Building a Classification Cascade for Visual Identification from One Example

Andras Ferencz
Computer Science, U.C. Berkeley
ferencz@cs.berkeley.edu

Erik G. Learned-Miller
Computer Science, UMass Amherst
elm@cs.umass.edu

Jitendra Malik
Computer Science, U.C. Berkeley
malik@cs.berkeley.edu

(DRAFT: Submitted to ICCV'05)

Abstract

Object identification (OID) is specialized recognition where the category is known (e.g. cars) and the algorithm recognizes an object's exact identity (e.g. Bob's BMW). Two special challenges characterize OID. (1) Inter-class variation is often small (many cars look alike) and may be dwarfed by illumination or pose changes. (2) There may be many classes but few or just one positive "training" examples per class. Due to (1), a solution must locate possibly subtle object-specific salient features (a door handle) while avoiding distracting ones (e.g. specular highlights). However, (2) rules out direct techniques of feature selection. We describe an on-line algorithm that takes one query image from a known category and builds an efficient "same" vs. "different" classification cascade by predicting the most discriminative feature set for that object. Our method not only estimates the saliency and scoring function for each candidate feature, but also models the dependency between features, building an ordered feature sequence unique to a specific query image, maximizing cumulative information content. Learned stopping thresholds make the classifier very efficient. To make this possible, category-specific characteristics are learned automatically in an off-line training procedure from labeled image pairs of the category, without prior knowledge about the category. Our method, using the same algorithm for both cars and faces, outperforms a wide variety of other methods.

1. Introduction

Object identification is specialized object recognition where the category is known (e.g. faces or cars) and one must recognize the exact identity of objects. The classes to be distinguished are not categories, e.g. cars versus non-cars (the problem of Object Categorization), but rather specific objects, like Bob's BMW or Jen's Ford. The hierarchical nature of categories suggests a continuum between these two problems: vehicles to cars to sedans to Phil's sedan. In this paper, we focus on the identification end of this continuum, where

Figure 1: *An Identification Problem: Which cars match?* The two cars on the left were photographed from camera 1. Which of the four images on the right, taken by camera 2, match the cars on the left?

the Object Identification (OID) problem poses different challenges than its coarser cousin, Object Categorization (OC). Specifically, in OID problems (1) the inter-class variation is often small (many cars look alike), and this variation is often dwarfed by illumination or pose changes (see Fig. 1); and (2) there are many classes (each object is a separate class) but few (in our case just one) positive "training" examples per class (e.g. one image representing "Bob's BMW").

People are good at identifying individual objects from familiar categories after seeing them only once. Consider faces. We zero in on discriminative features *for a person* such as a prominent mole or unusually thick eyebrows, yet are not distracted by equally unusual but non-repeatable features such as a messy strand of hair or illumination artifacts. Domain specific expertise makes this possible: having seen many faces one learns that a messy strand of hair is not often a reliable feature. Human vision researchers report that acquisition of this expertise is accompanied by significant behavioral and physiological changes. Diamond et al. [4] showed that dog experts perform dog identification differently than non experts; Tarr et al. [11] argued that the brain's fusiform face area does visual processing of categories for which expertise has been gained.

The processes that occur during Object Categorization (OC) and Object Identification (OID) can be formally characterized. In functional notation, the stages for OC are

1. (Off-line) trainer $\mathcal{T}_{cat}$: class training images $\mapsto \mathcal{C}_{cat}$,

2. (On-line) classifier $\mathcal{C}_{cat}$: test image $\mapsto$ class label.

There is nothing novel here, just the standard paradigm of statistical learning. It relies implicitly on having enough examples of each class to learn discriminative features.

For OID, we assume off-line access to plenty of examples of the category (cars, dogs, faces). We then must develop an on-line classifier for a future image of Bob's BMW, given only one example of it. We decompose the on-line process into two stages: (a) producing an "identifier", a classifier specialized to reidentify a specific object based on a single example of it, and (b) running the "identifier" on the incoming data stream. These on-line stages are preceded by the off-line process of learning *category specific* characteristics, resulting in an "identifier generator". Thus, the three stages for OID are

1. (Off-line) trainer $\mathcal{T}_{id}$: category training images $\mapsto \mathcal{H}_{id}$,
2. (On-line) identifier generator $\mathcal{H}_{id}$: object image $\mapsto \mathcal{C}_{id}$,
3. (On-line) classifier $\mathcal{C}_{id}$: test image $\mapsto \{\text{same, different}\}$.

We stress that step 1 learns *category specific* characteristics, while step 2 creates an *object specific* classifier. Now we address details.

First we need to pick a family of classifiers $\mathcal{C}_{id}$. Motivated by the success of patch (a.k.a. part or fragment) based representations ([12, 8]) for OC, we use them for OID as well. Specifically, we develop an OID system whose generated classifier $\mathcal{C}_{id}$ (step 3) is a patch-based classification cascade similar to that of Vidal-Naquet et al. [12], where evidence from features is accumulated incrementally until a "same" or "different" decision can be made. The tricky part is to give $\mathcal{H}_{id}$ the ability to pick out object specific discriminative features (e.g. a prominent door handle in one car, a roof rack in another). But how can we know that a patch containing a prominent door handle is discriminative, based on *a single image*, when we have never seen a door handle exactly like it before?

The core of our approach is to use *hyper-features*, which are generic position and appearance characteristics of a patch. Examples include location of a patch, edge contrast in the patch and the dominant oriented energy in the patch. We might, in the process of becoming a car identification expert, expect to learn that patches about half-way up with strong edge contrast and a dominant horizontal orientation are particularly informative. When given the specific example of Bob's BMW, the identifier generator $\mathcal{H}_{id}$ could produce an object-specific cascade with the first test based on the patch containing the door handle. Whereas for Jen's Ford, the same set of hyper-features will result in a different ordering of salient patches, resulting in a different classification cascade with the first test using a patch containing the roof rack (see Fig. 4).

More precisely, to instantiate $\mathcal{C}_{id}$ (step 2), the function $\mathcal{H}_{id}$ is given a *single image* of the object (e.g. Bob's BMW) and produces a sequence of patches ordered from most informative to least, that maximizes the cumulative information content. This sequence is object-specific, and may emphasize different parts of each object.

Off-line training $\mathcal{T}_{id}$ (step 1), given a set of image pairs from the category, each pair labeled "same" or "different," produces a class-specific $\mathcal{H}_{id}$ by learning (a) a saliency model for image patches as a function of patch characteristics like position and appearance (*hyper-features*), (b) a dependency model between image patches based on similarity of their hyper-features, and (c) a set of thresholds for the cascade. The specific hyper-features used are themselves automatically selected during this training step from a large pool of candidate patch characteristics.

In contrast to some other "one-shot" learning algorithms [10, 7], where off-line training involves finding priors for a fixed model, our $\mathcal{T}_{id}$ actually learns how to identify an arbitrary number of good features for the given category. Thus our final classifier $\mathcal{C}_{id}$, while always a cascade of image patches taken from the probe object, will have a different set of patches (in size, location, and count) for each object. To score a patch and its correspondent in a probe image, our technique uses generalized linear models (GLMs) to estimate a generative model for the dissimilarity between patch pairs. "Same" and "different" distributions based on the hyper-features of the patch are estimated. These distributions are used both to estimate the saliency of a patch (by computing the expected mutual information between the dissimilarity and decision variables) and to score a patch pair by comparing the likelihood under the same and different distributions. By estimating bivariate "same" and "different" distributions for neighboring patches, we model the dependency relationships, allowing us to compute a sequence of patches with high joint information content.

Section 2 summarizes our previously published work on using hyper-features for visual identification [1]. That work had a serious limitation: it assumed that the patches were independent. This assumption is clearly false, especially for nearby and overlapping patches. To make that system work, we allowed only a single patch size and added a simple penalty term for a patch that was not a local maximum. Here we allow the system to pick patches of varying sizes, forcing us to model the patch dependencies. This model and its estimation from the training data is described in Section 3. With this dependency model, we build the cascade in Section 4 by finding stopping thresholds for making "same" or "different" decisions. Section 5 details our extensive experiments on multiple car and face data sets.

2. Learning Hyper-Features

We begin by outlining the basic components of our system, some of which were previously detailed in [1]. As the main focus of this paper is modeling patch dependency (Section 3) and building the classification cascade (Section 4), we only summarize these components and ask the reader to refer to [1]

for additional details and motivations for our choices. We describe the training ($\mathcal{T}_{id}$), identifier generating ($\mathcal{H}_{id}$), and classification ($\mathcal{C}_{id}$) functions in reverse order, starting with the final form of the object-specific classifier. In the following, we assume that all images are known to contain objects of the given category (e.g. cars or faces) and have been brought into rough correspondence (see Section 5 for details).

2.1. Classifier $\mathcal{C}_{id}$

The classifier $\mathcal{C}_{id}$ decides if a test (a.k.a. database, right) image I^R is the same ($C = 1$) or different ($C = 0$) than the probe (a.k.a. query, left) image I^L it was trained for.

Patches. Our classifier consists of a sequence of image patches from the probe image I^L and denoted F_j^L for $1 \leq j \leq m$. Unlike our previous algorithm from [1], these patches can have different sizes and resolutions (by using different levels of a Gaussian pyramid). Generally speaking, larger patches are sampled at lower resolutions, keeping the complexity of the patches approximately constant. The grayscale (we currently don't use color information) pixels of the patch are encoded by applying a first derivative Gaussian odd-symmetric filter at four orientations (horizontal, vertical, and two diagonal), giving four signed numbers per pixel.

Matching. Each encoded patch F_j^L is matched to an equally sized area in the test image I^R, by searching for the most similar patch F_j^R within some small neighborhood around the expected location (according to the coarse alignment). The distance function that this search minimizes is one minus the normalized correlation $d_j = 1 - \mathit{CorrCoef}(F_j^L, F_j^R)$ between the encoded patches. The appearance distance d_j is used as evidence for deciding if I^L and I^R are the same ($C = 1$) or different ($C = 0$).

Likelihood Ratio Score. To convert d_j to a score, $\mathcal{C}_{id}$ stores probability distributions $P(d_j|C = 1)$ and $P(d_j|C = 0)$ for each patch and computes the log likelihood ratio. (Note: to limit the number of variables in this discussion, d_j can refer to both the random variable over which we estimate a distribution and the specific measured distance value for a particular patch pair). After m patches have been matched, assuming for now that the patches are statistically independent, we score the match between images I^L and I^R using the sum of log likelihood ratios of matched patches:

$$R = \sum_{j=1}^{m} \log \frac{P(d_j|C = 1)}{P(d_j|C = 0)}. \quad (1)$$

To evaluate this, we must evaluate $P(d_j|C = 1)$ and $P(d_j|C = 0)$. In our system, both of these take the form of gamma distributions $\Gamma(d_j; \theta_j^{C=1})$ and $\Gamma(d_j; \theta_j^{C=0})$, where the parameters $\theta_j^{C=1}$ and $\theta_j^{C=0}$ are defined as part of the classifier $\mathcal{C}_{id}$ for each patch and are set by $\mathcal{H}_{id}$ based on hyper-features.

Making a Decision. In [1], $\mathcal{C}_{id}$ matched a fixed number of patches (m), computed the score R by Eq. 1, and compared it to a threshold λ. $R > \lambda$ meant that I^L and I^R are the same. Otherwise they are declared different. In Section 4 of this paper, we define a cascade from the sequence of patches by applying thresholds after each patch has been matched.

To summarize, the classifier $\mathcal{C}_{id}$ is defined by a sequence of patches of varying sizes (denoted F_j^L) taken from the probe image I^L. Additionally, a pair of parameters $\theta_j^{C=1}$ and $\theta_j^{C=0}$ that define the distributions $P(d_j|C = 1)$ and $P(d_j|C = 0)$ are associated with each patch.

2.2. Classifier Generator $\mathcal{H}_{id}$

The classifier generator $\mathcal{H}_{id}$ must take in a single probe image I^L of a new object from the given category and produce a sequence of patches $F_1^L, ..., F_m^L$ and their associated gamma parameters, $\theta_1^{C=1}, ..., \theta_m^{C=1}$ and $\theta_1^{C=0}, ..., \theta_m^{C=0}$, for scoring based on the appearance distance measurement d_j (which is measured when the patch F_j^L is matched to a location in the test image I^R).

Estimating $\theta_j^{C=1}$ and $\theta_j^{C=0}$. Since being able to estimate a good $\theta_j^{C=1}$ and $\theta_j^{C=0}$ (θ_j for short) for any patch j is also the key to picking good patches, we start with this step. Conceptually, we want θ_j to be influenced by what patch F_j^L looks like and where it is on the object (see the discussion of hyper-features in Section 1). First, we extract a predefined set of hyper-features from the patch such as [x_pos, x_pos^2, $size$, $resolution$, $contrast^3$, $vertical\_energy$, ...]. Let $\mathbf{Z_j} = [Z_1, ..., Z_l]^T$ be a vector of these hyper-features for patch j, and let θ_j be parameterized as $\theta = \{\mu_j, \gamma_j\}$. Now we define a generalized linear model (GLM) [9], which links these hyper-features $\mathbf{Z}$ to the gamma distribution ($\Gamma()$) model for $P(d_j|C = 1)$ and $P(d_j|C = 0)$:

$$P(d_j|Z, C) = \Gamma(d_j;\ \alpha_{\mathbf{C}}^{\mu} \cdot \mathbf{Z_j},\ \alpha_{\mathbf{C}}^{\gamma} \cdot \mathbf{Z_j}), \quad (2)$$

where the second and third arguments to $\Gamma()$ are mean μ and shape γ parameters. Each α (there are four of these $\alpha_{C=0}^{\mu}, \alpha_{C=0}^{\gamma}, \alpha_{C=1}^{\mu}, \alpha_{C=1}^{\gamma}$) is a vector of parameters of length l that weights each hyper-feature monomial Z_i. The key point to notice is that given a hyper-feature encoding (the definition of which patch characteristics to extract) and the linear weights α, we can estimate the distributions $P(d_j|C = 1)$ and $P(d_j|C = 0)$ for any probe image patch F_j^L, based on its position and appearance.

Estimating Saliency. If we define the saliency of a patch as the amount of information about the decision C likely to be gained if the patch were to be matched, then it is straightforward to estimate saliency given $P(d_j|C = 1)$ and $P(d_j|C = 0)$. Intuitively, if $P(d_j|C = 1)$ and $P(d_j|C = 0)$ are similar distributions, we don't expect much useful information from a value of d_j. On the other hand, if the distributions are very different, then d_j can tell us a great deal about our decision. Formally, this can be measured as the mutual information between the decision variable C and the ran-

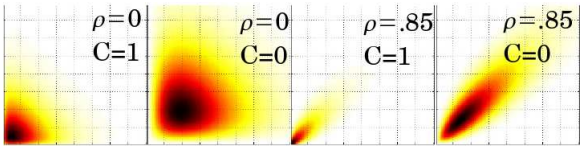

Figure 2: *Bivariate Gamma Distributions.* The two plots on the left and the two on the right each show a typical pair of bivariate gamma distributions for $C = 1$ and $C = 0$. In the left pair, the marginals are uncorrelated ($\rho = 0$), while in the right pair the marginals are highly correlated ($\rho = 0.85$).

dom variable d_j. To emphasize the fact that the distribution $P(d_j|C)$ is a fixed function of F_j^L (see above), we slightly abuse notation and refer to the random variable from which d_j is sampled as F_j^L. With this notation, computing the mutual information between F_j^L and C gives us a measure of the expected information gain from a patch with particular hyper-features:

$$I(F_j^L; C) = H(F_j^L) - H(F_j^L|C).$$

Here $H()$ is Shannon entropy.

Finding Good Patches. With the above mutual information formula, we can estimate the saliency of any patch. Thus defining a sequence of patches to examine in order, from among all candidate patches, is straightforward: for each candidate patch estimate the distributions $P(d_j|C = 1)$ and $P(d_j|C = 0)$ from the hyper-features; compute the mutual information $I(F_j^L; C)$; sort the patches by $I(F_j^L; C)$; and take the top m patches. The problem with this procedure is that the patches are not independent: once we have matched a patch F_j^L, the amount of *additional* information we are expected to derive from matching patch a F_i^L that overlaps F_j^L is less then $I(F_i^L; C)$ would suggest. We discuss a solution to this problem in Section 3.

However, assuming that this dependency problem can be solved, we have a complete algorithm for generating the classifier $\mathcal{C}_{id}$ from a single image, given a definition of the hyper-features to extract (the patch statistics $\mathbf{Z}$) and the linear weights α.

2.3. Off-line Training $\mathcal{T}_{id}$

The task of the off-line training step $\mathcal{T}_{id}$ is to define the hyper-feature encoding $\mathbf{Z}$ and to learn the weights α that link this encoding to the distributions $P(d_j|C = 1)$ and $P(d_j|C = 0)$. This step is given a large collection of image pairs from the category, where each left-right image pair is labeled as "same" or "different". A large number of patches F_j^L are sampled from the left images and matched to the right images (by finding the best matching F_j^R) in the same manner as during classification $\mathcal{C}_{id}$ (see Matching in Section 2.1), and the appearance distance d_j is recorded. For each patch, a large set of candidate hyper-features are also extracted from the position and appearance of the left patch F_j^L. This data gives rise to 2 generalized linear regression problems: one for the "same" ($C = 1$) set and one for the "different" ($C = 0$) set. Our solution involves (1) a feature selection step which finds a hyper-feature encoding ($\mathbf{Z}$) by choosing a small subset from the candidate set of hyper-features, and (2) a maximum likelihood estimation step to fit $\alpha^{\mu}_{C=1}, \alpha^{\gamma}_{C=1}$ and $\alpha^{\mu}_{C=0}, \alpha^{\gamma}_{C=0}$.

Figure 3: *Patch Correlations.* On each image, the patches most correlated with the white-circled patch are shown. Notice that in the left image, where the patch sits in an area with a highly visible horizontal structure, the most correlated patches all lie along the horizontal features. Contrast this with the right image, showing correlation of patches with a patch sitting on a wheel, where the most correlated patches are those that strictly overlap the white-circled patch.

3. Modeling Pairwise Relationships Between Patches

In Section 2, we described our model to score a probe image patch F_j^L and its best match F_j^R by modeling the distribution of their distance in appearance, d_j, conditioned on the match variable C. Furthermore, in Section 2.2, we described how to infer the saliency of the patch F_j^L for matching based on these distributions. As we noted in that section, this works for picking the first patch, but is not optimal for picking subsequent patches: once we have already matched and recorded the score of the first patch, the amount of information gained from a nearby patch is likely to be small, because their scores are likely to be correlated. Intuitively, the next chosen patch would ideally be a highly salient patch whose information about C is as independent as possible from the first patch. Similarly, the third patch should consider both the first and the second patches.

Let $F_{(k)}^L$ represent the kth patch picked for the cascade and let $F_{(1\ldots n)}^L$ denote the first n of these patches. Assume we have already picked patches $F_{(1\ldots n)}^L$ and we wish to choose the next one, $F_{(n+1)}^L$, from the remaining set of F_j^L's. We would like to pick the one that maximizes the *information gain* or the *conditional mutual information*:

$$I(F_{(n+1)}^L; C|F_{(1\ldots n)}^L) = I(F_{(1\ldots n+1)}^L; C) - I(F_{(1\ldots n)}^L; C).$$

This quantity is difficult to estimate, due to the need to model the joint distribution of all $F_{(1\ldots n)}^L$ patches. However, note that the information gain of a new feature is up-

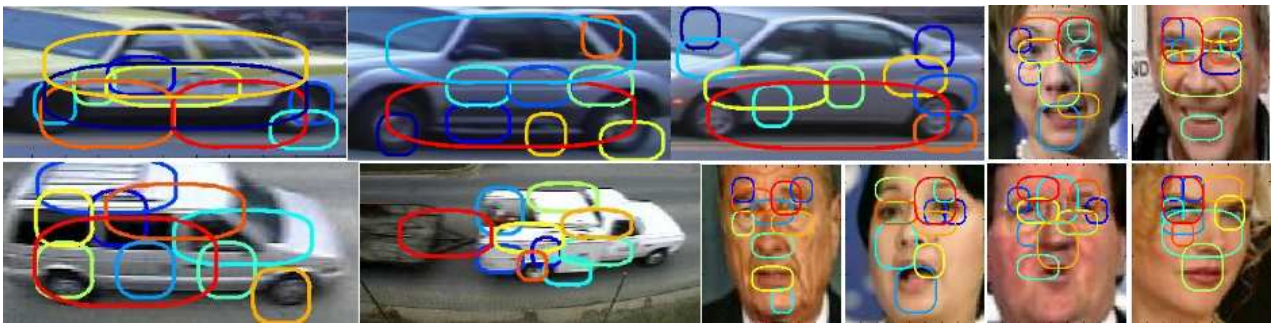

Figure 4: *The Ten Most Informative Patches.* The ten rectangles on each object show the top ten patches our algorithm selected for the classification cascade *for that object*. The face model seems to prefer features around the eyes, while the 2 car models (2 data sets, top and bottom) tend to both like wheels but differ in their interest in the roof region. Notice, however, that even within a category each cascade is unique, highlighting interesting appearance features for that object. The patches are color coded according to their order, from dark red (1) to dark blue (10).

per bounded by the information gain of that feature relative to any *single* feature that has already been chosen. That is,

$$I(F^L_{(n+1)}; C|F^L_{(1\ldots n)}) \leq \min_{1\leq i\leq n} I(F^L_{(n+1)}; C|F^L_{(i)}). \quad (3)$$

Thus, rather than maximizing the full information gain, we select the new feature that maximizes this upper bound on the amount of "new" information:

$$\arg\max_j \min_i I(F^L_j; C|F^L_{(i)}), \quad (4)$$

where i varies over the already chosen patches, and j varies over the remaining patches. This formulation (Eq. 4) follows that of Vidal-Naquet et al. [12].

3.1. Dependency Model

To compute (4), we need to estimate conditional mutual informations of the form

$$I(F^L_j; C|F^L_{(i)}) = I(F^L_j, F^L_{(i)}; C) - I(F^L_{(i)}; C).$$

In Section 2.2, we showed that we can determine the second term, $I(F^L_{(i)}; C)$, from the estimated gamma distributions for $P(d_{(i)}|C=1)$ and $P(d_{(i)}|C=0)$. Similarly, to calculate $I(F^L_j, F^L_{(i)}; C)$, we need an estimate of the bivariate distributions for $P(d_{(i)}, d_j|C=1)$ and $P(d_{(i)}, d_j|C=0)$. If the $d_{(i)}$ and d_j are independent conditioned on C, then these are straightforward to compute from the known marginal distribution parameters for $d_{(i)}$ and d_j. To model the dependent case, we employ Kibble's bivariate gamma distribution [6], which has four parameters: $K(\mu_1, \mu_2, \gamma, \rho)$, $0 < \rho < 1$. μ_1 and μ_2 are mean parameters for the marginals, and γ is a dispersion parameter for both marginals (the formulation requires these to be equal). ρ is the correlation between $d_{(i)}$ and d_j, and varies from 0, indicating full independence of the marginals, to 1, in which the marginals are completely correlated (see figure 2).

To make this formulation work, the marginal distribution parameters must be constrained to be equal ($\mu^{C=1}_j = \mu^{C=1}_{(i)}$, as well as $\gamma^{C=1}_j = \gamma^{C=1}_{(i)}$)[1]. Therefore, for the computation of the conditional mutual information of F^L_j conditioned on $F^L_{(i)}$, we force the marginal distribution of the already chosen patch ($F^L_{(i)}$) to be equal to the marginal distribution of the patch currently being considered (F^L_j). Given that our method for comparing all patches is the same, namely normalized correlation, this usually means a very minor perturbation to the estimated distribution of $F^L_{(i)}$ when the two patches are strongly correlated. On the other hand, when the marginals are originally fairly different, the two patches tend to be uncorrelated. In this case, the exact shapes of $F^L_{(i)}$'s distributions are less relevant to the computation of Eq. (4). Since we are always setting the first two parameters of the Kibble's distribution to be the same, we will henceforth write it with three parameters (e.g. $K(\mu^{C=0}_j, \gamma^{C=0}_j, \rho)$).

3.2. Predicting Patch Correlations from Hyper-Feature Differences

Given the above formulation, we have reduced the problem of finding the next best patch, $F^L_{(n+1)}$, to the problem of estimating the correlation parameter ρ of Kibble's bivariate gamma distribution for any pair of patches $F^L_{(i)}$ (one of the n patches already selected) and F^L_j (a candidate for $F^L_{(n+1)}$). The intuition is that patches that are nearby and overlapping or that lie on the same underlying image features (for example the horizontal line on the side of the car in Figure 3) are likely to be highly correlated, whereas two patches that are of different sizes and far away from one another are likely to be less so.

We model ρ, the last parameter of $K(\mu^{C=1}_j, \gamma^{C=1}_j, \rho)$ and $K(\mu^{C=0}_j, \gamma^{C=0}_j, \rho)$, similarly to our GLM estimate of its other parameters (see Section 2.2): we let ρ be a linear function of the *difference* of various hyper-features of the

[1] More precisely the γ's must be equal (this is a requirement of Kibble's formulation), while the μ's must satisfy $\frac{\mu^{C=1}_j}{\mu^{C=1}_{(i)}} = \frac{\mu^{C=0}_j}{\mu^{C=0}_{(i)}}$

two patches, $F^L_{(i)}$ and F^L_j. Clear candidates for these covariates are the difference in position and size of the two patches, as well as some image-based features such as the difference in the amount of contrast within each patch. To ensure $0 < \rho < 1$, we use a *sigmoid* link function

$$\rho = (1 - \exp(\beta \cdot \mathbf{Y}))^{-1},$$

where $\mathbf{Y}$ is our vector of hyper-feature differences and β is the GLM parameter vector.

Given a data set of patch pairs $F^L_{(i)}$ and F^L_j and associated distances $d_{(i)}$ and d_j (found by matching the "left" patches to a "right" image of the same or of a different vehicle), we estimate the linear coefficients β. This is done by maximizing the likelihood of $K(\mu^{C=1}_j, \gamma^{C=1}_j, \rho)$ using data taken from image pairs that are known to be the "same"[2] and $K(\mu^{C=0}_j, \gamma^{C=0}_j, \rho)$ using data taken from "different" image pairs. Also similarly to Section 2.3, we choose the encoding of $\mathbf{Y}$ automatically, by the method of forward feature selection [5] over candidate hyper-feature difference variables. As anticipated, the top ranked variables encoded differences in position, size, contrast, and orientation energy. Our final model uses the top 10 variables.

4. Building the Cascade

Now that we have a model for patch dependence, we can create a sequence of patches F^L_j (see Section 2.2) that, when they are matched, collectively capture the maximum amount of information about the decision C (same or different?). The sequence is ordered so that the first patch is the most informative, the second slightly less so and so on. The final step of creating a cascade is to define early stopping thresholds on the log likelihood ratio sum R that can be applied after each patch in the sequence has been matched and its score added to R (see Section 2.1).

We assume that we are given a global threshold λ (see Section 2.1) that defines a global choice between selectivity and sensitivity. What remains is the definition of thresholds at each step, $\lambda^{accept}_{(k)}$ and $\lambda^{reject}_{(k)}$, which allow the system to accept (declare "same") if $R > \lambda^{accept}_{(k)}$ or reject (declare "different") if $R \leq \lambda^{accept}_{(k)}$, otherwise continue by matching patch $k+1$. To learn these thresholds, we run $\mathcal{H}_{id}$ on the left images and the resulting classifier $\mathcal{C}_{id}$ on the right images of our training data set. This will produce a performance curve for each choice of k, the number of patches included in classification score, including $k = m$, the sum for which λ is defined. Our goal for the cascade is for it to make decisions as early as possible (tight thresholds) but, on the training set, never make a mistake on any pair which was correctly classified using all m patches and the threshold λ. These two constraints exaclty define the thresholds $\lambda^{accept}_{(k)}$ and $\lambda^{reject}_{(k)}$.

[2] $\mu^{C=1}_j$ and $\gamma^{C=1}_j$ are estimated from F^L_j by the method of Section 2.3 and are fixed for this optimization.

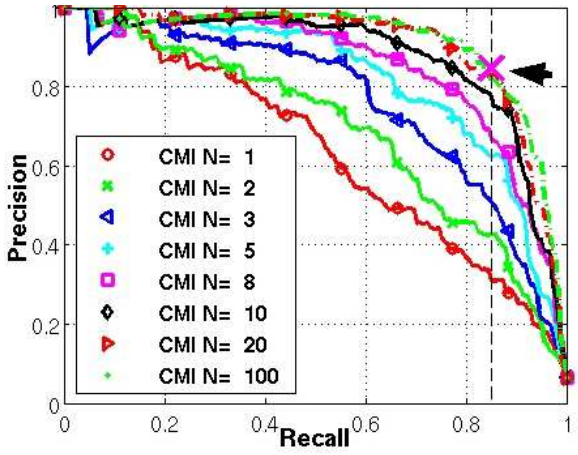

Figure 5: *Precision vs. Recall Using Different Numbers of Patches.* These are precision vs. recall curves for our full model. Each curve represents the performance tradeoff between precision and recall, when the system uses a fixed number of patches. The lowest curve uses only the single most informative patch, while the top curve uses up to 100 patches. The 85% recall rate, where the different models of Figure 6 are compared, is noted by a vertical black dashed line. A magenta X, at recall = 84.9 and precision = 84.8, marks the performance of the cascade model.

5. Results and Conclusion

The goal of this work was to create an identification system that could be applied to different categories, where the algorithm would automatically learn (based on off-line training examples) how to select category-specific salient features from a new image. In this section, we demonstrate that after category training, our algorithm is in fact able take a single image of a novel object and solely based on it create a highly effective "same" vs. "different" classification cascade of image patches. Specifically, we wish to show that for visual identification each of the following leads to an improvement in performance in terms of accuracy and/or computational efficiency:

1. breaking the object up into patches (a.k.a parts, fragments), matching each one separately and combining the results,
2. differentiating patches by estimating a scoring and saliency function for each patch (based on its hyper-features),
3. modeling the dependency between patches to create a sequence of patches to be examined in order, and
4. applying early termination thresholds to the patch sequence to create the cascade.

We tested our algorithm on 3 different data sets: (1) cars from 2 cameras with significant pose differential, (2) faces from news photographs, and (3) cars from a wide-area tracking system with 33 cameras and 1000's of unique vehicles. Examples from these 3 data sets are shown in Figure 4, with

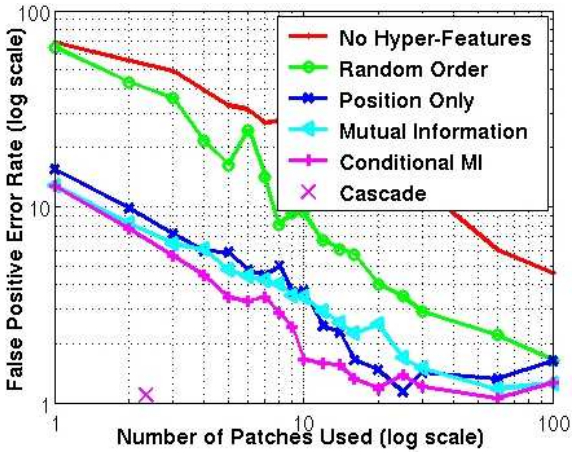

Figure 6: *Comparing Performance of Different Models.* The curves plot the performance of various models, as measured by the false-positive rate (fraction of different pairs labeled incorrectly as same), at a fixed recall rate of 85%. The y-axis shows the log error rate, while the x-axis plots the log number of patches the models were allowed to use (up to a max of 100). As the number of patches increases, the performance improves until a point, after which it levels off and, for the models that order patches according to information gain, even decreases (when non-informative patches begin to pollute the score). The (red) model that does *not use hyper-features* (i.e. uses the same distributions for all patches), performs very poorly compared to the hyper-feature versions, even when it is allowed to use 100 patches. The second curve from the top uses our hyper-feature model to score the patches, but *random selection* to pick the patch order. The *position only model* uses only position-based hyper-features for selecting patch order (i.e. it computes a fixed patch order for all cars). The light blue model sorts patches by *mutual information*, without considering dependencies. The last curve shows our full model based on selecting patches according to their *conditional mutual information*, using both positional and image-based hyper-features. Finally, the magenta X at 2.58 patches and 1.02% error shows the performance of the cascade model.

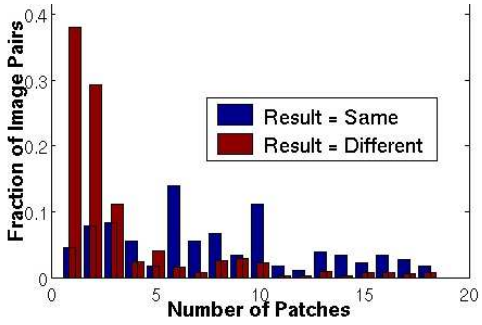

Figure 7: *How many patches does it take to make a decision?* This histogram shows the number of patches that were matched by the classification cascade before a decision could be made. On average, 2.2 patches were required to make a negative (declaring a difference) decision, and 8.0 patches to make a positive one.

Recall Rate	60%	70%	80%	90%
PCA + MahCosine	82%	73%	62%	59%
Filter + NormCor	83%	73%	67%	57%
No Hyper-Features	86%	73%	68%	62%
Random 10 Patches	79%	71%	64%	60%
Top 1 CMI Patch	86%	76%	69%	63%
Top 50 CMI Patches	92%	84%	75%	67%
CMI Cascade	**92%**	**84%**	**76%**	**66%**

Table 1: *Precision vs. Recall for Faces.*
Each column denotes the precision associated with a given recall rate along the P-R curve. *PCA + MahCosine* and *Filter + NormCor* are whole face comparison techniques. *PCA + MahCosine* is the best curve produced by [3], which implments PCA and LDA algorithms with face-specific preprocessing. *Filter + NormCor* uses the same representation and comparison method as our patches, but applied to the whole face. The last 4 all use our patch based system with hyper-features. The last 3 uses conditional mutual information based patch selection, where the number of patches allowed is set to 1, 50, and variable (cascade), respectively. The cascades use between 2-3 patches on average to make a decision.

the top 10 patches of the classification cascade. For each data set, a different automatic preprocessing step was applied to detect objects and approximately align them. After this, the same identification algorithm was applied to all 3 sets. For lack of space, we detail our experiments on data set 1, enumerate the results of data set 2, and only summarize our experience with data set 3. Qualitatively, our results on the three are consistent in showing that each of the above aspects of our system improves the performance, and that the overall system is both efficient and effective.

5.1. Cars 1

358 unique vehicles (179 training, 179 test) were extracted using a blob tracker from 1.5 hours of video from 2 cameras located one block apart. The pose of the cameras relative to the road (see Figure 1) was known from static camera calibration, and alignment included warping the sides of the vehicles to be approximately parallel to the image plane. Within training and testing sets, about 2685 pairs (true to false ratio of 1:15) of mismatched cars were formed from non-corresponding images, one from each camera. These included only those car pairs that were superficially similar in intensity and size. Using the best whole image comparison method we could find (normalized correlation on blurred filter outputs) on this set produces 14% false positives at a 15% miss rate. This data set is available from [Anonymous].

Figure 6 compares several versions of our model by plotting the false-positive rate (y-axis) with a fixed miss rate of 15% (85% recall), for a fixed budget of patches (x-axis). The 85% recall point was selected based on Figure 5, by pick-

ing the equal error point given the 1 to 15 true-to-false ratio. The *Random Order* curve uses our hyper-feature model for scoring, but chooses the patches randomly. By comparing this curve to its neighbors, notice the performance gain associated with differentiating patches based on hyper-features both for scoring (*No Hyper-Features* vs. *Random Order*) and for patch selection (*Random Order* vs. *Mutual Information*). Comparing *Mutual Information* vs. *Conditional MI* shows that modeling patch dependence is important for choosing a small number of patches (see range 5-20) that together have high information content (Section 3). Comparing *Position Only* (which only uses positional hyper-features) vs. *Conditional MI* (which uses both positional and appearance hyper-features) shows that patch appearance characteristics are significant for both scoring and saliency estimation. Finally, the cascade performs (1.02% error, with mean of 2.58 patches used) as well as the full model and better than any of the others, even when these are given an unlimited computation budget.

Figure 5 shows another way to look at the performance of our full model given a fixed patch (computation) budget (the *Conditional MI* curve of Figure 6 represents the intersection of these curves with the 85% recall line). The cascade performance is also plotted here (follow the black arrow). The distribution of the number of patches it took to make a decision in the cascade model is plotted in Figure 7.

5.2. Faces

We used a subset of the "Faces in the News" data set described in [2], where the faces have been automatically detected from news photographs and registered by their algorithm. Our training and test sets each used 103 different people, with 2 images per person. This is an extremely difficult data set for any identification algorithm, as these face images were collected in a completely uncontrolled manner (news photographs). Table 5 summarizes our results for running the same algorithm as above on this set. Note the same pattern as above: the patch based system generally outperforms whole object systems (here we compare against state of the art PCA and LDA algorithms with face specific preprocessing using CSU's implementation [3]); estimating a scoring and saliency function through hyper-features greatly improves the performance of the patch based system; the cascade, using less than 3 patches on average, performs as well as always using the best 50 patches (performance actually declines above 50 patches).

5.3. Cars 2

We are helping to develop a wide-area car tracking system where this component must reidentify vehicles when they pass by a camera. Detection is performed by a blob tracker and the images are registered by aligning the centroid of the object mask (the cameras are located approximately perpendicular to the road). We tested our algorithm on a subset of data collected from 33 cameras and 1000's of unique vehicles, by learning an identifier generating function ($\mathcal{H}_i d$) for each camera pair (this way, the system incorporates the typical distortions that a vehicle undergoes between these cameras). Typical equal error rates for our classification cascade are 3-5% for near lane (vehicle length $\sim$140 pixels) and 5-7% for far lane ($\sim$60 pixels), again using 2-3 patches on average. Whole object comparison (several different techniques) and using patches without hyper-features generally resulted in error rates that were twice as large.

References

[1] Anonymous. Learning hyper-features for visual identification. *Included as Supplementary Material*.

[2] Tamara L. Berg, Alexander C. Berg, Jaety Edwards, Michael Maire, Ryan White, Yee Whye Teh, Erik Learned-Miller, and David A. Forsyth. Names and faces in the news. *CVPR*, 2004.

[3] D. Bolme, R. Beveridge, M. Teixeira, and B. Draper. The csu face identification evaluation system: Its purpose, features and structure. *ICVS*, 2003.

[4] R. Diamond and S. Carey. Why faces are and are not special: An effect of expertise. *Journal of Experimental Psychology*, Gen(115):107–117, 1986.

[5] George H. John, Ron Kohavi, and Karl Pfleger. Irrelevant features and the subset selection problem. In *International Conference on Machine Learning*, pages 121–129, 1994. Journal version in AIJ, available at http://citeseer.nj.nec.com/13663.html.

[6] W.F. Kibble. A two-variate gamma type distribution. *Sankhya*, 5:137–150, 1941.

[7] Robert Fergus Li Fei-Fei and Pietro Perona. A Bayesian approach to unsupervised one-shot learning of object categories. In *International Conference on Computer Vision*, 2003.

[8] M. Welling M. Weber and P. Perona. Unsupervised learning of models for recognition. *ECCV*, 2000.

[9] P. McCullagh and J. A. Nelder. *Generalized Linear Models*. Chapman and Hall, 1989.

[10] E. Miller, N. Matsakis, and P. Viola. Learning from one example through shared densities on transforms. In *IEEE Computer Vision and Pattern Recognition*, 2000.

[11] Michael Tarr and Isabel Gauthier. FFA: A flexible fusiform area for subordinate-level visual processing automatized by expertise. *Nature Neuroscience*, 3(8):764–769, 2000.

[12] Michel Vidal-Naquet and Shimon Ullman. Object recognition with informative features and linear classification. In *International Conference on Computer Vision*, 2003.