A Design Principle for Coarse-to-Fine Classification

Sachin Gangaputra
Johns Hopkins University
Baltimore, MD 21218
sachin@jhu.edu

Donald Geman
Johns Hopkins University
Baltimore, MD 21218
geman@jhu.edu

Abstract

Coarse-to-fine classification is an efficient way of organizing object recognition in order to accommodate a large number of possible hypotheses and to systematically exploit shared attributes and the hierarchical nature of the visual world. The basic structure is a nested representation of the space of hypotheses and a corresponding hierarchy of (binary) classifiers. In existing work, the representation is manually crafted. Here we introduce a design principle for recursively learning the representation and the classifiers together. This also unifies previous work on cascades and tree-structured search. The criterion for deciding when a group of hypotheses should be "retested" (a cascade) versus partitioned into smaller groups ("divide-and-conquer") is motivated by recent theoretical work on optimal search strategies. The key concept is the cost-to-power ratio of a classifier. The learned hierarchy consists of both linear cascades and branching segments and outperforms manual ones in experiments on face detection.

1. Introduction

Promising results have been obtained in recent work using hierarchical search and template-matching [3, 8, 10, 22] and attentional cascades [6, 7, 23] to detect instances of generic object classes in greyscale images. By examining many hypotheses (e.g., object categories and poses) at the same time, and by exploiting the simple observation that the background is statistically dominant (i.e., the most likely explanation in any given region), computation is focused on ambiguous areas of the image, yielding small error rates as well as rapid scene parsing.

The advantage of hierarchical representation is modularity: the global problem is decomposed into more tractable sub-components and one monolithic classifier is replaced by a hierarchy of classifiers, each dedicated to a specific subset of hypotheses sharing common properties. Indeed, the construction of the hierarchy relies on the fact that there exist natural groupings amongst object instances, i.e., common "attributes," such as shape, color, scale and location. (In related work, these attributes are actual "parts" of objects and recent work on spatial arrangements of parts [2, 5], shared parts and features [15, 21], and compositional vision [12] is also motivated by exploiting shared attributes for efficient learning and representation.) In work on coarse-to-fine classification, the attributes are employed to recursively partition the set of hypotheses into ever finer and more homogeneous subsets or "cells". The final hierarchy is then a sequence of nested partitions.

The construction is hardly unique; there are a great many ways to recursively decompose a set of hypotheses. In prior work (e.g., [8]), the hierarchy was manually designed, selecting one attribute at a time as the basis for splitting. Notably absent is a principled construction. Moreover, it is not clear that building the representation should be separated from learning the classifiers and measuring their performance, or, for that matter, from anticipating the order in which they will be applied to the image data. Furthermore, an ad hoc construction may no longer be feasible in extending this approach to accommodate many more object categories, presentations and complex descriptions.

Automatically designing a hierarchy evokes a fundamental question: given several hypotheses, should they be explored simultaneously or separately? More specifically, given a set A of active classes or hypotheses, and several ways to break A down into subsets (including keeping A intact), what is the most efficient way to test if one of the hypotheses in A is true? Of course this depends on somehow equalizing for cost and accuracy among the alternatives.

The design principle we adopt is motivated by recent theoretical work [4, 13] on optimal search strategies. In that work, once a hierarchy of classifiers is given, a sufficient condition for the optimality of coarse-to-fine search is expressed in terms of the ratio of cost-to-power of each classifier. Here, we use this criterion to iteratively construct the hierarchy in the first place. That is, given a leaf cell in the pending hierarchy, and given several candidate splits, including the trivial partition which keeps the cell intact, we choose the partition which minimizes the ratio of cost

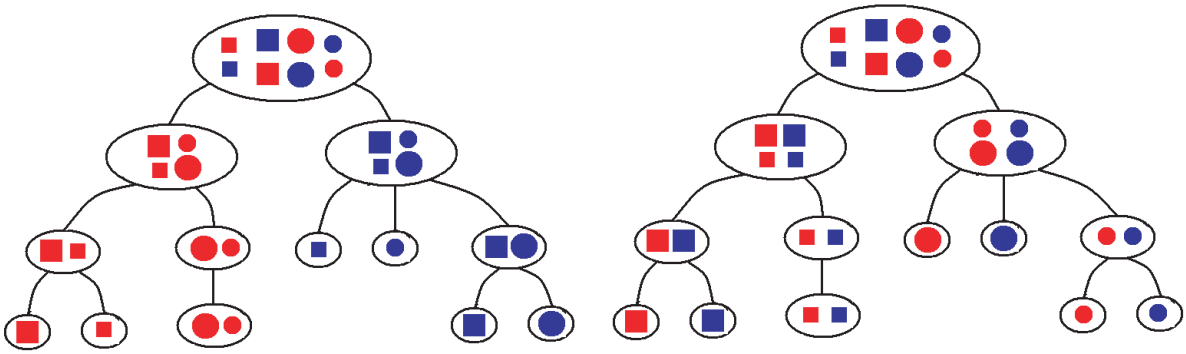

Figure 1. Two of many possible hierarchical representations based on the attributes "shape", "color" and "size".

to power. In particular, since a classifier was constructed for every current terminal cell at some previous iteration, at every step we are deciding between retesting ("cascade") and splitting ("divide-and-conquer"). Retesting does not produce the same classifier since the negative population changes.

This method is then entirely different from bottom-up procedures based on agglomerative clustering and similarity measures. We build the hierarchy of subsets and classifiers at the same time, top-down and recursively. The resulting architecture differs significantly from those manually constructed, especially in dedicating resources to difficult subsets of hypotheses (e.g., small faces). It also unifies previous work on cascades and coarse-to-fine search.

In Section 2, we outline the statistical framework for a hierarchical representation and search. The motivation for our design principle and the algorithm for constructing the hierarchy are presented in Section 3, still in a general context. Then, in Section 4, we illustrate these ideas with experiments in face detection based on Adaboost learning and finally draw some conclusions in Section 5.

2. Statistical Framework

Let $\mathcal{Y}$ denote the set of "hypotheses" of interest. For example, each hypothesis might be an object category and geometric pose. The objective is to determine the true explanation, say Y, of the observed data, e.g., a subimage. The alternative to $Y \in \mathcal{Y}$ is $Y = 0$, denoting "other" or "background" which is typically *a priori* the most likely explanation. Given a subimage I corresponding to an unlabeled shape, the goal is to efficiently determine whether or not $Y(I) \in \mathcal{Y}$. Rather than build a single classifier to test $Y \in \mathcal{Y}$ vs. $Y = 0$, which is likely to be inaccurate when $\mathcal{Y}$ is heterogeneous, or build only individual classifiers to check the elements of $\mathcal{Y}$ one at a time ("template-matching"), which is likely to be expensive, the search is accomplished by a dynamical "coarse-to-fine" process. This is based on grouping hypotheses with common properties and checking entire subsets simultaneously, proceeding systematically from coarse-grained to fine-grained explanations. When properly designed, the fine classifiers are rarely implemented.

2.1. Efficient Representation

The main assumption is that there are natural groupings $A \subset \mathcal{Y}$ among the hypotheses and that testing for these groupings – or "attributes" – is far more efficient than testing for arbitrary subsets or testing for many hypotheses individually. For example, the attributes might represent certain physical properties of the objects, such as "color", "shape" and "size," as illustrated in Fig. 1, an idealized example with eight elements in $\mathcal{Y}$.

The natural data structure to represent $\mathcal{Y}$ based on these attributes is then tree-structured and multi-resolutional, with subsets of hypotheses corresponding to varying levels of precision. We denote this by

$$\mathcal{H}_{attr} = \{A_\xi, \xi \in T\} \tag{1}$$

where T is a tree graph. Fig. 1 shows two such representations. In the applications to object detection in [1, 8, 19] and elsewhere, the sets A_ξ for nodes near the root are quite heterogeneous with respect to both category and pose, whereas near the leaves they represent homogeneous subsets, for instance a single category or even sub-category over a small range of poses (e.g., position, scale and orientation), or several sub-categories at poses which render the shapes nearly identical.

2.2. Efficient Search

There is a binary classifier $X_\xi = X_{A_\xi}$ for every cell A_ξ, $\xi \in T$. The family is denoted by

$$\mathcal{H}_{test} = \{X_\xi, \xi \in T\}.$$

The classifier X_ξ represents a "test" for the hypothesis $H_\xi : \{Y \in A_\xi\}$ against a suitable alternative hypothesis

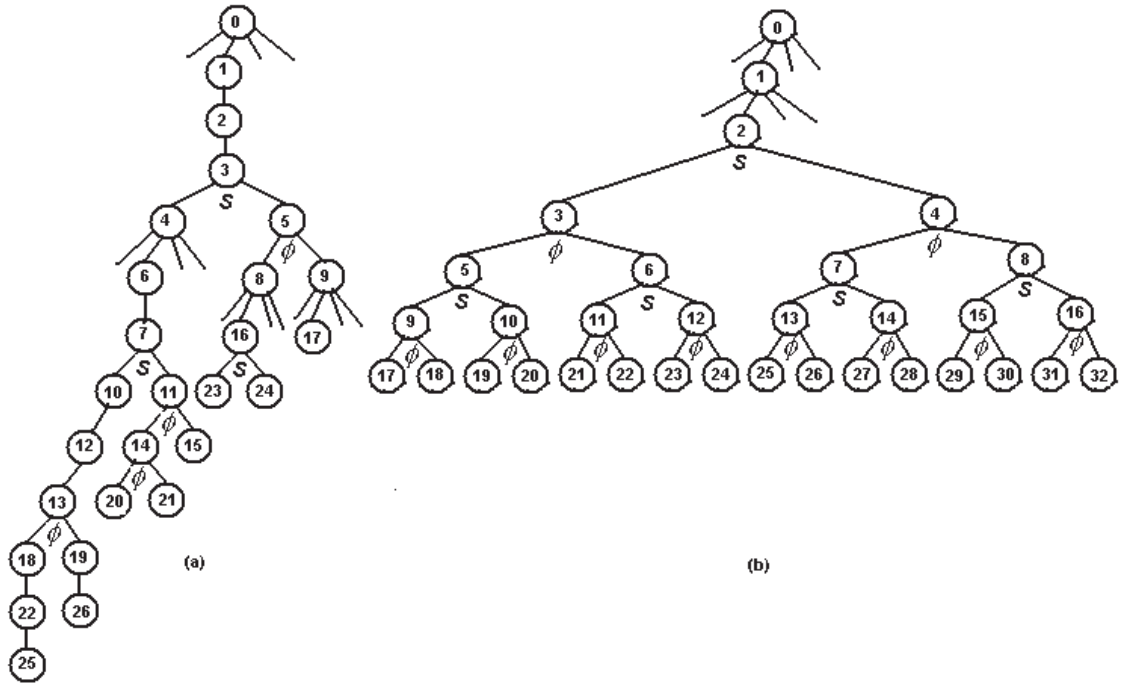

Figure 2. (a) The first 26 classifiers of the learned hierarchy; (b) A manually designed hierarchy. The numbers indicate the order in which the nodes are added to the evolving structure. The labels s and ϕ refer to splits based on scale and tilt, respectively. The quaternary splits are on location.

$H_{alt(\xi)} : B_{alt(\xi)} \subset \{Y \notin A_\xi\}$. In practice, these classifiers are induced from positive and negative training examples using some learning algorithm, for example, naive Bayes [8], Adaboost [9] or support vector machines [19]. However, the framework is largely algorithm-independent. The sole restriction is that the corresponding decision surface can be adjusted in order to accommodate a (nearly) null false negative constraint, i.e., $\alpha(X_A) = P(X_A = 0|Y \in A) = 0$ for every classifier in $\mathcal{H}_{test}$. This can generally be accomplished *at the expense of low selectivity (i.e., low power)* by adjusting some bias or threshold.

The search strategy is breadth-first, coarse-to-fine. That is, starting from the root, the classifiers are executed sequentially and adaptively, with X_ξ performed if and only if all ancestor tests are performed and are positive. More specifically, X_ξ is performed if and only if $X_\eta = 1$ for every node $\eta \in \mathfrak{A}(\xi)$, where $\mathfrak{A}(\xi)$ is the set of nodes which are between the parent of ξ and the root, inclusively. In particular, if the root classifier is negative, the procedure terminates and "background" is declared. Similarly, the search terminates if the test at the root is positive but all its child tests are negative. This promotes extremely efficient computation since, generally, large subsets of hypotheses are simultaneously pruned. The natural alternate hypotheses $B_{alt(\xi)}$ at node ξ accounts for this: whenever X_ξ is implemented, the distribution of the image data should be conditioned on the event

$$B_{alt(\xi)} = \{Y \notin A_\xi\} \cap \{X_\eta = 1 \forall\, \eta \in \mathfrak{A}(\xi)\}. \quad (2)$$

Notice that conditioning the null hypothesis $\{Y \in A_\xi\}$ on the event that the test X_ξ is performed is superfluous due to the null false negative constraint.

Coarse-to-fine search results in a set of detections:

$$D = \{y \in \mathcal{Y} : X_{A_\xi} = 1 \;\; for \;\; all \;\; \xi \;\; such \; that \; y \in A_\xi\}.$$

These are all the hypotheses not ruled out by any test performed. Equivalently, the set of detections is the union of the leaves of $\mathcal{H}_{attr}$ which represent the terminal point of a complete chain of positive responses.

This strategy is highly efficient because most image areas attract little computation. It has been examined both mathematically and in applications. It is theoretically optimal under certain assumptions about the cost and power of the classifiers and their statistical dependency structure under the background hypothesis. However, in all previous implementations, $\mathcal{H}_{attr}$ is manually constructed, and made independently of $\mathcal{H}_{test}$.

2.3. Alternative Hierarchies

Manual construction is based on more or less arbitrary choices. In general, there is no unique way of ordering the splitting process because the attributes often define independent physical properties of the objects. Both hierarchies

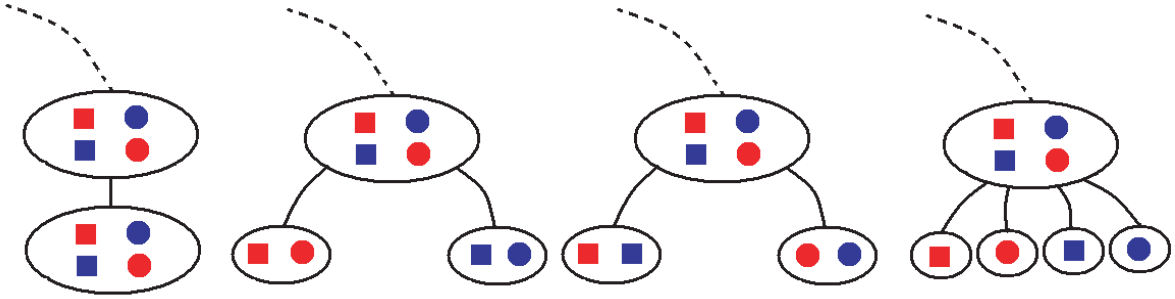

Figure 3. Four possible decompositions of a group of hypotheses. Choosing the best one is based on the ratio of the cost to the power of the resulting classifier.

in Fig. 1 might appear reasonable. The hierarchy used for face detection in [8] is shown in Fig. 2(b): The attributes are position (p), scale (s) and tilt (ϕ) and the splits are either quaternary in position or binary in scale and tilt. Moreover, in previous work, a cell A_ξ is either never "retested" or only retested, as in the cascades in [23]. We argue that distinguishing between the two hierarchies in Fig. 1 should involve the performance of the classifiers that will be built, especially in terms of the efficiency *per unit cost* in rejecting the background hypothesis. In particular, given two hierarchies of equal computational complexity, we should favor the one with fewer average false positives, i.e., a smaller expected $|D|$ under the background hypothesis.

3. Hierarchy Design

Our objective is to construct $\mathcal{H}_{attr}$ and $\mathcal{H}_{test}$ simultaneously from data. We assume we have some learning algorithm $(A, \mathcal{L}) \rightarrow X_A$ for constructing a binary classifier X_A for any subset $A \subset \mathcal{Y}$ from a set of training examples $\mathcal{L} = \mathcal{L}_+ \cup \mathcal{L}_-$. Here, $\mathcal{L}_+ \sim \{Y \in A\}$ and $\mathcal{L}_- \sim B_{alt(A)} \subset \{Y \notin A\}$ denote the set of positive and negative training examples, respectively. We also assume that each X_A can be manipulated to have a very small missed detection rate, i.e., $\alpha(X_A) = P(X_A = 0 | Y \in A) \approx 0$, at the possible expense of a considerable false detection rate. Each classifier X_A is then largely characterized by its cost $c(X_A)$ and power $\beta(X_A)$. The cost $c(X_A)$ reflects the computational or processing expense incurred in performing the test; this can typically be measured in terms of dedicated CPU time. The power $\beta(X_A)$ is the selectivity of the test in the standard sense: $\beta(X_A) = P(X_A = 0 | B_{alt(A)})$. Notice that $1 - \beta$ represents the false positive rate.

3.1. The Cost-to-Power Ratio

In order to rank candidate splits we would like to have a single parameter which captures the overall efficiency of a classifier in the context of the hierarchy. For this purpose, we borrow a performance metric from [4] and [13]. In that work the objective is to determine conditions under which coarse-to-fine strategies minimize total computation in determining D [4] or determining if $D = \emptyset$ [13]. In both cases, the hierarchy of subsets of $\mathcal{Y}$ and the hierarchy of classifiers are assumed to be given and there is no analysis of how they might have been constructed. The basic idea, as originally set forth in [13], is that the cost-to-power ratio is the right way to normalize the immediate processing cost in order to account for additional processing due to false positives. In [4] it is shown that, under certain assumptions about the probability distribution under which mean total computation is measured, a sufficient condition for the optimality of coarse-to-fine search (as opposed to any other way of visiting some or all the nodes of $\xi \in T$ and performing X_ξ) is that

$$\forall \xi \in T, \ \frac{c(X_\xi)}{\beta(X_\xi)} \leq \sum_{\eta \in \mathcal{C}(\xi)} \frac{c(X_\eta)}{\beta(X_\eta)} \tag{3}$$

where $\mathcal{C}(\xi)$ denotes the direct children of ξ in T. In this work, our objective is essentially reversed: we assume from the outset that the hierarchy will be processed coarse-to-fine and we wish to find a principled way of constructing it. Extending the cost-to-power ratio to a candidate partition provides a natural metric.

3.2. Ranking Candidate Splits

The construction is recursive. Suppose, at step k, we have constructed $\mathcal{H}_{attr}^{(k)}$ and $\mathcal{H}_{test}^{(k)}$. Let $A = A_\xi$ be the subset of hypotheses at a leaf ξ of $T^{(k)}$. The particular leaf chosen for expansion is the one with the currently highest estimated false positive rate (see §3.3). Suppose there are several candidate partitions of A. We will write $\Lambda(A) = \{A_1, A_2, ..A_n\}$ to denote such a partition: the $A_i's$ are pairwise disjoint and $\cup A_i = A$. One of these candidates is the trivial partition $\Lambda(A) = \{A\}$. Notice that at least one classifier, namely X_ξ, has already been built for deciding between $\{Y \in A\}$ against the appropriate alternative. Consequently one choice is to build another classifier for this sub-problem, corresponding to a (partial) cascade.

The learning algorithm will not yield the same classifier because the alternative hypothesis has changed and hence the set of negative examples is different.

Fig. 3 illustrates four different partitions for a cell in our fictional example; from left to right, the four splits correspond to "retesting", splitting on color and shape, and testing individual hypotheses (a form of "template-matching").

In order to choose among these, and anticipating the processing strategy, we define the cost and power of a partition Λ as follows. For each $A_i \in \Lambda$ a classifier X_{A_i} is built using our learning algorithm. Now define a single classifier for the partition Λ:

$$X_\Lambda = \begin{cases} 1 & if\ X_{A_i} = 1\ for\ some\ i \\ 0 & otherwise \end{cases} \quad (4)$$

Hence X_Λ responds positively if *any* of its components is positive, i.e., if processing continues. The cost of the composite test is clearly $c(X_\Lambda) = \sum_i c(X_{A_i})$ since *each* of the classifiers X_{A_i} will be performed whenever the parent X_ξ is positive. The power of the composite test is

$$\begin{aligned} \beta(X_\Lambda) &= P(X_\Lambda = 0 | B_{alt(\xi)}) \\ &= P(X_{A_i} = 0, i = 1, ..., n | B_{alt(\xi)}) \end{aligned}$$

where $B_{alt(\xi)}$ was defined in (2). Hence the composite power is the probability under the alternative hypothesis that all the tests fail, which is the desired outcome in that case.

One of the conditions in [4] leading to the criterion (3) is that the classifiers represent independent random variables under the alternative hypothesis (taken there as the background distribution). We do not want to make this assumption, i.e., we do not assume that $P(X_{A_i} = 0, i = 1, ..., n | B_{alt(\xi)}) = \prod_{i=1}^n P(X_{A_i} = 0 | B_{alt(\xi)})$. In practice, we simply estimate the composite power from training data.

Finally, we choose the partition Λ that minimizes the cost-to-power ratio $c(X_\Lambda)/\beta(X_\Lambda)$. Specifically, given partitions $\Lambda_1, \Lambda_2, ..., \Lambda_m$ of A, choose:

$$i^* = \arg\min_{1 \le i \le m} \frac{c(X_{\Lambda_i})}{\beta(X_{\Lambda_i})}. \quad (5)$$

The subset A is then decomposed into $|\Lambda_{i^*}|$ children and the elements of Λ_{i^*} are adjoined to $\mathcal{H}_{attr}^{(k)}$ to form $\mathcal{H}_{attr}^{(k+1)}$. Similarly, the corresponding classifiers are added to $\mathcal{H}_{test}^{(k)}$.

3.3. Special Case: Recursive False Positive Minimization

The above formulation is very general. It accommodates constructing classifiers of variable cost and power. Alternatively, one can compare candidate splits by equalizing for one of these quantities and optimizing over the other. In particular, we can normalize for cost by arranging to have $c(X_{\Lambda_i}) \equiv c$ for each candidate partition Λ_i. This would occur, for example, if the cost of a classifier were proportional to the size of the underlying set A of hypotheses, i.e., if $c(X_A) \propto |A|$ for every $A \subset Y$, in which case we obviously have $c(X_{\Lambda_i}) \propto |A|$ for every partition. In practice, this can be arranged by controlling some parameter in the learning algorithm, for instance the number of features employed. Once the cost is normalized, the chosen partition is the one that maximizes power (equivalently, results in the smallest false positive rate).

As noted above, the construction is iterative and the hierarchy is expanded one node at a time. Given $\mathcal{H}_{attr}^{(k)}$, we start by identifying the leaf node with the highest overall false positive rate. That is, for each current leaf ξ, we estimate $P(X_\xi = 1, X_\eta = 1, \eta \in \mathfrak{A}(\xi) | background)$, where "background" denotes the event $Y \notin \mathcal{Y}$. This is very easy in practice as it only requires sending background samples down the existing hierarchy and counting the number reaching each leaf.

4. Application to Face Detection

To measure the efficacy of our design principle we choose one of the problems in object detection which has previously been investigated using hierarchies of classifiers. This includes work on detecting printed characters [1, 10, 11], buildings [14] and faces [7, 8, 19, 23]. We focus on face detection, which has been widely studied in the computer vision literature. The problem is to find all instances of frontal, upright faces in greyscale scenes. (More recently, this has been extended to include profile detection.) In addition to the work cited above, existing methods include neural networks [16, 18], support vector machines [17] and Gaussian models [20]. The best ROC curves generated on the CMU+MIT test set are remarkably similar. For instance, the current state-of-the-art is order one false positive per image at approximately ninety percent detection rate. In addition, detection is sometimes very rapid, in fact essentially real-time [23]. However, most methods require very large training sets, e.g., thousands or even tens of thousands of sample faces, in order to achieve this performance.

4.1. Representation

Since we are searching for instances from a single object class, the set of hypotheses corresponds to possible geometric poses. (Other instantiation parameters could be included.) Specifically, the pose will refer to the position, tilt and scale of a face, denoted $\theta = (p, s, \phi)$, where p is the midpoint between the eyes, s is the distance between the eyes and ϕ is the angle between the vertical and the segment orthogonal to the segment joining the eyes. Relative to a subimage I, the set $\mathcal{Y}$ of hypotheses is the set of poses

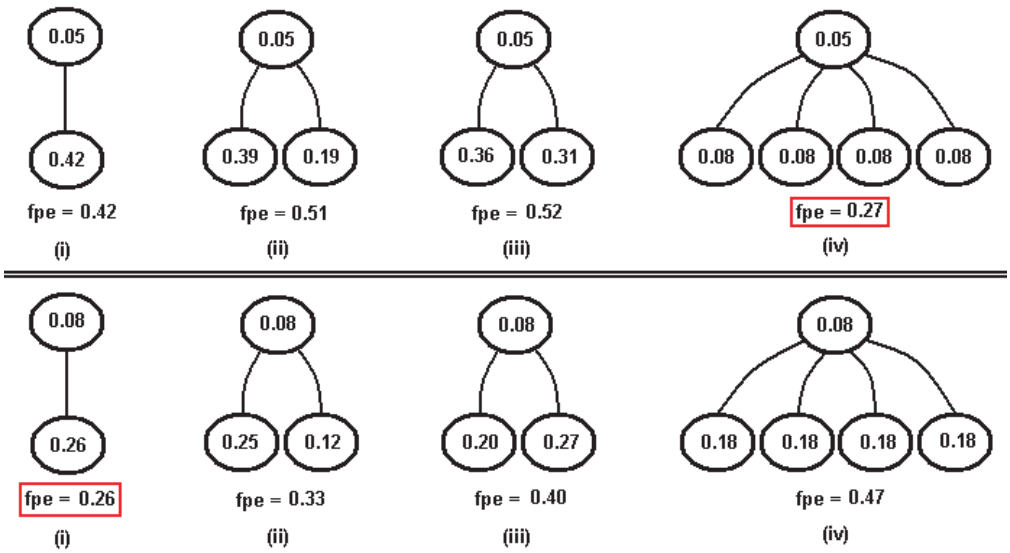

Figure 4. The first two steps in learning the hierarchy. Top panel: Four possible decompositions of the root node, whose false positive rate is 0.05. The conditional false positive rates of both the individual classifiers (in circles) and the aggregate classifier (denoted fpe) are given. Bottom panel: Choices for the next level given the quaternary split is chosen for the root.

of a single face within I such that the position is restricted to an 8×8 window centered in I and the scale and tilt are appropriately restricted. More specifically, $\mathcal{Y} = \{(p, s, \phi) \in \Re^4 : p \in [-4, 4]^2, s \in [8, 16], \phi \in [-20^o, 20^o]\}$. The background hypothesis is that there is no face centered in the given image at these scales and tilts.

A scene is processed by visiting non-overlapping 8×8 blocks, processing the surrounding image data to extract features (e.g., edges) and classifying this subimage according to the coarse-to-fine search strategy described in §2.2. Faces at larger scales are detected by downsampling the image and repeating the search process. With four levels of downsampling, faces of size $8 \leq s \leq 128$ are detected.

In existing work a hierarchy $\mathcal{H}_{attr}$ is constructed by choosing some decomposition of $\mathcal{Y}$ into nested partitions based on splits involving the three pose parameters. For example, one such hierarchy results from two quaternary splits in location (from 8×8 to 4×4 to 2×2) and then two alternating binary splits on tilt and scale, yielding a pose hierarchy with seven levels. One section is illustrated in Fig. 2(b). For this section of T, $\mathcal{H}_{test}$ would consist of 33 binary classifiers including the root classifier, which is dedicated to detecting faces invariantly over $\mathcal{Y}$.

4.2. Baseline Learning Algorithm

Our positive training samples are synthesized from the standard ORL database which contains ten pictures for each of forty individuals for a total of 400 images. For each cell A_ξ in $\mathcal{H}_{attr}$, we synthesize 1600 faces with poses more or less uniformly sampled from A_ξ. The negative (non-face) training examples for constructing X_ξ are obtained by processing a large set of "background" images and collecting the false positives which arrive at ξ, up to a maximum of 6000. This ensures that the non-face instances used for training at every node represent a sample of images responding positively to all the preceding classifiers X_η for $\eta \in \mathfrak{A}(\xi)$. In this way, we are training against those particular non-faces that increasingly resemble faces.

The same learning algorithm is applied to each cell; only the training set changes. We use binary Adaboost [9] to build each test. Our features are binary, oriented edge fragments, as in [1, 8]. Other features and learning algorithms could have been used in place of edges and Adaboost, as, for example, in [19], where each X_ξ is a support vector machine based on wavelet coefficients. Finally, the null false negative constraint is enforced by adjusting parameters based on a held-out validation set; this promotes better generalization (less over-fitting) than using the whole training set for learning everything about $\mathcal{H}_{attr}$ and $\mathcal{H}_{test}$.

4.3. Learned Construction

We use the procedure outlined in §3.3. The hierarchy $\mathcal{H}_{attr}$ is recursively built starting from the root node and comparing candidate partitions based on equalizing the total cost and estimating the conditional false positive rate. Specifically, we consider four possible decompositions of A_ξ at each current leaf node ξ: (i) no decomposition (i.e., retesting); (ii) a binary split in scale; (iii) a binary split in orientation; and (iv) a quaternary split in position. Every time option (i) has the best cost-to-power ratio, we are effectively building a linear "cascade" segment. The other options lead to binary and quaternary forks. Sometimes it

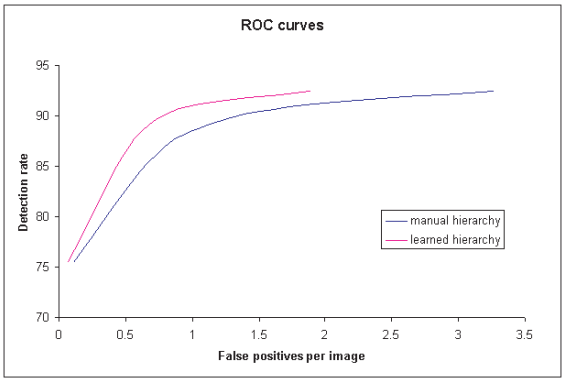

Figure 5. ROC curves on the CMU+MIT testset

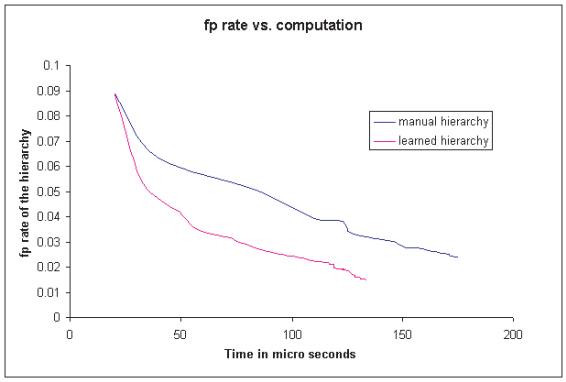

Figure 6. Estimated false positive rate vs. average computational time (to execute the hierarchy) for varying numbers of classifiers added to the hierarchy; the order is given in Fig. 2(a).

is more efficient to retest and sometimes it is more efficient to decompose the problem into simpler pieces.

The cost $c(X_A)$ of classifier X_A is assumed proportional to the number of features; overhead charges are ignored. In order to equalize $c(X_\Lambda)$ the same for every partition Λ, we use 400 features for option (i), 200 features for options (ii) and (iii), and 100 features for option (iv). The (local) false positive rates of X_Λ and the (global) false positive rates of each pending leaf in the hierarchy are empirically determined from the training data.

The first twenty-seven classifiers in the learned hierarchy are shown in Fig. 2(a). Recall that at each step, the algorithm visits the leaf node that currently performs the worst in terms of filtering the background. The ordering in Fig. 2(a) reflects this process.

The first two steps in the process are illustrated in Fig. 4. We start by building a classifier for $A = \mathcal{Y}$, the whole pose set. Its false positive rate is 0.05 and this test is represented by the top circle in the upper half of Fig. 4. The four decompositions correspond to (i)-(iv). The value fpe denotes the false positive rate of the partition. The individual false positive rates of the classifiers are also shown; notice that the classifiers within a partition are not independent. (For instance, in (iv), $1 - 0.27 \neq (1 - 0.08)^4$.) But they are surprisingly close.

From Fig. 4, we deduce that the best strategy for decomposing the root is the quaternary split on location. The bottom half of Fig. 4 depicts what happens when the new leaves are examined; these correspond to restricting the position to a 4×4 region and leaving scale and orientation unrestricted. In this case, the best decomposition is (i); only 26% of the background images which survive the classifier at the root, denoted "0" in Fig. 2(a), also survive the restricted location test, which is denoted by "1" in Fig. 2(a).

4.4. Analysis

The resulting hierarchy contains both linear cascades and branching forks. Unlike the ones manually designed, the learned hierarchy is asymmetric and non-uniform with respect to the pose space. We notice longer chains for small scales, corresponding to a more concentrated effort to detect smaller faces. This is natural, as a small face is more difficult to discern in a cluttered background. In some cases, no decomposition "significantly" reduces the number of false positives. The maximum level reached (i.e., number of classifiers in the branch from root to leaf) is 23. For example, the resolution in location stops at 2×2 whereas scales are finely resolved.

4.5. Experimental Results

The system is implemented in C++ on a standard Pentium 4 1.8GHz PC. Performance is estimated using the CMU+MIT [18, 20] test set. As elsewhere, images with faces displaying out-of-plane rotations are excluded. There are then 164 images with 556 faces exhibiting tilts in the range -20^o to 20^o range. Some are sketches, which are exceedingly difficult to detect. As in cited work on cascades, processing a 320×240 image takes only a small fraction of a second.

	Detection	FP / image
Learned hierarchy	89.1%	0.67
Manual hierarchy	89.1%	1.11
Viola-Jones [23]	90.8%	0.73
Rowley-Baluja-Kanade [18]	89.2%	0.73

Table 1. Performance measures for various face detection systems.

In order to properly compare the learned hierarchy (Fig. 2(a)) to the manual hierarchy (Fig. 2(b)), the classifiers in both are constructed with the same protocol – the same training database, feature set, learning algorithm, and so forth. The performance of the two systems are depicted in Fig. 5 with ROC curves. During construction, both the current overall false positive rate and average computational

time are estimated at each iteration (i.e., expansion of a node). Error is plotted against computation for each method (manual vs. learned) in Fig. 6; each point on the curve corresponds to expanding the corresponding hierarchy. For the same average computation, the learned hierarchy has a substantially lower false positive rate.

We also compare the new method with other face detection systems in Table 1. The results are also comparable to other well-known systems. It should be noted that different systems are tested on slightly different subsets of the MIT+CMU dataset, but the results are nonetheless very similar. The test subset we use is more general because it contains faces with varying tilts.

5. Conclusion

We have taken a step in rendering coarse-to-fine classification a more data-driven procedure. In so doing, we have proposed a general principle for deciding among several ways of examining a group of hypotheses; in our applications, these represent possible explanations of image data. The alternatives include holistic examination (searching for everything at once) and versions of "divide-and-conquer" (conducting multiple searches for coherent subsets). The basic principle is to assign a numeric measure to each candidate decomposition, namely the cost of the corresponding aggregate classifier divided by its statistical power. This quantity plays a pivotal role in recent work on optimal search strategies; further reconciling that work with our work is a subject of ongoing research.

Experiments in face detection show a significant improvement over manually-constructed hierarchies after adjusting for all other factors. Although high-performance face detection was not our primary objective, additional improvements, notably fewer false positives at any given detection rate, might be obtained in several ways. Two promising avenues are the use of more powerful features than the simple binary edge variables utilized here and training the classifiers in the hierarchy with a much larger set of faces.

References

[1] Y. Amit, D. Geman, and X. Fan. A coarse-to-fine strategy for multi-class shape detection. *IEEE Transactions PAMI*, 28:1606–1621, 2004. 2, 5, 6

[2] Y. Amit and A. Trouve. Pop: Patchwork of parts models for object recognition. Technical report, University of Chicago, 2004. 1

[3] S. Baker and S. Nayar. Pattern rejection. *Proceedings IEEE CVPR*, pages 544–549, 1996. 1

[4] G. Blanchard and D. Geman. Sequential testing designs for pattern recognition. *Annals of Statistics*, 33:155–1202, 2005. 1, 4, 5

[5] M. Burl and P. Perona. Recognition of planar object classes. *Proceedings IEEE CVPR*, pages 223–230, 1996. 1

[6] M. Elad, Y. Hel-Or, and R. Keshet. Pattern detection using a maximal rejection classifier. *Pattern Recognition Letters*, 23(12):1459–1471, 2002. 1

[7] C. Eveland, D. Socolinsky, C. Priebe, and D. Marchette. A hierarchical methodology for class detection problems with skewed priors. *Journal of classification*, 2005. 1, 5

[8] F. Fleuret and D. Geman. Coarse-to-fine face detection. *International Journal of Computer Vision*, 41:85–107, 2001. 1, 2, 3, 4, 5, 6

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European Conference on Computational Learning Theory*, pages 23–37. Springer-Verlag, 1995. 3, 6

[10] D. Gavrila. Multi-feature hierarchical template matching using distance transform. *Proceedings IEEE ICPR*, 1998. 1, 5

[11] S. Geman, K. Manbeck, and E. McClure. Coarse-to-fine search and rank-sum statistics in object recognition. Technical report, Brown University, 1995. 5

[12] S. Geman, D. Potter, and Z. Chi. Composition systems. *Quaterly of Applied Mathematics*, LX:707–736, 2002. 1

[13] F. Jung. *Reconnaissance d'objects par focalisation et detection de changements*. PhD thesis, Ecole Polytechnique, Paris, France, 2001. 1, 4

[14] F. Jung. Detecting new buildings from time-varying aerial stereo pairs. Technical report, IGN, 2002. 5

[15] S. Krempp, D. Geman, and Y. Amit. Sequential learning with reusable parts for object detection. Technical report, Johns Hopkins University, 2002. 1

[16] R. Osadchy, M. Miller, and Y. LeCun. Synergistic face detection and pose estimation with energy-based model. In *Advances in Neural Information Processing Systems (NIPS 2004)*. MIT Press, 2005. 5

[17] E. Osuna, R. Freund, and F. Girosi. Training support vector machines: an application to face detection. *Proceedings IEEE CVPR*, pages 130–136, 1997. 5

[18] H. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions PAMI*, 20:23–38, 1998. 5, 7

[19] H. Sahbi. *Coarse-to-fine support vector machines for hierarchical face detection*. PhD thesis, Versailles University, 2003. 2, 3, 5, 6

[20] K. Sung and T. Poggio. Example-based learning for view-based face detection. *IEEE Transactions PAMI*, 20:39–51, 1998. 5, 7

[21] A. Torralba, K. Murphy, and W. Freeman. Sharing features: efficient boosting procedures for multiclass object detection. *Proceedings IEEE CVPR*, pages 762–769, 2004. 1

[22] S. Ullman, M. Vidal-Naquet, and E. Sali. Visual features of intermediate complexity and their use in classification. *Nature Neuroscience*, pages 1–6, 2002. 1

[23] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. *Proceedings IEEE CVPR*, 2001. 1, 4, 5, 7