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

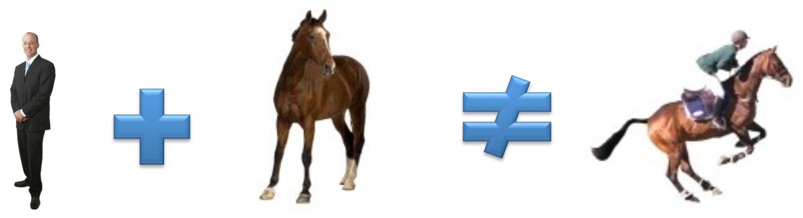
Mohammad Rastegari   Ali Diba   Devi Parikh   Ali Farhadi

University of Maryland · CVPR 2013 Portland, Oregon June 23-28 · Virginia Tech Invent the Future · University of Washington

Goal:

Finding a best combination of keywords (*attributes*) in an image search query.

Visual Phrase:
[Sadeghi et al CVPR2011]



- ***Intuition***: In a multi-attribute image search, some combinations of attributes can be learned jointly, resulting in a better classifier.

Why not:

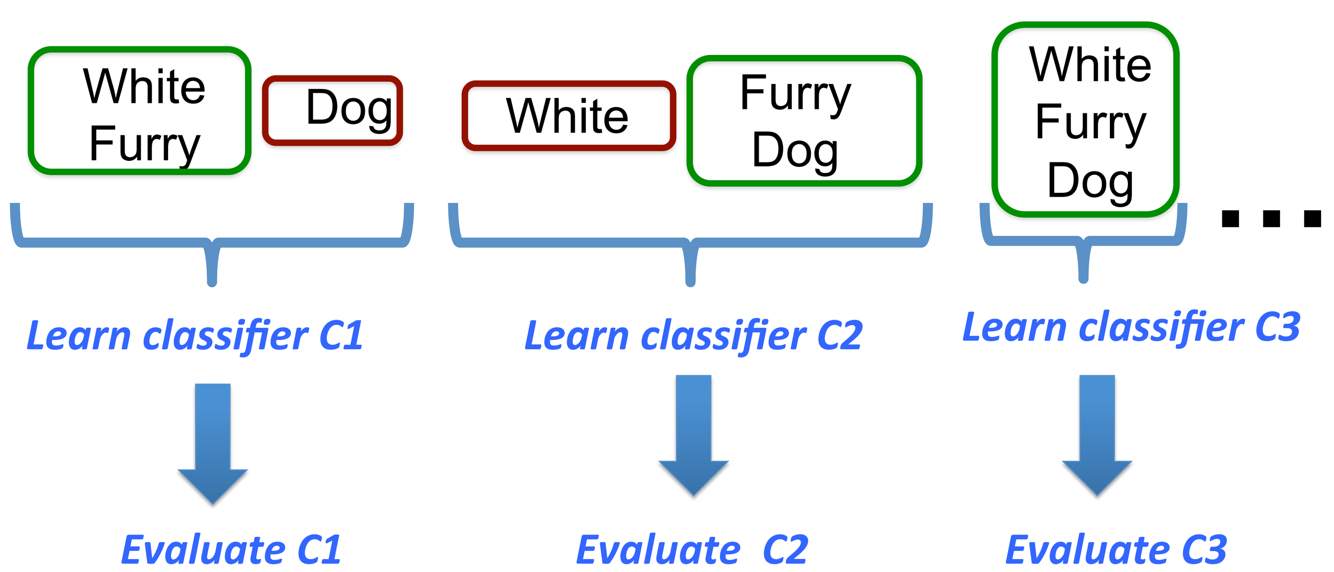
– *Learning individual detector for each attribute?* It may not be effective due to significant difference in appearance (Joint attribute may have similar appearance across images)

– *Learning one detector by merging all attributes?* It may not be powerful due to the lack of jointly labeled training data.

✓Efficient ✗Inaccurate

Naïve Solution (Upper Bound):

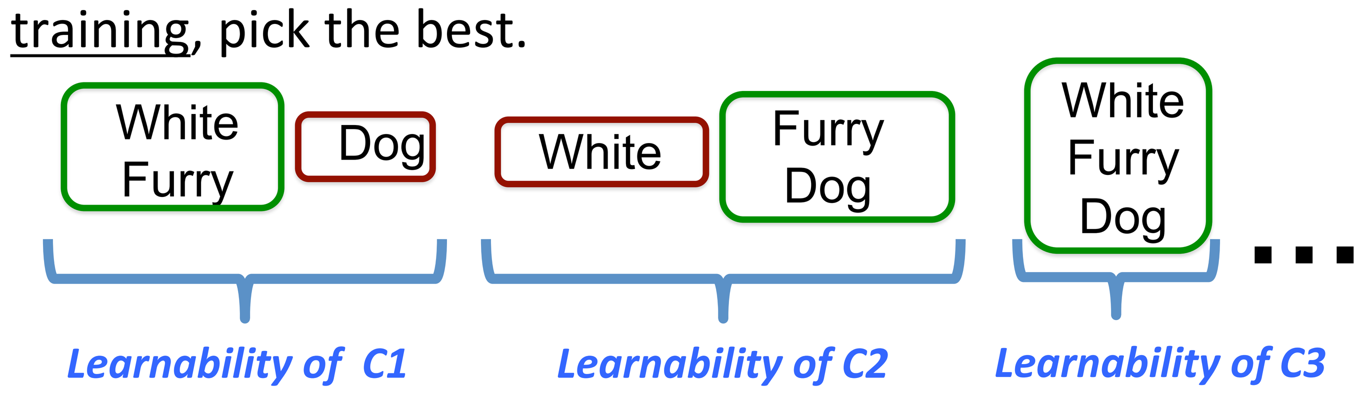
Learn and evaluate all possible combinations on a validation set, pick the best.



✗Inefficient ✓Accurate

Our Approach:

Estimate "learnability" of each combination efficiently without training, pick the best.



✓Efficient ✓Accurate

Learnability Function:

Set of **attributes**: $\mathcal{A} = \{a_1, a_2, ..., a_k\}$

A **component**: $c_i \in \mathcal{S} = \mathcal{P}(\mathcal{A})$ $\quad \mathcal{S} = \{c_1, c_2, ..., c_m\},\ m = 2^k$

A **combination**: $\mathcal{C} \subset \mathcal{S}$

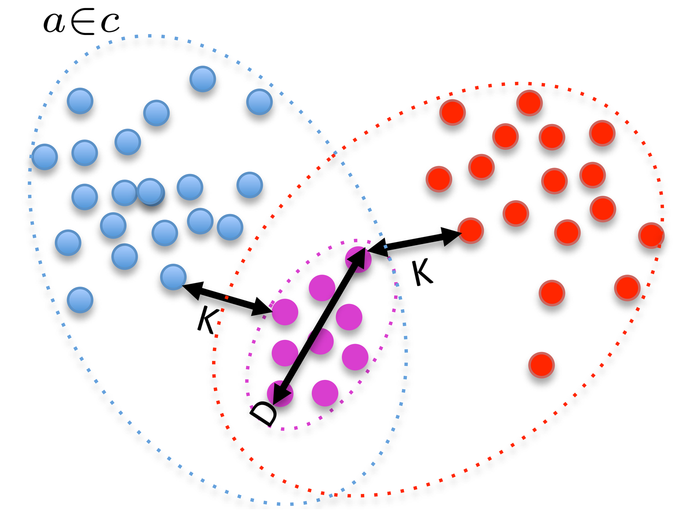
Margin: $\mathcal{K}(c_1, c_2)$ Average pair wise distance between two sets of instances labeled by c_1 *and* c_2

Diagonal: $\mathcal{D}(c)$ Average distance in a set of instances labeled by c

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

Complexity for computing the Margin K between two sets with n_1 and n_2 elements is $O(n_1 n_2)$

Complexity for computing the Diagonal D of a set with n elements is $O(n^2)$



In **Binary** feature space :

Margin → $O(n_1+n_2)$

Diagonal → $O(n)$

Recent binary code methods are very accurate: ***DBC*** [Rastegari et al. ECCV12] and ***ITQ*** [Gong et al CVPR11]

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 → K do
    Z(k) ← Σ_k B2(:, k)  Comment: Counting Number of zeros in k^th dimension of B2
    O(k) ← Σ_k ¬B2(:, k)  Comment: Counting Number of ones in k^th dimension of B2
end for
for i = 1 → N do
    for k = 1 → K do
        if B1(i, j) = 0 then
            P(i, j) ← O(k)
        else
            P(i, j) ← Z(k)
        end if
    end for
end for
S ← Σ P  Comment: Sum of all elements in P
```

Optimization:

$$\max_x \quad \mathcal{L}(\mathcal{S} \odot x) - \lambda |x|$$
$$Z^T x \geq \mathbf{1}$$
$$x \in \{0,1\}^m$$

NP-*Hard*!!

$$Z(i,j) = \begin{cases} 1 & a_j \in c_i \\ 0 & a_j \notin c_i \end{cases}$$

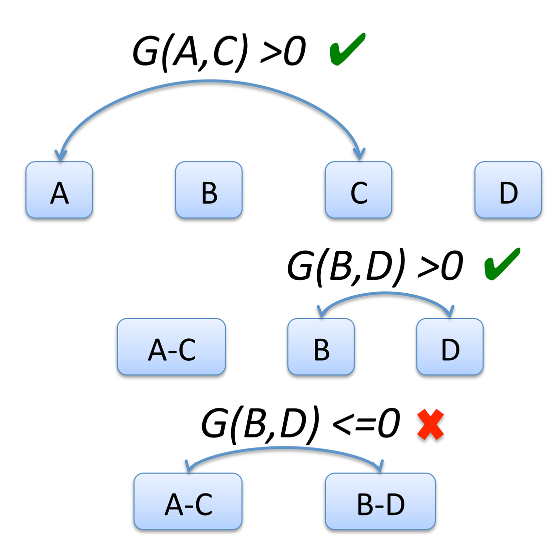
Gain Function:

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

- The higher $G(a_i, a_j)$ the higher is the reward for merging a_i *and* b_j

Greedy Algorithm:

- *For every pairs of attributes compute **G***
- *Pick the pair with maximum **G***
- *If the maximum **G** >0 then :*
  - *1- Merge the two corresponding attributes*
  - *2- Add the new merged-attribute*
  - *3- Remove the two independent attribute*



Reducing 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 $\mathcal{G}(a_i, a_j)$. □

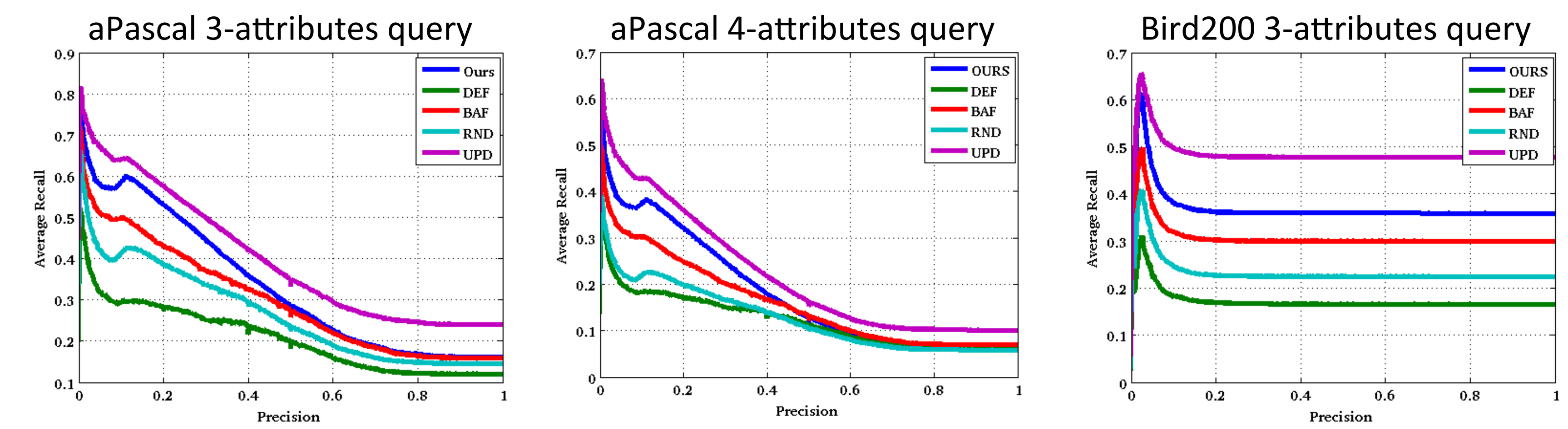
$O(k^3)$ vs. $O(2^k)$

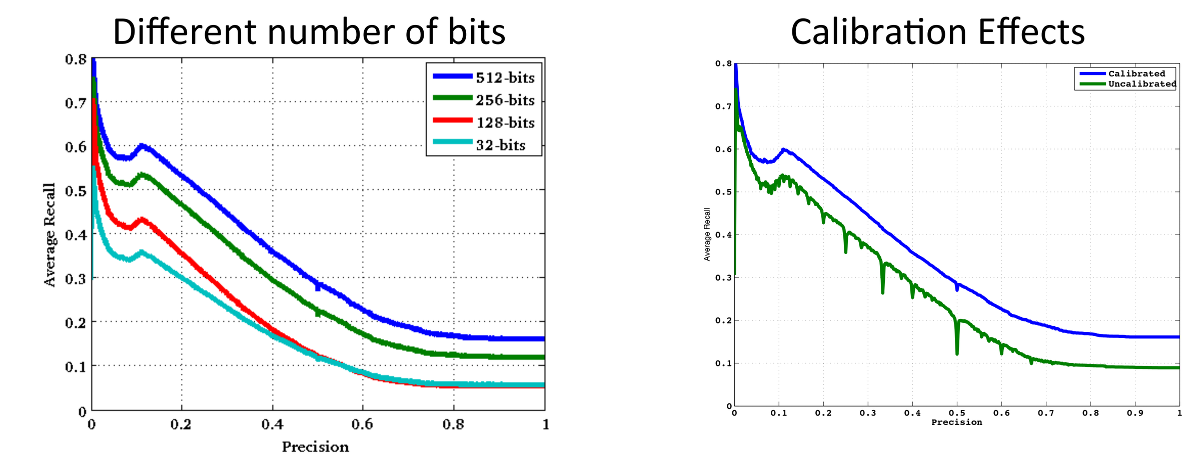
Experiments:

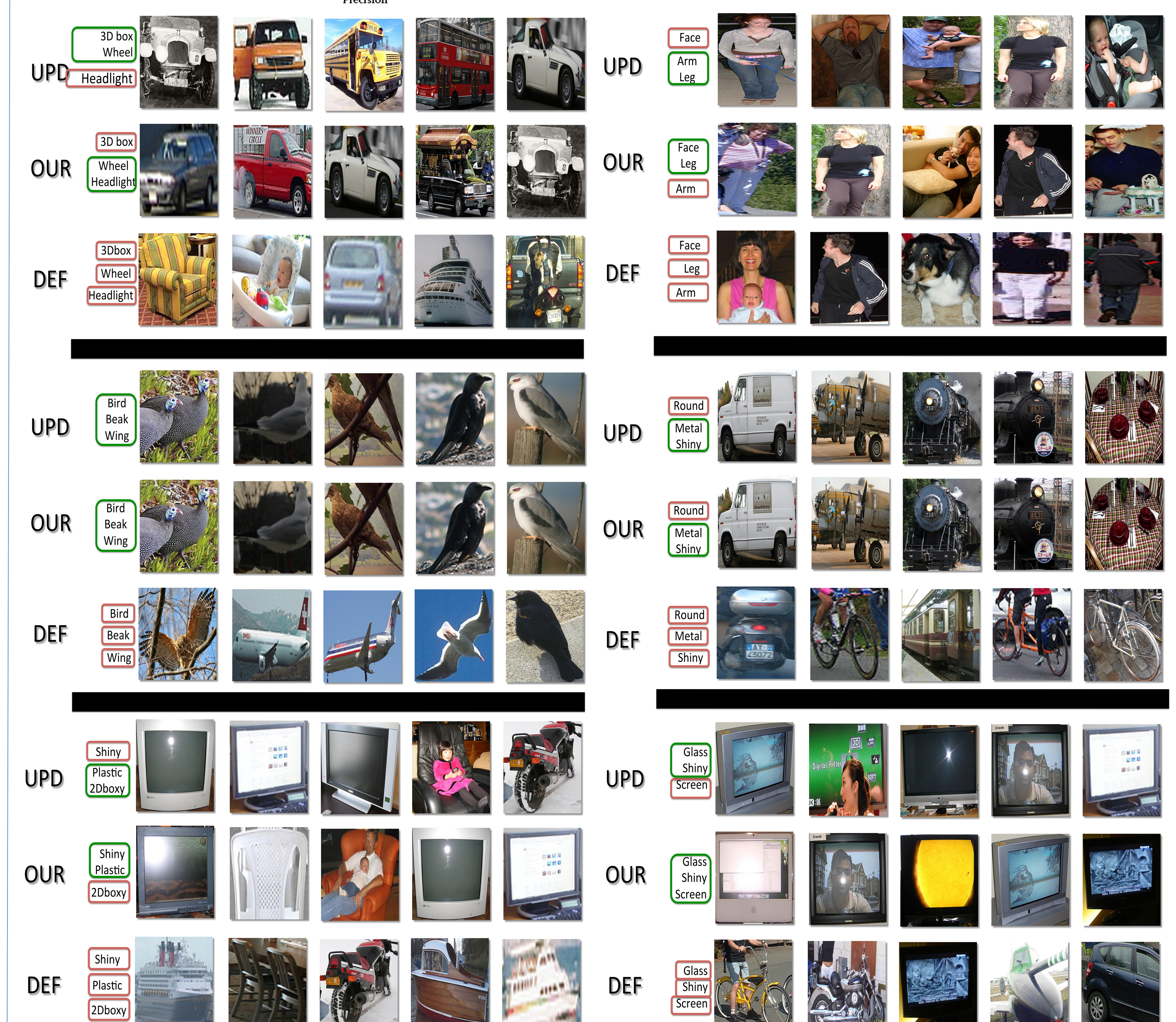
Datasets:

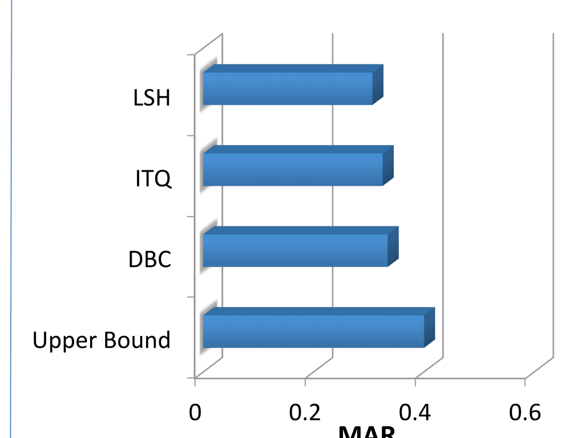
1- aPascal [Farhadi et al. 2009]

2- Caltech-UCSD Bird200 [Welinder et al. 2010]

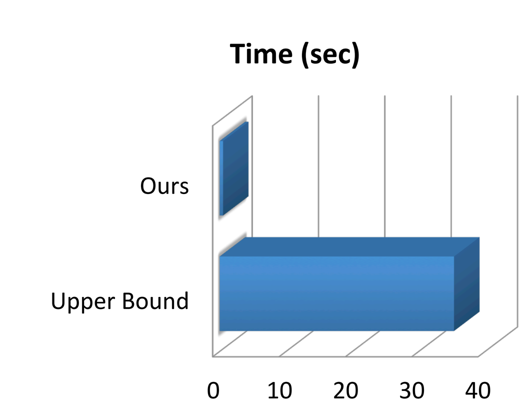








Our method is robust across the binary code methods



Running time for finding the best combination.