ with a subset $t_x \subseteq T$ of values. For tags, T is the set of all possible user tags and t_x are the tags for image x; for groups (and sets), T is the set of all groups (sets), and t_x are the groups (sets) to which

x belongs. For sets and groups, we use the entire vocabulary T; in the case of tags we follow [37] and select only the τ most frequently occurring tags on the training set.

We compute the distance between images using the Jaccard similarity between their image metadata. Concretely, for $x, x' \in X$ we compute

$$d(x, x') = 1 - |t_x \cap t_{x'}|/|t_x \cup t_{x'}|. \qquad (3)$$

To prevent an image from appearing in its own neighborhoods, we set $d(x, x) = 0$ for all $x \in X$.

Generating candidate neighborhoods introduces several hyperparameters, namely the neighborhood size m, the max rank M, the type of metadata used to compute distances, and the tag vocabulary size τ. We show in Section 5.2 that the type of metadata is the only hyperparameter that significantly affects our performance.

3.2. Label Prediction

Given an image $x \in X$ and a neighborhood $z = \{z_1, \ldots, z_m\} \in Z$, we design a model that incorporates visual information from both the image and its neighborhood in order to make predictions for the image. Our model is essentially a fully-connected two layer neural network applied to features from the image and its neighborhood, except that we pool over the hidden states for the neighborhood images.

We use a CNN [31, 27] ϕ to extract d-dimensional features from the images x and z_i. We compute an h-dimensional hidden state for each image by applying an affine transform and an elementwise ReLU nonlinearity $\sigma(\xi) = \max(0, \xi)$ to its features. To let the model treat hidden states for the image and its neighborhood differently, we apply distinct transforms to $\phi(x)$ and $\phi(z_i)$, parameterized by $W_x \in \mathbb{R}^{d \times h}, b_x \in \mathbb{R}^h$ and $W_z \in \mathbb{R}^{d \times h}, b_z \in \mathbb{R}^h$.

At this point we have hidden states $v_x, v_{z_i} \in \mathbb{R}^h$ for x and each $z_i \in z$; to generate a single hidden state $v_z \in \mathbb{R}^h$ for the neighborhood z we pool each v_{z_i} elementwise so that $(v_z)_j = \max_i (v_{z_i})_j$. Finally to compute label scores $f(x, z; w) \in \mathbb{R}^{|Y|}$ we concatenate v_x and v_z and pass them through a third affine transform parameterized by $W_y \in \mathbb{R}^{2h \times |Y|}, b_y \in \mathbb{R}^{|Y|}$. To summarize:

$$v_x = \sigma(W_x \phi(x) + b_x) \qquad (4)$$

$$v_z = \max_{i=1,\ldots,m} \left(\sigma(W_z \phi(z_i) + b_z) \right) \qquad (5)$$

$$f(x, w; z) = W_y \begin{bmatrix} v_x \\ v_z \end{bmatrix} + b_y \qquad (6)$$

The learnable parameters are W_x, b_x, W_z, b_z, W_y, and b_y.

3.3. Learned Weights

An example of a learned matrix W_y is visualized in Figure 3. The left and right sides multiply the hidden states for the image and its neighborhood respectively. Both sides
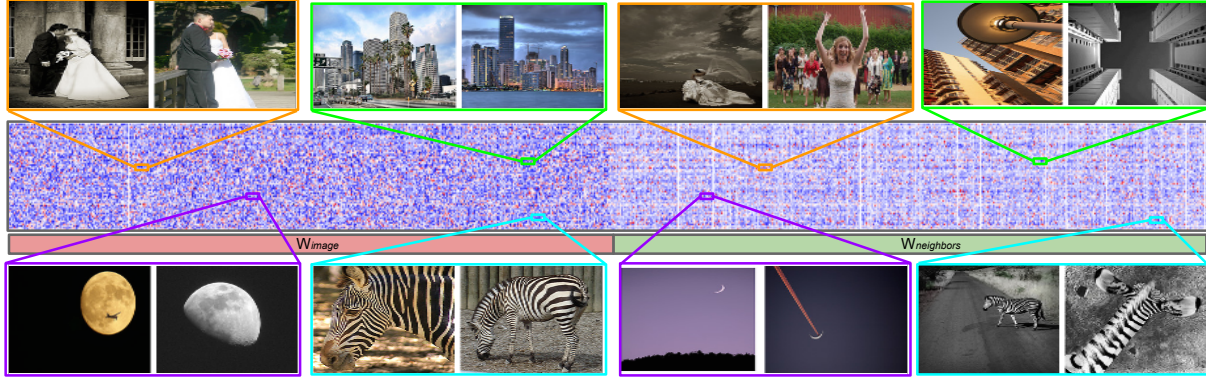
Figure 3: Learned weights W_y. The model uses features from both the image and its neighbors. We show examples of images whose label scores are influenced more by the image and by its neighborhood; images with the same ground-truth labels are highlighted with the same colors. Images that are influenced by their neighbors tend to be non-canonical views.

contain many nonzero weights, indicating that the model learns to use information from both the image and its neighborhood; however the darker coloration on the left suggests that information from the image is weighted more heavily.

We can follow this idea further, and use Equation 6 to compute for each image the portion of its score for each label that is due to the hidden state of the image v_x and its neighborhood v_z. The left side of Figure 3 shows examples of correctly labeled images whose scores are more due to the image, while the right shows images more influenced by their neighborhoods. The former show canonical views (such as a bride and groom for *wedding*) while the latter are more non-canonical (such as a *zebra* crossing a road).

3.4. Implementation details

We apply L_2 regularization to the matrices W_x, W_z, and W_y and apply dropout [22] with $p = 0.5$ to the hidden layers h_x and h_z. We initialize all parameters using the method of [21] and optimize using stochastic gradient descent with a fixed learning rate, RMSProp [42], and a minibatch size of 50. We train all models for 10 epochs, keeping the model snapshot that performs the best on the validation set. For all experiments we use a learning rate of 1×10^{-4}, L_2 regularization strength 3×10^{-3} and hidden dimension $h = 500$; these values were chosen using grid search.

Our image feature function ϕ returns the activations of the last fully-connected layer of the BLVC Reference CaffeNet [24], which is similar to the network architecture of [27]. We ran preliminary experiments using features from the model of VGG [39], but this did not significantly change the performance of our model. For all models our loss function ℓ is a sum of independent one-vs-all logistic classifiers.

4. Experimental Protocol

4.1. Dataset

In all experiments we use the NUS-WIDE dataset [8], which has been widely used for image labeling and re-

trieval. It consists of 269,648 images collected from Flickr, each manually annotated for the presence or absence of 81 labels. Following [37] we augment the images with metadata using the Flickr API, discarding images for which metadata is unavailable. Following [15] we also discard images for which all labels are absent. This leaves 190,253 images, which we randomly partition into training, validation, and test sets of 110K, 40K, and 40,253 images respectively. We generate 5 such splits of the data and run all experiments on all splits. Statistics of the dataset can be found in Table 1. We will make our data and features publicly available to facilitate future comparisons.

NUS-WIDE	Labels	Tags	Sets	Groups
# unique elements	81	$10,000$	$165,039$	$95,358$
# image per (.)	5701.3 / 1682	270.3 / 91	2.3 / 1	26.1 / 2
# (.) per image	2.4 / 2	14.2 / 11	2.0 / 1	13.1 / 8

Table 1: Dataset statistics. Image and (.) counts are reported in the format mean / median.

4.2. Metrics

Prior work uses a variety of metrics and experimental setups on NUS-WIDE, making direct comparisons of results difficult. Following prior work [36, 18, 44, 15] we assign a fixed number of labels to each image and report (overall) precision $Prec_I$ and recall Rec_I; we also compute the precision and recall for each label and report the mean across labels as the *per-label* metrics $Prec_L, Rec_L$.

NUS-WIDE has a highly uneven distribution of labels; the most common (*sky*) has over 68,000 examples and the least common (*map*) has only 53. As a result the overall precision and recall statistics are strongly biased towards the common labels. The precision and recall for uncommon labels are extremely noisy since they are based on only a handful of test-set examples, and the mean per-label statistics inherit this noise since they weight all classes equally.

Mean Average Precision (mAP) is another widely used metric [37, 34]; it directly measures ranking quality, so it naturally handles multiple labels and does not require
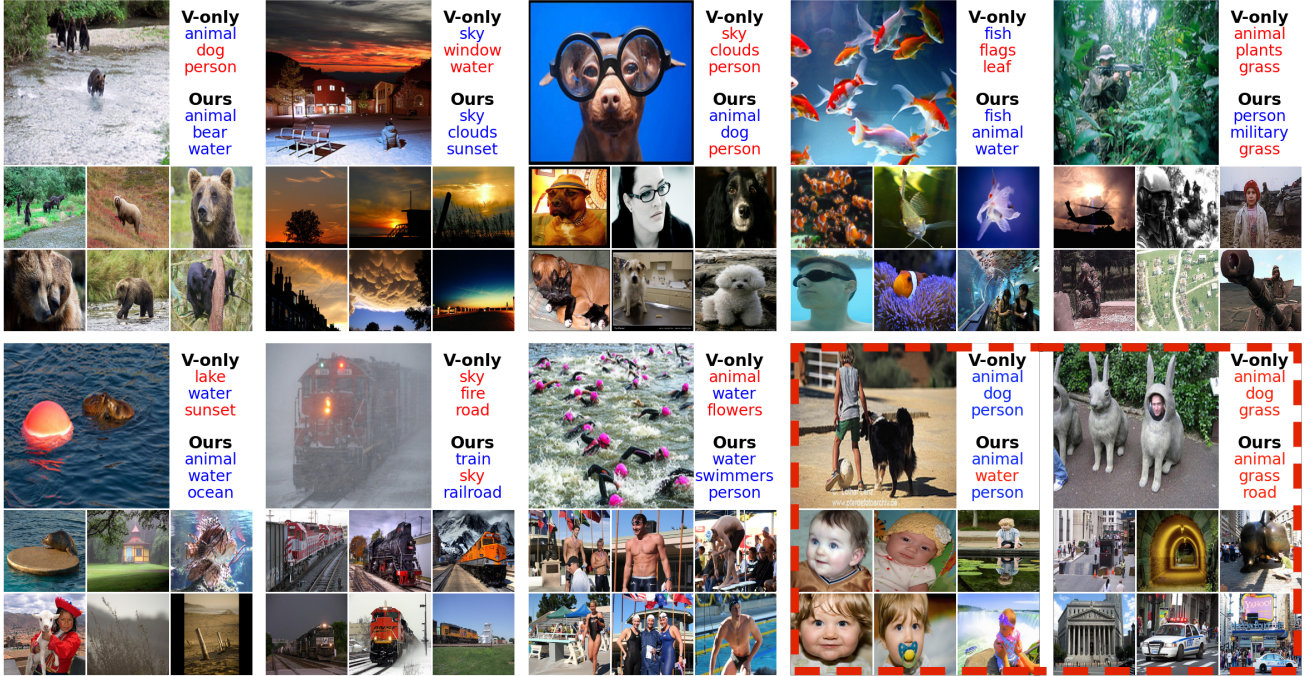
Figure 4: Example results. For each image we show the top 3 scoring labels using the visual-only (V-only) model and our model using tag nearest neighbors; correct labels are shown in blue and incorrect labels in red. We also show the 6 nearest neighbors to each image; its neighborhoods are drawn from these images. The red dashed lines show failure cases.

Method	mAP_L	mAP_I	Rec_L	$Prec_L$	Rec_I	$Prec_I$
Tag-only Model + linear SVM [37]	46.67	-	-	-	-	-
Graphical Model (all metadata) [37]	49.00	-	-	-	-	-
CNN + softmax [15]	-	-	31.22	31.68	59.52	47.82
CNN + ranking [15]	-	-	26.83	31.93	58.00	46.59
CNN + WARP [15]	-	-	35.60	31.65	60.49	48.59
Upper bound	$100.00_{\pm 0.00}$	$100.00_{\pm 0.00}$	$68.52_{\pm 0.35}$	$60.68_{\pm 1.32}$	$92.09_{\pm 0.10}$	$66.83_{\pm 0.12}$
Tag-only + logistic	$43.88_{\pm 0.32}$	$77.06_{\pm 0.14}$	$47.52_{\pm 2.59}$	$46.83_{\pm 0.89}$	$71.34_{\pm 0.16}$	$51.18_{\pm 0.16}$
CNN [27] + kNN-voting [36]	$44.03_{\pm 0.26}$	$73.72_{\pm 0.10}$	$30.83_{\pm 0.37}$	$44.41_{\pm 1.05}$	$68.06_{\pm 0.15}$	$49.49_{\pm 0.11}$
CNN [27] + logistic (visual-only)	$45.78_{\pm 0.18}$	$77.15_{\pm 0.11}$	$43.12_{\pm 0.39}$	$40.90_{\pm 0.39}$	$71.60_{\pm 0.19}$	$51.56_{\pm 0.11}$
Image neighborhoods + CNN-voting	$50.40_{\pm 0.23}$	$77.86_{\pm 0.15}$	$34.52_{\pm 0.47}$	$\mathbf{56.05}_{\pm 1.47}$	$72.12_{\pm 0.21}$	$51.91_{\pm 0.20}$
Our model: tag neighbors	$52.78_{\pm 0.34}$	$\mathbf{80.34}_{\pm 0.07}$	$43.61_{\pm 0.47}$	$46.98_{\pm 1.01}$	$74.72_{\pm 0.16}$	$\mathbf{53.69}_{\pm 0.13}$
Our model: tag neighbors + tag vector	$\mathbf{61.88}_{\pm 0.36}$	$80.27_{\pm 0.08}$	$\mathbf{57.30}_{\pm 0.44}$	$54.74_{\pm 0.63}$	$\mathbf{75.10}_{\pm 0.20}$	$53.46_{\pm 0.09}$

Table 2: Results on NUS-WIDE. Precision and recall are measured using $n = 3$ labels per image. Metrics are reported both per-label (mAP_L) and per-image (mAP_I). We run on 5 splits of the data and report mean and standard deviation.

choosing a fixed number of labels per image. As with other metrics, we report mAP both per-label (mAP_L) and per-image (mAP_I). mAP_L is less noisy and hence preferable to other per-label metrics since it considers the full ranking of images instead of only the top labels for each image.

5. Experiments

5.1. Multilabel Image Annotation

We show that our model achieves state-of-the art results for multilabel image annotation on NUS-WIDE. Our best model computes neighborhoods using tags with a vocabulary size of $\tau = 5000$, neighborhood size $m = 3$ and

max-rank $M = 6$. Preliminary experiments at combining all types of metadata did not show improvements over using tags alone. We also show the result of augmenting the hidden state of our model with a binary indicator vector of image tags. All results are shown in Table 2.

Baselines. First we report the results of McAuley and Leskovec [37] and Gong *et al.* [15] as in their original papers. Then we compare our model with four baselines:

1. Tag-only + logistic: the tag-only model of [37] represents each image with a sparse binary vector indicating its tags, while their full model uses all available metadata

(tags, groups, galleries, and sets) and incorporates a graphical model to model pairwise interactions between these features. Unfortunately these results are not directly comparable to ours, since they do not discard images without ground-truth labels; as a result they use 244K images for their experiments while we use only 190K. We reimplement a version of their tag-only model by training one-vs-all logistic classifiers on top of binary tag indicator features. Our reimplementation performs slightly worse than their reported numbers due to the difference in dataset size.

2. CNN + logistic loss: the results of [15] have been obtained using a deep convolutional neural networks in the style of [27] equipped with various multilabel loss functions. Again, these results are not directly comparable to ours because they train their networks from scratch on the NUS-WIDE dataset, while we use networks that were pretrained on ImageNet [9]. We reimplement a version of their model by training one-vs-all logistic classifiers using the features extracted from our pretrained network. This is an extremely strong baseline; note that it already outperforms [15], highlighting the power of the pretrained network.

3. CNN + kNN voting: as an additional baseline we implement a simple nearest neighbor approach. For each test image we compute the L_2 distance between its CNN features and the features of all images in the training set; the ground-truth labels of the retrieved training images are then used in a voting scheme similar to [36, 33].

4. Image neighborhoods + CNN-voting: for each test image we compute its M-nearest neighbors on the test set using user tags as in our full model, but instead of passing these neighbors to our parametric model we apply the CNN+logistic visual-only model to the image and its neighbors. Then we set the label scores of the test image to be a weighted sum of its visual-only label scores and the mean of the visual-only label scores of its neighbors.

Upper bound. As discussed in Section 4.2, we assign the top $n = 3$ labels to each image and report precision both per-class and per-image (recall that the average number of labels per image is approximately 2.4). However many images do not have exactly 3 ground-truth labels; this means that no classifier can achieve unit precision and recall. To estimate upper bounds for these metrics, we train one-vs-all logistic classifiers where each image its represented by a binary indicator vector encoding its ground-truth labels. As seen in Table 2, even this perfect classifier achieves far from perfect performance on many of the evaluation metrics.

Results. Table 2 shows that our model outperforms prior work on nearly all metrics. The per-class precision and recall metrics display high variance; as a result we do not believe them to be the best indicators of performance. The mAP metrics give a clearer picture of performance, since they display lower variance and do not rely on annotating

(a) Example PR curves
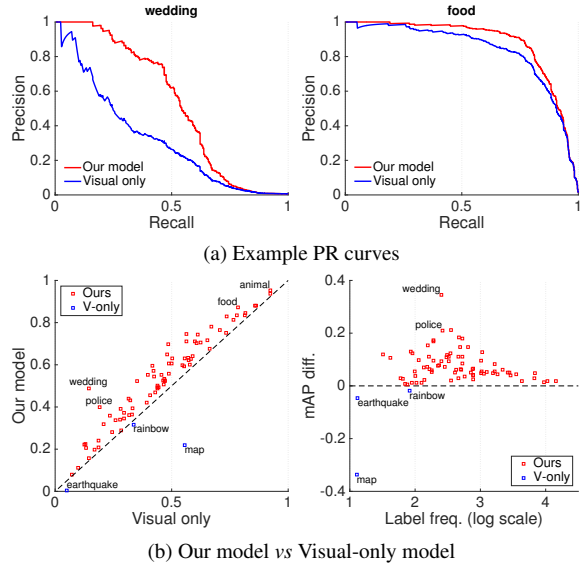

(b) Our model *vs* Visual-only model

Figure 5: (a) Our model shows large improvements for labels with high intra-class variability (e.g. *wedding*) and for labels where the visual model performs well (e.g. *food*). (b) Left: AP for each label of our model *vs* the baseline; we improve for all but three labels (*map*, *earthquake*, *rainbow*). (b) Right: difference in class AP between our model and the visual-only model vs label frequency.

each image with a fixed number of labels. On these metrics our model outperforms all baselines by a significant margin.

As an extension, we append the binary tag vector to the representation learned by our model (tag neighbors + tag vector); this does not significantly change performance as measured by per-image metrics, but does show improvement on per-class metrics. This suggests that the binary tag vector is especially useful for rare classes which may have strong correlations with certain user tags. Although it increases per-class performance, this extension significantly increases the number of learnable parameters and makes generalization to new types of metadata impossible.

In order to qualitatively understand some of the cases where our model outperforms the baselines, Figure 4 compares the top three labels produced by our model and by the visual-only baseline. The additional visual information provided by the neighborhoods can help resolve ambiguities in non-canonical views; for example in the image of swimmers the visual-only model appears to mistake the colorful swim caps for flowers, but the neighborhood provides canonical views of swimmers.

In few cases the neighborhood can hurt performance. For example in the image of the boy with a dog, the visual-only model correctly produces a *dog* label but our model replaces this with a *water* label, likely because no neighbors contain dogs but two neighbors contain visible bodies of water. However the aggregate metrics of Table 2 make it
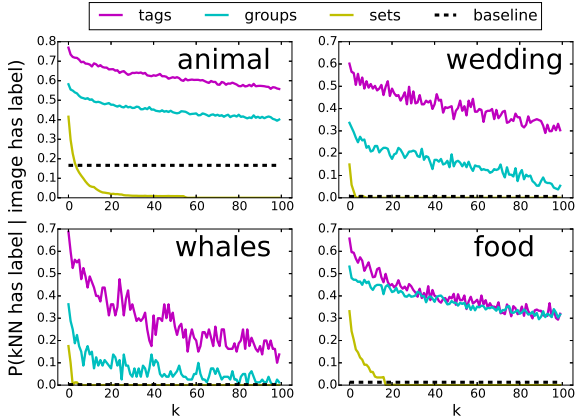
Figure 6: Probability that the kth nearest neighbor of an image has a particular label given that the image has the label, as a function of k and using different metadata. The dashed lines give the overall probability that an image has the label. Across all metadata and all classes, an image and its neighbors are likely to share labels.

clear that neighborhoods are beneficial more often than not.

There are cases where both models fail; for example see the lower right image of Figure 4 which shows a person crouching inside a statue of a rabbit. The ground-truth labels for this challenging image are *statue* and *person*, which are produced by neither model.

More quantitatively, Figure 5b compares the average precision (AP) of both our model and the visual-only baseline for each label; our model outperforms the baseline on all but three labels: *map*, *earthquake*, and *rainbow*. Of these, *map* is the only label where our model is significantly outperformed by the baseline. Figure 5b also reveals that these three labels are among the most infrequent; they have only 53, 56, and 397 instances respectively in the entire dataset, and an average of only 12.8, 13.2, and 82.0 instances respectively on the test sets. With so few test instances the performance of both models on these labels is highly susceptible to noise. It is also interesting to note that the middle frequencies are the ones in which our model gives the major boost in performance, while for the very frequent labels it is still able to give slight improvements.

Figure 5a also shows two example precision-recall curves. The *wedding* label has high intra-class variability, making it difficult to recognize using visual features alone; our model is able to give a large boost in performance by taking advantage of image metadata. Our model also gives improvements on labels such as *food* where the performance of the visual-only baseline is already quite strong.

5.2. Neighborhood Hyperparameters

Our method for generating image neighborhoods introduces several hyperparameters: the type of metadata used, the size m of each neighborhood, the max-rank M for

Method	mAP$_L$	mAP$_I$
CNN [27] + logistic (visual-only)	$45.78_{\pm 0.18}$	$77.15_{\pm 0.11}$
Our model: visual neighbors	$47.45_{\pm 0.19}$	$78.56_{\pm 0.14}$
Our model: group neighbors	$48.87_{\pm 0.22}$	$79.11_{\pm 0.13}$
Our model: set neighbors	$48.02_{\pm 0.33}$	$78.40_{\pm 0.25}$
Our model: tag neighbors	$\mathbf{52.78}_{\pm 0.34}$	$\mathbf{80.34}_{\pm 0.07}$

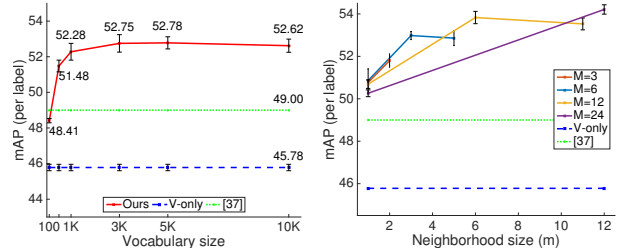Table 3: Our model trained with different image neighborhoods *vs* the visual-only model.

Figure 7: Performance of our model as we vary the neighborhood size m, max-rank M, and tag vocabulary size τ. In all cases our model outperforms the baselines.

neighbors, and the tag-vocabulary size τ. Here we explore the influence of these hyperparameters on our model.

Effects on performance. The most important hyperparameter for generating neighborhoods is the type of data used. We show in Table 3 the performance of our model using different types of metadata: tags give the highest performance, followed by groups and then sets. In all cases our model outperforms the visual-only baseline. We also show the effect of using Euclidean distance of visual features to build neighborhoods (visual neighbors). This setup slightly outperforms the visual-only baseline but is outperformed when using metadata, showing both the ability of our method to handle a variety of neighbor types, and the importance of image metadata.

To study the effects of the neighborhood size m, the max-rank M, and the tag vocabulary size τ we show in Figure 7 the performance of our model as we vary these hyperparameters. Varying the max-rank M gives the largest variation in performance, but in all cases we show improvements over the visual-only baseline and the results of [37].

Label correlations. We can better interpret the influence of neighborhood hyperparameters by studying the correlations between the labels of images and their nearest neighbors. With strong correlations, visual evidence for a label among an image's neighbors is evidence that the image should have the same label; as such, our model should perform better when these correlations are stronger.

To this end, we plot in Figure 6 the probability that the kth nearest neighbor of an image has a particular label given that the image itself has the label; on the same axis we show

the baseline probability that a random image in the dataset has the label. This experiment shows that the nearest neighbors of images are indeed very likely to share labels with an image, and helps to explain the influence of various hyperparameters. An image's labels are most highly correlated with its tag neighbors, followed by groups and then sets; this matches the results of Table 3. The flat shape of all curves in Figure 6 suggests that the 20th nearest neighbor is nearly as informative as the 10th, suggesting that larger max-ranks M may increase performance.

5.3. Generalization Experiments

One advantage of our model is that we only use metadata of images nonparametrically as a means to compute image neighborhoods. As a result, our model can easily cope with situations where different types of metadata are available during training and testing.

Vocabulary Generalization. Our best-performing model relies on user tags to generate image neighborhoods. In a real-world setting, the vocabulary of user tags may change over time: new tags may become popular, and older tags may fall into disuse. Any method that depends on user tags should be able to cope with these challenges.

Ideally, to test our model's resilience to changes in user tags over time, we would train the model using a snapshot of Flickr images at one point in time and test the model using a snapshot from a different point in time.

Unfortunately we do not have access to this type of data. As a proxy to such an experiment, we instead randomly divide the 10K most commonly occurring user tags into two sets. During training we use the first set of user tags to generate neighborhoods, and use the second during testing. We vary the degree to which the training tags and the testing tags overlap; with an overlap of 0% there are no tags shared between training and testing, and an overlap of 100% uses the same vocabulary of user tags for training and testing. Results are shown in Figure 8.

We see that the performance of our model degrades as we decrease the overlap between the training and testing tags; however even in the case of 0% overlap our model is able to outperform both the visual-only model and [37].

Metadata Generalization. As a test of our model's ability to generalize across different types of metadata, we perform an experiment where we use different types of metadata during training and testing. For example, we generate neighborhoods with tags during training and instead use sets during testing. Table 4 shows the quantitative results of this experiment; in all cases our model outperforms the visual-only baseline. These results suggest that our model could be applied in cases where some types of metadata are unavailable during testing.
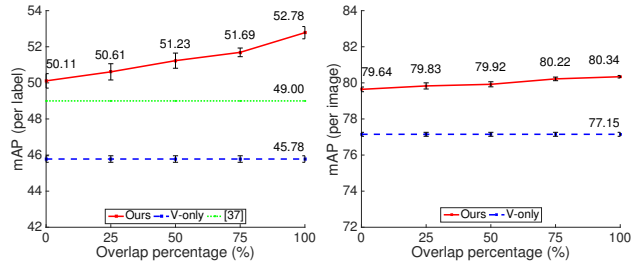

Figure 8: Performance as we vary overlap between tag vocabularies used for training and testing. Our model gives strong results even in the case of disjoint vocabularies.

Train: \ Test:	Tags	Sets	Groups
Tags	$52.78_{\pm 0.34}$	$47.12_{\pm 0.35}$	$48.14_{\pm 0.33}$
Sets	$52.21_{\pm 0.29}$	$48.02_{\pm 0.33}$	$48.49_{\pm 0.16}$
Groups	$50.32_{\pm 0.28}$	$47.82_{\pm 0.24}$	$48.87_{\pm 0.22}$

Table 4: Metadata generalization experiment. We use different types of metadata during training and testing, and report mAP_L for all possible pairs. All combinations outperform the visual-only model ($45.78_{\pm 0.34}$).

We can explain the results of this experiment by again examining Figure 6. When we train using one signal and test using another, our train and test data are no longer drawn from the same distribution, breaking one of the core assumptions of supervised learning. However the parametric portion of our model only views image metadata through the lens of nearest neighbors; Figure 6 shows that changing the method of computing these neighbors does not drastically change the nature of the correlations between the labels of an image and its neighbors.

6. Conclusion

We have introduced a framework that exploits image metadata to generate neighborhoods of images, and uses a strong parametric visual model based on deep convolutional neural networks to blend visual information between an image and its neighbors. We use our model to achieve state-of-the-art performance for multilabel image annotation on the NUS-WIDE dataset. We also show that our model gives impressive results even when it is forced to generalize to new types of metadata at test time.

References

[1] L. Ballan, T. Uricchio, L. Seidenari, and A. Del Bimbo. A cross-media model for automatic image annotation. In *ICMR*, 2014.

[2] K. Barnard, P. Duygulu, D. Forsyth, N. De Freitas, D. M. Blei, and M. I. Jordan. Matching words and pictures. *JMLR*, 3:1107–1135, 2003.

[3] A. Bergamo and L. Torresani. Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach. In *NIPS*, 2010.

[4] G. Carneiro, A. B. Chan, P. J. Moreno, and N. Vasconcelos. Supervised learning of semantic classes for image annotation and retrieval. *IEEE TPAMI*, 29(3):394–410, 2007.

[5] L. Chen, D. Xu, I. W. Tsang, and J. Luo. Tag-based web photo retrieval improved by batch mode re-tagging. In *CVPR*, 2010.

[6] X. Chen and A. Gupta. Webly supervised learning of convolutional networks. In *ICCV*, 2015.

[7] X. Chen, A. Shrivastava, and A. Gupta. NEIL: Extracting visual knowledge from web data. In *ICCV*, 2013.

[8] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. NUS-WIDE: a real-world web image database from National University of Singapore. In *CIVR*, 2009.

[9] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A large-scale hierarchical image database. In *CVPR*, 2009.

[10] S. K. Divvala, A. Farhadi, and C. Guestrin. Learning everything about anything: Webly-supervised visual concept learning. In *CVPR*, 2014.

[11] K. Duan, D. Crandall, and D. Batra. Multimodal learning in loosely-organized web images. In *CVPR*, 2014.

[12] C. Fang, H. Jin, J. Yang, and Z. Lin. Collaborative feature learning from social media. In *CVPR*, 2015.

[13] Z. Feng, S. Feng, R. Jin, and A. K. Jain. Image tag completion by noisy matrix recovery. In *ECCV*, 2014.

[14] R. Fergus, L. Fei-Fei, P. Perona, and A. Zisserman. Learning object categories from google's image search. In *ICCV*, 2005.

[15] Y. Gong, Y. Jia, T. K. Leung, A. Toshev, and S. Ioffe. Deep convolutional ranking for multilabel image annotation. In *ICLR*, 2014.

[16] Y. Gong, Q. Ke, M. Isard, and S. Lazebnik. A multi-view embedding space for internet images, tags, and their semantics. *IJCV*, 106(2):210–233, 2014.

[17] D. Grangier and S. Bengio. A discriminative kernel-based approach to rank images from text queries. *IEEE TPAMI*, 30(8):1371–1384, 2008.

[18] M. Guillaumin, T. Mensink, J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, 2009.

[19] M. Guillaumin, J. Verbeek, and C. Schmid. Multimodal semi-supervised learning for image classification. In *CVPR*, 2010.

[20] J. Hays and A. A. Efros. IM2GPS: estimating geographic information from a single image. In *CVPR*, 2008.

[21] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

[22] G. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[23] S. J. Hwang and K. Grauman. Learning the relative importance of objects from tagged images for retrieval and cross-modal search. *IJCV*, 100(2):134–153, 2012.

[24] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell. Caffe: Convolutional architecture for fast feature embedding. In *MM*, 2014.

[25] M. M. Kalayeh, H. Idrees, and M. Shah. NMF-KNN: Image annotation using weighted multi-view non-negative matrix factorization. In *CVPR*, 2014.

[26] G. Kim, E. P. Xing, and A. Torralba. Modeling and analysis of dynamic behaviors of web image collections. In *ECCV*, 2010.

[27] A. Krizhevsky, I. Sutskever, and G. Hinton. ImageNet classification using deep convolutional neural networks. In *NIPS*, 2012.

[28] N. Kumar and S. Seitz. Photo recall: Using the internet to label your photos. In *CVPR Workshops*, 2014.

[29] T. Lan and G. Mori. A max-margin riffled independence model for image tag ranking. In *CVPR*, 2013.

[30] V. Lavrenko, R. Manmatha, and J. Jeon. A model for learning the semantics of pictures. In *NIPS*, 2003.

[31] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proc. of the IEEE*, 86(11):2278–2324, 1998.

[32] L.-J. Li and L. Fei-Fei. OPTIMOL: Automatic online picture collection via incremental model learning. *IJCV*, 88(2):147–168, 2010.

[33] X. Li, C. G. M. Snoek, and M. Worring. Learning social tag relevance by neighbor voting. *IEEE TMM*, 11(7):1310–1322, 2009.

[34] X. Li, T. Uricchio, L. Ballan, M. Bertini, C. G. M. Snoek, and A. Del Bimbo. Socializing the semantic gap: A comparative survey on image tag assignment, refinement and retrieval. *arXiv preprint*, arXiv:1503.08248, 2015.

[35] Y. Li, D. Crandall, and D. Huttenlocker. Landmark classification in large-scale image collections. In *ICCV*, 2009.

[36] A. Makadia, V. Pavlovic, and S. Kumar. A new baseline for image annotation. In *ECCV*, 2008.

[37] J. McAuley and J. Leskovec. Image labeling on a network: using social-network metadata for image classification. In *ECCV*, 2012.

[38] Z. Niu, G. Hua, X. Gao, and Q. Tian. Semi-supervised relational topic model for weakly annotated image recognition in social media. In *CVPR*, 2014.

[39] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015.

[40] N. Srivastava and R. Salakhutdinov. Multimodal learning with deep boltzmann machines. In *NIPS*, 2012.

[41] Z. Stone, T. Zickler, and T. Darrell. Autotagging facebook: Social network context improves photo annotation. In *CVPR Workshops*, 2008.

[42] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. *Coursera: Neural Networks for Machine Learning*, 2012.

[43] D. Tsai, Y. Jing, Y. Liu, H. A. Rowley, S. Ioffe, and J. M. Rehg. Large-scale image annotation using visual synset. In *ICCV*, 2011.

[44] Y. Verma and C. Jawahar. Image annotation using metric learning in semantic neighbourhoods. In *ECCV*, 2012.

[45] G. Wang, D. Hoiem, and D. Forsyth. Learning image similarity from flickr groups using stochastic intersection kernel machines. In *ICCV*, 2009.

[46] J. Weston, S. Bengio, and N. Usunier. Wsabie: Scaling up to large vocabulary image annotation. In *IJCAI*, 2011.

[47] A. Yu and K. Grauman. Predicting useful neighborhoods for lazy local learning. In *NIPS*, 2014.

[48] A. Zamir, S. Ardeshir, and M. Shah. Gps-tag refinement using random walks with an adaptive damping factor. In *CVPR*, 2014.

[49] S. Zhang, J. Huang, Y. Huang, Y. Yu, H. Li, and D. N. Metexas. Automatic image annotation using group sparsity. In *CVPR*, 2010.

[50] G. Zhu, S. Yan, and Y. Ma. Image tag refinement towards low-rank, content-tag prior and error sparsity. In *MM*, 2010.