Multi-Attribute Queries: To Merge or Not to Merge?

Mohammad Rastegari
University of Maryland
mrastega@cs.umd.edu

Ali Diba
Sharif University of Technology
diba@ce.sharif.edu

Devi Parikh
Virginia Tech
parikh@vt.edu

Ali Farhadi
University of Washington
ali@cs.uw.edu

Abstract

Users often have very specific visual content in mind that they are searching for. The most natural way to communicate this content to an image search engine is to use keywords that specify various properties or attributes of the content. A naive way of dealing with such multi-attribute queries is the following: train a classifier for each attribute independently, and then combine their scores on images to judge their fit to the query. We argue that this may not be the most effective or efficient approach. Conjunctions of attribute often correspond to very characteristic appearances. It would thus be beneficial to train classifiers that detect these conjunctions as a whole. But not all conjunctions result in such tight appearance clusters. So given a multi-attribute query, which conjunctions should we model? An exhaustive evaluation of all possible conjunctions would be time consuming. Hence we propose an optimization approach that identifies beneficial conjunctions without *explicitly training the corresponding classifier. It reasons about geometric quantities that capture notions similar to intra- and inter-class variances. We exploit a discriminative binary space to compute these geometric quantities efficiently. Experimental results on two challenging datasets of objects and birds show that our proposed approach can improve performance significantly over several strong baselines, while being an order of magnitude faster than exhaustively searching through all possible conjunctions.*

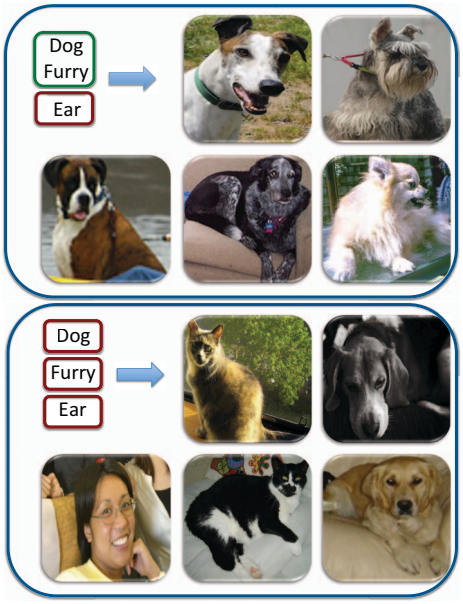

Figure 1. In a multi-attribute image search, some combinations of attributes can be learned jointly, resulting in a better classifier. In this paper, we propose a model to predict which combinations will result in a better classifier without having to train a classifier for all possible cases. For example, when looking for dog, furry, ear, our method selects to train a furry-dog classifier and fuse it with an ear classifier. We compare this selection with the default case where one classifier is trained per attribute. Here we show top five retrieved images.

1. Introduction

We often find ourselves searching for images with very specific visual content. For instance, if we witness a crime we might help law enforcement agents search through mugshots of criminals to find the specific individual we saw. Victims of disasters may search through hospital databases to find missing loved ones. Graphic designers may search for illustrations of specific styles. Bird watchers may search for photographs of birds with a particular appearance to identify its species. In such scenarios, the most natural way for users to communicate their target visual content is to describe it in terms of its attributes [3, 7] or visual properties. Given the specificity of the desired content, the user typically needs to specify multiple attributes in order to appropriately narrow the search results down.

A common way of dealing with such multi-attribute queries is to train classifiers for each of the attributes in-

1063-6919/13 $26.00 © 2013 IEEE
DOI 10.1109/CVPR.2013.425

dividually and combine their scores to identify images that satisfy all specified attributes. If a user is interested in images of white furry dogs, one would run three classifiers and combine them (white & furry & dog) to indirectly get a white-furry-dog classifier. However this may not be the most effective or most efficient solution. White furry dogs may have a very characteristic easy-to-detect appearance, and running just one white-furry-dog classifier trained to directly detect only white furry dogs could result in more accurate and faster results. But there may not be enough white furry dog examples to train such a classifier. Or, white furry dogs may look a lot like the rest of the dogs leading to a harder classification problem and poorer performance than combining three independent classifiers. Given a multi-attribute query such as white furry dog, it is critical to determine which combinations of classifiers should be trained to ensure effective and efficient retrieval results: white-furry & dog, or white-furry-dog, or white & furry & dog, etc. This is the problem we address in this paper.

An exhaustive solution to this problem would involve training all possible combinations of the multiple attributes involved (5 combination in the case of white furry dogs), and evaluating their accuracy on a held out set of images to determine the optimal combination. This would be computationally expensive especially as the number of attributes in the query grows, and requires sufficient amount of validation data. In this paper we propose an optimization approach that given a multi-attribute query, efficiently identifies which components would be beneficial *i.e*. which attributes should be merged, without having to enumerate and train all possible combinations. We use the intuition that geometric notions that capture the compactness ($\sim$intra-class variance) of the set of images that satisfy a combination (*e.g*. white-dog), and the margin of these images from other distractor images ($\sim$inter-class variance) provide good proxies for the likely effectiveness of a classifier trained to recognize the combination. We show that these geometric quantities can be evaluated efficiently in a discriminative binary space. We evaluate our algorithm on aPascal and Bird200 datasets and show that our method can find combinations that are both more accurate and faster than independent classifiers.

2. Related Work

We now describe the connections of our work to existing work on dealing with multi-attribute queries and visual phrases. We also briefly mention other uses of binary spaces in literature.

Multi-attribute queries: Attributes or semantic concepts are often used for improved multimedia retrieval [17, 10, 9, 22, 1, 19, 14]. Fewer works have looked at the challenges that arise in multi-attribute queries in particular. Siddiquie *et al*. [16] model the natural correlation between attributes to improve search results. For instance, if a face has a mustache then it is likely to be a male face. Scheirer *et al*. [15] recently proposed a novel calibration method to more effectively combine scores of independent multiple attribute classifiers. Our work is orthogonal to these efforts. We are interested in identifying which attributes should be merged to then train a classifier directly for the conjunction for improved search results. Note that we identify beneficial conjunctions for each given multi-attribute query, and do not reason about global statistics of pre-trained attribute classifiers.

Visual phrases: The attribute combinations we reason about can be thought of as being analogous to the notion of visual phrases introduced by Sadeghi *et al*. [12]. They showed that some *object* combinations correspond to a very characteristic appearance that makes detecting them as one entity much easier. For instance, one can detect a person riding a horse more accurately if modeled as one entity, than detecting the person and horse independently and then combining their responses. They used a pre-defined vocabulary of visual phrases. Our work is distinct in that it deals with attribute combinations rather than object compositions. More importantly, the goal of our work is to identify which combinations should be trained on a *per query basis*. Li *et al*. [8] proposed an approach to identify which groups of objects should be modeled together. They reason about consistent spatial arrangements of objects in images. This would be analogous to reasoning about ground truth attribute co-occurrence patterns when dealing with multi-attribute queries. In contrast, in our work we explicitly reason about the variation in appearances of images under the different attribute combinations. As a result, the combinations we identify are grounded to the appearance features of images, which significantly affect the accuracy of resultant classifiers.

Binary spaces: There has been significant progress in recent years in mapping images to binary spaces. One might learn a mapping that preserves correlations between semantic similarities and binary codes [13], or local similarities [4, 20, 5]. Recently, discriminative binary codes have shown promising results in mapping images to a binary space where linear classifiers can perform even better than sophisticated models [11]. We use this mapping to project images to a binary space where computing simple geometric measures like compactness or diameters of a group of images and their margins from other images is very efficient.

3. Our Model

Given a multi-attribute query, our goal is to figure out which combinations of attributes would be better to use without having to train classifiers for all possible combinations. What makes a combination desirable? The most important criteria is the *learnability* of a combination. In other words, we should learn a classifier for a combination of two attributes if it results in a better classifier for the conjunction than combining scores of independent attribute classi-

fiers post-training. For three attributes like white and furry and dog[1], a combination can include multiple components like white and furry-dog. We argue that geometric reasoning in terms of the tightness and margin of each component in a combination is a reasonable proxy for what would have happened if we would have trained a classifier for each component in the combination. Geometrically speaking, a good combination should have components that occupy tight regions of the feature space and have large margins. Figure 2 shows an illustration where purple instances are the ones that have both blue and red attributes. What justifies learning a red-blue classifier instead of red and blue classifiers independently is that purple instances occupy a tight area in the feature space with big margins from other blue and red instances. If it was not the case, then we could have learned separate red and blue classifiers; they are more widely applicable and would not sacrifice training data. To efficiently compute these geometric measurements we propose to map the images from the original feature space into a binary space where discriminative properties are preserved. In this section we assume that such a mapping exists. Later in the experiments we show that our formulation is not very sensitive to the choice of the mapping as long as discriminative properties are preserved/enhanced in the binary space. This is not a restrictive condition as most existing binary mapping approaches in literature meet this criteria.

We estimate the learnability of a combination based on the diameter of the components in the combination and the margin within and across components. To setup notations, let's assume there are n attributes involved in a given multi-attribute query, $A = \{a_1, ..., a_n\}$. For example, {white,furry,dog}. There are 2^n different ways to form components. For instance, {white}, {furry,dog}, {white,dog}, {furry}, etc. The set of all possible components is the powerset of A, which we call $\mathcal{S} = \{S_1, S_2, \cdot, S_m\}, m = 2^n$. A combination is a subset of $\mathcal{S}$ that covers A *e.g.* {{white,furry}, {dog}}, which we write as {white-furry,dog} in shorthand. We define the learnability of a combination $\mathcal{C}$ as

$$\mathcal{L}(\mathcal{C}) = \sum_{c \in \mathcal{C}} [\sum_{c' \in \mathcal{C}, c' \neq c} \mathcal{K}(c, c') + \sum_{a \in c} \mathcal{K}(c, c \setminus a) - \mathcal{D}(c)]$$

where c indexes components in the combination $\mathcal{C}$, a indexes attributes in each component, $\mathcal{D}(c)$ is the diameter of each component defined as $\max_{x,y \in c} d(x, y)$ where x and y are images that belong to a component and d is the distance between them. The diameter captures the range of visual appearances of images within a component. The higher the variety of appearances, the less learnable the corresponding component. $\mathcal{K}(c, c')$ is the margin between two components c and c' defined as $\min_{x \in c, y \in c'} d(x, y)$. This captures how distant the images belonging a component are from images of other components. The more distant they are, the easier it is to learn a classifier for the component. Finally, $\mathcal{K}(c, c \setminus a)$ is the margin between images that satisfy all attributes of a component, and those that satisfy all but one attribute. For example the margin between purple and red in Figure 2. For components that consist of only one attributes the within component margins are zero.

[1]For generality of discussion, we treat all words involved in a query as "attributes"

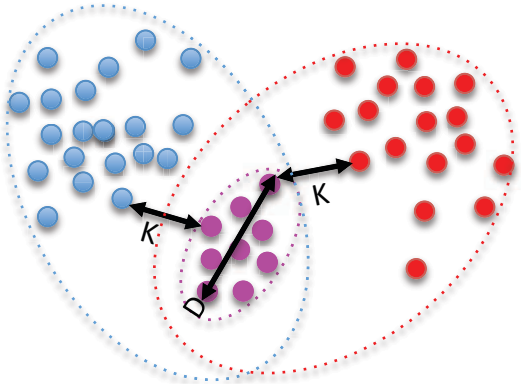

Figure 2. What makes merging two attributes desirable? When instances that satisfy both attributes occupy a tight region in the feature space and have enough margin to the instances that have one of the attributes. This figure depicts a case where training a merged red-blue classifier is beneficial. Because purple dots (instances that have both red and blue attributes) have small diameter (D) and enough margins (K) with the rest of blue and red dots.

We are interested in finding the optimal combination $\mathcal{C}^*$ that obtains best learnability score and covers all members of A without being inefficiently redundant . We can formulate this problem as the following integer program:

$$\begin{aligned} \max_{x} \quad & \mathcal{L}(\mathcal{S} \odot x) - \lambda |x| \\ & Z^T x \geq \mathbf{1} \\ & x \in \{0, 1\}^m \end{aligned} \tag{1}$$

where $\odot$ is the set selection operator, Z is an $m \times n$ binary set system matrix indicating which attributes appear in which component, λ is the trade off factor between the number of components in a combination (efficiency) and the learnability score, and x is the indicator vector that identifies which components will make it to the final combination.

Set covering problem can be reduced to our problem. The optimization 1 is harder than standard weighted set covering problem because our learnability function $\mathcal{L}$ defines over all component in a combination. The corresponding weighted set cover formulation requires the weighting function to be defined over each component independently. The interdependencies between components in our learnability function make this optimization NP-hard. However, our learnability function doesn't face an interdependency issue in case of two attributes. This suggests defining a gain func-

tion for pairs of attributes that takes into account the same measurements (diameter and margins) as in our learnability function:

$$\mathcal{G}(a_i, a_j) = \mathcal{K}(a_i a_j, a_i) + \mathcal{K}(a_i a_j, a_j) - \mathcal{D}(a_i a_j)$$

Given two attributes, positive values for the gain function recommend merging the two attributes and negative values encourage training separate classifiers for each attribute and then merging their scores. The higher the gain function the higher is the reward for merging two attributes. Our gain function exposes an interesting property that helps prune the search space drastically.

Lemma 1. If attributes a_i and a_j are merged because $\mathcal{G}(a_i, a_j) \geq 0$ then for any other attribute a_k, $\mathcal{G}(a_i a_j, a_k) \geq \mathcal{G}(a_i, a_k)$ or $\mathcal{G}(a_j, a_k)$

Proof. It's simple to show that if $A \subset B$ then $\mathcal{D}(A) \leq \mathcal{D}(B)$, and if $C \subset D$ then $\mathcal{K}(A, C) \geq \mathcal{K}(B, D)$. We can show that $\mathcal{G}(a_i a_j, a_k) = \mathcal{K}(a_i a_j a_k, a_i a_j) + \mathcal{K}(a_i a_j a_k, a_k) - \mathcal{D}(a_i a_j a_k) + > \mathcal{K}(a_i a_j a_k, a_i a_j) + \mathcal{K}(a_i a_j a_k, a_k) - \mathcal{D}(a_i a_k) + > \mathcal{K}(a_i a_k, a_i) + \mathcal{K}(a_i a_k, a_k) - \mathcal{D}(a_i a_k) + = \mathcal{G}(a_i, a_k)$. The same holds for $\mathcal{G}(a_i, a_j)$. □

What this lemma implies is that once two attributes are merged, we need not consider merging any other attribute with either of these attributes individually. This suggests the following recursive greedy solution to find the highest scoring and covering combination.

Our greedy solution starts with computing the gain for all pairs of attributes. It picks the pair with the highest gain. If the highest gain is positive, then we merge those attributes and add a new merged-attribute to our set of attributes and remove the two independent ones. Meaning that if a_i and a_j provide the biggest positive gain we add $a_i a_j$ as a new attribute to A and remove a_i and a_j from the set. The Lemma above shows that it is safe to remove the independent attribute from the set as no other attribute can join either of a_i or a_j independently and result in higher scoring combination. The new A now has $n-1$ elements. We can recursively repeat this procedure till we cover all attributes. If there is no pair with positive gain, we move to triplets. This never happened in our experiments.

Efficient Computation of Geometric Measurements: Margins and diameters can be computed efficiently in a binary feature space; $O(NK)$ where N is the number of images and k is the dimensionality of bit vectors. The core part for computing both margin and diameter is to compute the average of all pairwise distances. A naive algorithm would be to go over all pairs and compute their distances and get mean of them. But since we are using binary codes for each dimension of the binary codes we can compute number of **zero** bits and number of **one** bits. Then the sum of the distance of any given bit to all other bits can be computed in $O(constant)$. Algorithm 1 explains this algorithm more formally.

Algorithm 1 Efficient Sum of Pairwise Hamming Distances

Input: $B1$, $B2$ are a binary matrix of size $N \times K$.
Output: S: sum of hamming distances between all pairs of rows in $B1$ and $B2$.

for $k = 1 \rightarrow K$ **do**
  $Z(k) \leftarrow \sum_k B2(:, k)$ **Comment:** Counting Number of zeros in k^{th} dimension of $B2$
  $O(k) \leftarrow \sum_k \neg B2(:, k)$ **Comment:** Counting Number of ones in k^{th} dimension of $B2$
end for
for $i = 1 \rightarrow N$ **do**
  for $k = 1 \rightarrow K$ **do**
    if $B1(i, j) = 0$ **then**
      $P(i, j) \leftarrow O(k)$
    else
      $P(i, j) \leftarrow Z(k)$
    end if
  end for
end for
$S \leftarrow \sum P$ **Comment:** Sum of all elements in P

4. Experimental Results

We evaluate our method in several different settings. We conduct experiments on two challenging datasets: the aPasclal [3] and the Caltech Bird200 dataset [18]. We compare our method with four different baselines described later. We also test our method with different binary code mapping methods and show that our method is robust to the choice of binary mapping. We also evaluate the impact of different binary code sizes on the performance of our approach. In addition to accuracy, we also compare the running time of our method to that of baselines. We find that our low complexity $O(NK)$ gives us one order of magnitude speed up. We also present qualitative results and analysis that reveal the tendencies of different attributes to merge with other attributes.

4.1. Datasets

aPASCAL [3]: This dataset contains the 20 PASCAL object categories. On average each category has 317 images. Each image is labeled by 64 attributes that describe different object properties such having a particular body part, types of materials, etc. We experiment with the low-level features provided by the author of [3] on the data set website and also train/test splits provided with the dataset. The features and attribute annotations are not labeled for entire image. They are computed only for bounding box of the objects.

Caltech-UCSD Bird200 [21] This data set is a challenging subordinate recognition dataset. It includes 200 different species of North American birds with on average 300 images per category. Each image is annotated with 312 bird attributes such as color and shapes of wings, beaks, etc. We used the low-level features provided by [2] describing color, shape and contours. Similar to aPascal, here, we don't use entire image, we ony use the area that the bounding box of the image specifies for a bird in that image. We devide each

category in half and took one haf as train set and the other half as test set.

4.2. Baseline Methods

We compare our method with four different baseline approaches for selecting the combinations to be trained for a given multi-attribute query: **Default (DEF):** As the name suggestions, this approach uses the most natural strategy of training classifiers for each of the attribute independently and then combining the result scores. **Random Selection (RND):** This approach randomly selects a combination from all possible combinations and learns a classifier for each component of that combination. **Upper Bound (UPD):** Here we exhaustively train all possible combinations, evaluate their performance *on the test set*, and select the best one. The resultant performance corresponds to the upper bound one can hope to achieve by picking the optimal combinations to train. Of course, our proposed approach avoids training all possible combinations, and selects a good combination very efficiently. A comparison to this upper bound informs us of the resultant loss in performance by trading it off for efficiency. **Best Attribute First (BAF):** Intuitively, if an attribute predictor is accurate enough (in the limit, perfect), there is no benefit to merging it with another attribute. This baseline is based on this intuition. It determines which attributes to merge by looking at their prediction accuracies on the test set. Attributes with an accuracy higher than a threshold are left alone, while the rest are merged. We search for a threshold that gives us highest overall accuracy on all the queries.

4.3. Evaluation

Having identified the best combination (*e.g.* {white-furry,dog}), we train a classifier for each of the components {white,furry} and {dog} using (with C = 1). All training images that are both white and furry are positive examples to train a white-furry component classifier, and all remaining images are negative examples. Given a test image, we compute its score for each of the component classifiers. A naive way of combining these component classifiers would be to threshold the scores and compute a logical-AND. However in practice, the scores of the different classifiers are not calibrated. We use [6] to calibrate the scores, which fits a weibull distribution to the scores of a classifier to generate probability estimates. We later show the benefits of this calibration. We threshold the calibrated probabilities and compute the logical-AND to determine if a test image is positive (relevant to the multi-attribute query) or not. Varying the threshold gives us a precision-recall curve. One might argue that by taking the product of the calibrated scores and then thresholding that we may get better performance. But in our experments it drops the performace remarkably. In order to report results across multiple queries, we average the recall across all queries for fixed precision values to obtain an "average" precision-recall curve.

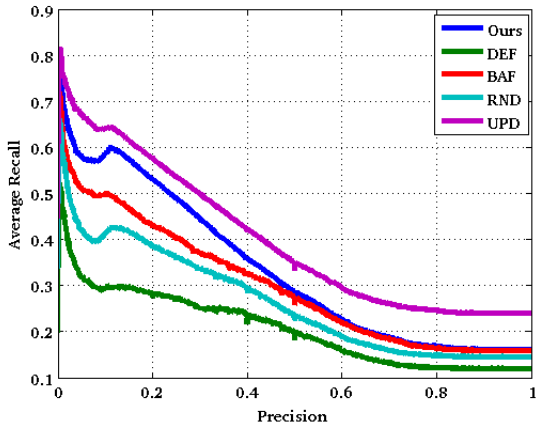

Figure 3. We evaluate our method on retrieving images in aPascal test set using 3-attribute queries. We compare it with three baselines and also the best possible upper bound. We use 512-dimensional bit codes for this experiment. Each point in this plot corresponds to average recalls over selected combinations on several fixed precisions. The threshold for BAF is 0.7.

Comparison with Baselines: We generated 500 random 3-attribute queries that had atleast 100 corresponding images in the train and test splits. We also generated another set of 500 3-attribute queries that had between 5 and 50 examples in the train and test splits. This allows us to evaluate our approach on queries with sufficient as well as few examples. Figure 3 shows our results for the aPascal. We see that our method outperforms all baselines, and is not significantly worse than the upper-bound, especially at high recall. For these experiments we used 512 bits codes extracted using Discriminative Binary Codes [11]. Figure 4 shows results using 4-attribute queries, with similar trends. Figure 5 shows our results on the Birds dataset with queries of length 3. Our method outperforms the baselines by large margin. The effects of different parts in learnability function at 0.2 precision is as follow: Recall .15 .45 .61. $\mathcal{K}(c, c')$: 102 170 213. $\mathcal{K}(c, c \setminus a)$: 23 56 79. $\mathcal{D}(c)$: 162 106 62. Increase in the margin and decrease in the diameter results in better recall.

Binary Code Length: We now investigate the effect of different length of binary codes on the performance of our method. Figure 6 shows results aPascal using the same length 3 queries described earlier. Using fewer bits hurts performance. Figure 7 shows similar trends on the Birds dataset.

Sensitivity to Binary Mapping Methods: We now evaluate our model using binary codes generated by different methods. We chose two state-of-the-art binary mapping methods DBC [11] and ITQ [5] and also classical LSH [4]. Table 1 compares the performance of our approach using these three methods on the aPascal dataset. Here we use mean of the average recalls over all fixed precisions (MAR)

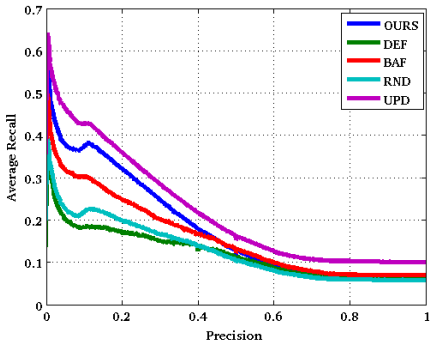

Figure 4. We evaluate our method on retrieving images in aPascal test set using 4-attribute queries. Experimental setup is similar to that of Figure 3. The threshold for BAF is 0.82 .

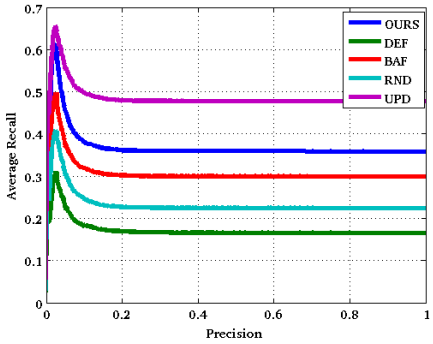

Figure 5. We evaluate our method on retrieving images in Bird test set using 3-attribute queries. Experimental setup is similar to that of Figure 3.

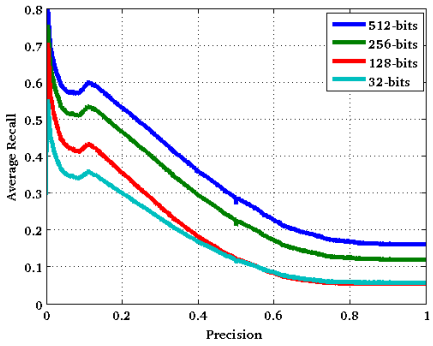

Figure 6. We investigate the effects of the dimensionality of binary space on our performance on the aPascal dataset.

as a measure for comparison. We used 512 bits for all of the methods. DBC perform slightly better because DBC preserves categorical similarities between images. We trained DBC on the whole train set of aPascal dataset. To make the most of ITQ we used the attribute labels of the train set to learn ITQ coupled with CCA. The binary codes produced

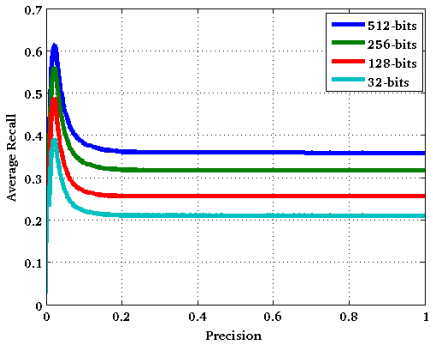

Figure 7. We investigate the effects of the dimensionality of binary space on our performance on the Bird dataset.

Method	MAR
Upper Bound	0.4007
DBC-512bits	0.3348
ITQ-CCA-512bits	0.3257
LSH-512bits	0.3071

Table 1. Comparison between different binary mapping methods in terms of Mean Average Recall.

by ITQ-CCA are expected to preserve pairwise similarities. For both cases we use their publicly available MATLAB code. Our model is not sensitive to the choice of binary mapping (compare DBC and ITQ) as long as discriminative properties can be preserved.

Running Time Evaluation: Here we report the run time of our approach. First, we only consider the average time required to find the best combination for a given query. Table 2 compares our method with UPD on 1000 queries of length 3 on the aPascal dataset. Our method is one order of magnitude faster than UPD which verifies that our algorithm for computing the sum of pairwise distance in the binary space is very fast and efficient. Second, we consider the entire retrieval task which involves identifying the best combination, learning the corresponding component classifiers and finally evaluating them on test images. Table 3 compares our model with UPD and DEF. Interestingly, our method is also faster than DEF. This is because in DEF we always need to train $n(n$: query length) classifiers but in our model on average we need to learn 1.4 classifiers. This comparison assumes that no computations are being done off line. One advantage of DEF over our method is that training and testing in DEF can be done off line.

Calibration Effect: As discussed earlier, calibration is very important when combining multiple component classifiers. Figure 8 empirically verifies this by comparing the performance of UPD with and without calibration on the aPascal data set. Without calibration the performance is almost 5% worse.

Qualitative Evaluation: Finally, we look at some qual-

Method	Time(Second)
Upper Bound	35.325
Ours	0.508

Table 2. Time for finding best combination: Trying all possible combinations of attributes and picking the best one is very expensive. This table compares the time needed to compute the upper bound versus the time that our algorithm needs to decide which combination to pick.

Method	Time(Second)
Upper Bound	167.68
Default	42.56
Ours	22.34

Table 3. Average Retrieval Time : Comparisons between the entire time needed to perform the default case, our method, and the upper bound. This table assumes that no classifiers for the default case are trained off line.

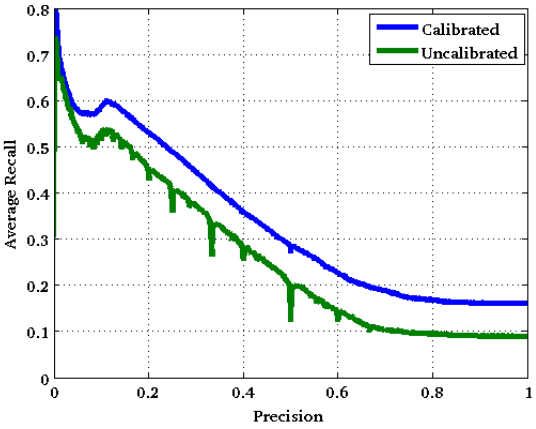

Figure 8. Calibration Effects

itative retrieval results comparing our approach to DEF and UPD. Figure 10 presents top five images retrieved by different methods for several multi-attribute queries.

We now look at which attributes tend to merge with other attributes often, and which ones typically stay un-merged. We created a wordle using wordle.net as seen in Figure 9. The bigger the font size of a word, more likely is the corresponding attribute to merge with other attributes.

5. Conclusion

We address the task of image search using multi-attribute queries. We argue that given a query, the default strategy of training independent classifiers for each attribute and combining their scores to find images that satisfy the query may not be the most effective or efficient strategy. The appearances of images that simultaneously satisfy some combination of attributes may be *significantly* more consistent than a

Figure 9. Some attributes have the tendency to be merged and some prefer to stay separated. The bigger the names in this figure the higher the tendency of the attribute to merge. It is interesting to see that attributes like occluded tend to merge frequently. This is probably because of the fact that the appearance of attributes like this varies a lot as they appear with other attributes. On the other side, attributes like beak and furniture leg tend to be separated as their appearance does not change in combinations.

group of images that all satisfy a single attribute. This motivates the use of classifiers that directly detect combinations of attributes. However, not all combinations result in consistent appearances. In this paper we proposed a novel optimization approach that given a multi-attribute query efficiently identifies which attributes should be merged without exhaustively training classifiers for all possible combinations. Results on two challenging datasets demonstrate the superiority of our approach over strong baselines in terms of performance and run time.

Acknowledgements: This work was partially supported by MURI from the Office of Naval Research under the Grant N00014-10-1-0934.

References

[1] M. Douze, A. Ramisa, and C. Schmid. Combining attributes and fisher vectors for efficient image retrieval. In *CVPR*, 2011. 2

[2] K. Duan, D. Parikh, D. J. Crandall, and K. Grauman. Discovering localized attributes for fine-grained recognition. In *CVPR*, 2012. 4

[3] A. Farhadi, I. Endres, D. Hoiem, and D. Forsyth. Describing Objects by their Attributes. In *CVPR*, 2009. 1, 4

[4] Gionis, Indyk, and Motwani. Similarity search in high dimensions via hashing. 1999. 2, 5

[5] Y. Gong and S. Lazebnik. Iterative quantization: A procrustean approach to learning binary codes. CVPR '11, 2011. 2, 5

[6] M. P. Kumar, B. Packer, and D. Koller. Self-Paced Learning for Latent Variable Models. In *NIPS*, 2010. 5

[7] C. H. Lampert, H. Nickisch, and S. Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *CVPR*, 2009. 1

[8] C. Li, D. Parikh, and T. Chen. Automatic discovery of groups of objects for scene understanding. In *CVPR*, 2012. 2

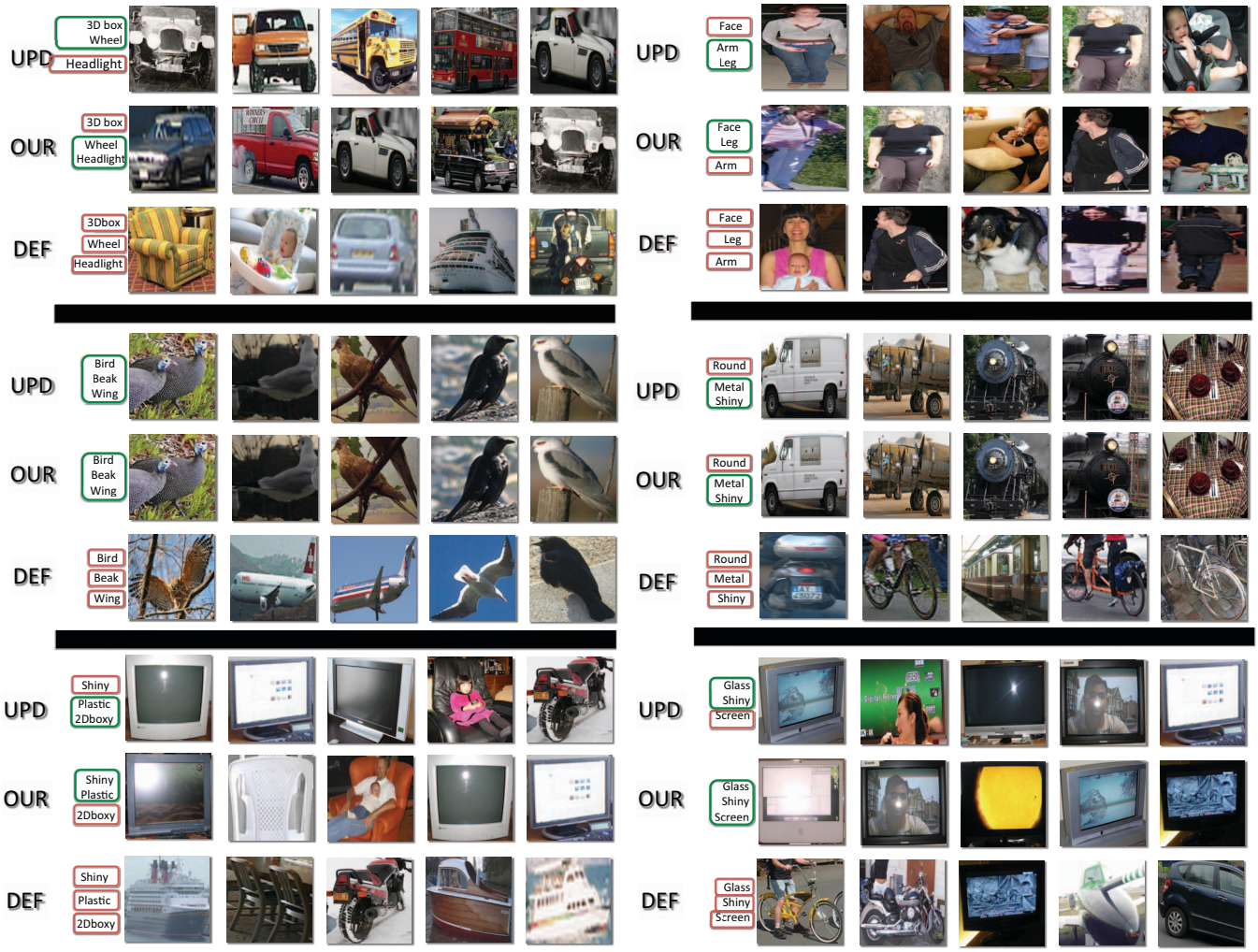

Figure 10. Qualitative comparisons between our method, the default case and the upper bound. Green boxes correspond to merged classifiers and red ones are for independent classifiers. It is interesting to see that when considered beak, wing and bird independently, retrieved images are mixed between planes and birds. This is due to the labeling in aPascal that both birds and planes wing and beaks are labeled with the same label. Once merged with bird the classifier can find the right images.

[9] M. Naphade, J. Smith, J. Tesic, S. Chang, W. Hsu, L. Kennedy, A. Hauptmann, and J. Curtis. Large-scale concept ontology for multimedia. *IEEE Multimedia*, 13(3), 2006. 2

[10] N. Rasiwasia, P. Moreno, and N. Vasconcelos. Bridging the gap: Query by semantic example. *Trans Multimedia*, 9(5), Aug 2007. 2

[11] M. Rastegari, A. Farhadi, and D. A. Forsyth. Attribute discovery via predictable discriminative binary codes. In *ECCV (6)*, 2012. 2, 5

[12] M. A. Sadeghi and A. Farhadi. Recognition Using Visual Phrases. In *CVPR*, 2011. 2

[13] R. Salakhutdinov and G. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 2009. 2

[14] B. Saleh, A. Farhadi, and A. Elgammal. Object-centeric anomaly detection by atribute-based reasoning. In *CVPR*, 2013. 2

[15] W. Scheirer, N. Kumar, P. N. Belhumeur, and T. E. Boult. Multi-attribute spaces: Calibration for attribute fusion and similarity search. In *The 25th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2012. 2

[16] B. Siddiquie, R. S. Feris, and L. S. Davis. Image Ranking and Retrieval based on Multi-Attribute Queries. In *CVPR*, 2011. 2

[17] J. Smith, M. Naphade, and A. Natsev. Multimedia semantic indexing using model vectors. In *ICME*, 2003. 2

[18] A. Wagner, J. Wright, A. Ganesh, Z. Zhou, H. Mobahi, and Y. Ma. Towards a Practical Face Recognition System: Robust Alignment and Illumination by Sparse Representation. *IEEE PAMI*, 2011. 4

[19] X. Wang, K. Liu, and X. Tang. Query-specific visual semantic spaces for web image re-ranking. In *CVPR*, 2011. 2

[20] Y. Weiss, R. Fergus, and A. Torralba. Multidimensional spectral hashing. In *ECCV (5)*, 2012. 2

[21] P. Welinder, S. Branson, T. Mita, C. Wah, F. Schroff, S. Belongie, and P. Perona. Caltech-UCSD Birds 200. Technical report, California Institute of Technology, 2010. 4

[22] E. Zavesky and S.-F. Chang. Cuzero: Embracing the frontier of interactive visual search for informed users. In *ACM MIR*, 2008. 2