Constructive Feature Learning and the Development of Visual Expertise

Justus H. Piater PIATER@CS.UMASS.EDU
Roderic A. Grupen GRUPEN@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts, Amherst, MA 01003 USA

Abstract

We present a framework for learning features for visual discrimination. The learning system is exposed to a sequence of training images. Whenever it fails to recognize a visual context adequately, new features are sought that discriminate further between the true and false classes. Features consist of hierarchical combinations of primitive features (local edge and texture characteristics) that are sampled from example images. The system continues to learn better features even after all recognition errors have been eliminated, similarly to mechanisms underlying human visual expertise. Whenever the probabilistic recognition algorithm returns any posterior class probabilities greater than zero and less than one, the system attempts to find new features that improve discrimination between the classes in question. Our experiments indicate that this procedure tends to improve classification accuracy on independent test images, while reducing the number of features used for recognition.

1. Introduction

The attributes or features available to an inductive machine learning algorithm limit the possible performance of this algorithm on a given task. A given feature set may place a target concept outside of the space of concepts expressible by an algorithm. Moreover, an expressible concept may still be hard to discover by this algorithm; using a different feature set, the same task may be easy to learn by the same algorithm. Since learning algorithms differ in their inductive biases, the same feature set may be well suited for one algorithm but inappropriate for another (John et al., 1994).

Enormous manual effort is often expended to identify useful features. However, any manually designed feature set is subject to at least these two limitations: First, it is not clear how well these features are tuned to any given learning algorithm. Second, the performance of the algorithm is limited by the given features. One way to address both problems is to have a learning algorithm construct its own features. These issues are receiving increased attention in the machine learning literature (Precup & Utgoff, 1998).

This paper addresses the problem of learning object discrimination by a computer vision system. Presently, most feature-based visual recognition systems employ fixed feature sets and suffer from the drawbacks mentioned above. Training a recognition system is most often a clearly demarcated off-line process. After the training phase is complete, further learning is either not possible, or may even be counterproductive because of overfitting problems.

In previous work we introduced a framework for constructing visual features by hierarchical composition of primitive features (Piater & Grupen, 1999). This paper describes in detail an incremental procedure for learning discriminative composite features using a Bayesian network classifier, and introduces our concept of *expert learning*. Since feature learning is driven by the classifier itself, features are constructed that are tuned to the classifier. The procedure is biased to find few but highly discriminative features. Learning can continue even in the absence of any misclassifications by looking for features that have more discriminative power than previously learned features. This is an effective countermeasure against overfitting, and tends to increase correct and to decrease false recognitions on an independent test set, as well as to reduce the number of features employed by the classifier. This behavior resembles that of humans who develop expertise through extensive training, resulting in increased accuracy and faster responses (Tanaka & Taylor, 1991; Gauthier & Tarr, 1997).

The following section summarizes relevant aspects of our visual features, as introduced in earlier work. Section 3 explains Bayesian network classifiers and their use in our system. Incremental feature learning and expert learning are described in Section 4, followed by experimental results.

2. An Infinite Feature Space

Our objective is to learn features such as to avoid the two drawbacks of fixed feature sets mentioned above. We begin by specifying a small set of primitive features that can be combined into compound features according to a small

Proceedings of the 17th International Conference on Machine Learning, Stanford, CA, June 29–July 2, 2000. Morgan Kaufmann.

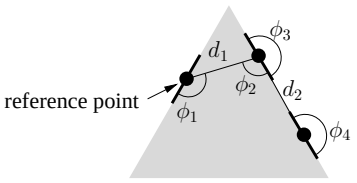

Figure 1. A geometric feature of order 3, composed of three primitives. The feature is defined by the angles ϕ and the distances d, and the orientation of this specific instance is denoted by θ. Each primitive is either an edgel or a texel.

number of rules that will be discussed below. All features that can be represented in this way form an infinite feature *space*. The structural complexity of a feature, i.e. the number of primitive features that form a compound, naturally provides a partial ordering of this space. Our learning procedure searches the feature space beginning with structurally simple features, and considers more complex features as needed (Amit et al., 1997). The underlying assumption is that structurally simple features are easier to discover and have less discriminative potential than complicated features, but are still useful for some aspects of the learning problem.

2.1 Primitive Features and their Composition

In our current system, primitive features are local, oriented appearance descriptors represented as vectors of local filter responses. These filters are oriented derivatives of 2-D Gaussian functions, with orientations chosen such that they form a steerable basis (Freeman & Adelson, 1991). Here, the steerability property permits the efficient computation of filter responses of Gaussian-derivative kernels at any orientation. This technique is used to normalize features for orientation in the image plane, thus achieving rotational invariance (Rao & Ballard, 1995).

Currently, our system uses two specific variants of such descriptors. An *edgel* is encoded as a 2-vector containing the filter responses to the two first-derivative basis filters. It represents the magnitude and orientation of a localized spatial intensity gradient. A *texel* is represented as an 18-vector consisting of the responses to the basis filters of the first three derivatives at two scales. This represents a local texture signature. Like edgels, texels have an associated orientation that is defined by the first derivatives. For more detail, the interested reader is referred to our earlier work (Piater & Grupen, 1999).

Primitive features by themselves are not very discriminative. However, spatial combinations of these can express a wide range of shape and texture characteristics at various degrees of specificity. We employ the following four complementary types of feature composition (Piater & Grupen, 2000): *Geometric* relations are given by the relative angles and distances between the constituent lower-order features (Figure 1). Geometric features are useful for representing e.g. corners, angles, and collinearity. *Topological* relations here refer to relaxed geometric relationships between component features that allow some degree of variability in angles and distances. Topological compound features are more robust to viewpoint changes than are geometric features, at the expense of specificity. *Conjunctive* features assert the presence of their component features without making any statement about their geometric or topological relationship. *Disjunctive* features are considered to be present in a scene if at least one component feature is detected. This can express statements such as "If I see a dial *or* a number pad, I may be looking at a telephone."

Features are computed at various scales, generated by successively subsampling images by factor two. This achieves a certain degree of scale invariance. Moreover, many compositions of edgels are inherently tolerant to changes in scale. For example, the arrangement shown in Figure 1 applies equally to triangles of any size.

Our feature space encompasses an infinite variety of localized contrast descriptors. For practical purposes, its main limitation is its ignorance of color and contours, both of which are highly meaningful to the human visual system.

2.2 Measuring the Value of a Feature

The presence of a given feature $\mathbf{x}^*$ at a point i in the image is denoted by its *strength* $s \in [0, 1]$. For primitive features, this is computed as $s = \max\{0, r(\mathbf{x}^*, \mathbf{x}(i))\}$, where r is the normalized cross correlation function. The value $\mathbf{x}^*$ is a model feature vector, and the function $\mathbf{x}(i)$ returns the corresponding feature vector at location i. For geometric features, the feature vector of the compound feature is the concatenation of the individual feature vectors of the constituent features. In the case of topological and conjunctive features, the strength of the compound feature is the minimum of the strengths of its constituents; for disjunctive features, the maximum is used.

The value of a feature of a given image is the maximum strength of this feature at any location in the image, a real value between zero and one. Finding the maximum strength of a feature in principle involves measuring its strength at each point in the image. For efficiency, we restrict our search for the strongest feature to salient "interest" points that are likely to return a high response (Piater & Grupen, 2000).

3. Multiple Bayes Nets for Recognition

Our system employs Bayesian network classifiers to recognize images based on the features introduced above. This section briefly describes a general Bayes net classifier model and shows how it is applied in our system.

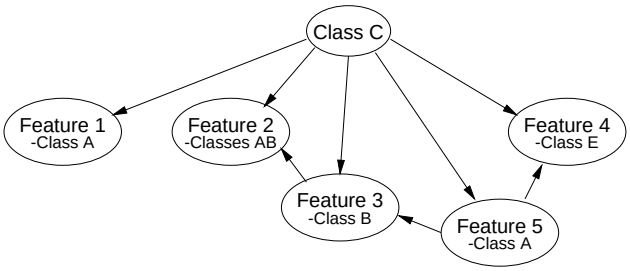

Figure 2. A Bayesian network for a class C. A network such as this is created for each class. Each feature is specialized to distinguish Class C from one or more other classes. Note some interdependent features.

3.1 Bayes Net Classifiers

In a Bayesian network, each node represents a random variable. The network structure specifies a set of conditional independence statements: The variable represented by a node is conditionally independent of its non-descendants in the graph, given the values of the variables represented by its parent nodes. Here, each class is modeled by its own Bayes net. The presence of a visual context (e.g. an object) is modeled as a discrete random variable with two states, *true* and *false*. A visual context gives rise to observable features that are represented by random variables whose distributions are conditioned on the presence of an object of this class. Assuming that the features are conditionally independent given the class, the resulting Bayes net has the topology of a star, with arcs connecting the class node to each of the feature nodes. Given observed feature values, the class priors and conditional feature probabilities, the posterior class probabilities can be inferred by simple application of Bayes' Theorem to each component network.

If some features are not independent, corresponding arcs must be inserted between the appropriate feature nodes. For example, in Figure 2, Feature 3 may participate in a geometric composition with Feature 2, which is also in the feature set. Then, the presence of Feature 3 in an image implies the probable presence of Feature 2. Thus, in the Bayes net there is an arc from node 3 to node 2. An analogous argument holds for topological and conjunctive features, such as Feature 5 in Figure 2, which combines Features 3 and 4. In the case of disjunctive features, the direction of the argument (and that of the additional arrows) is reversed.

To propagate evidence, more sophisticated mechanisms are needed than in the simple case of class-conditional independence. For the purposes of this paper, suffice it to say that after instantiation of some of the variables (nodes) with actually observed values, the net can be brought to *equilibrium* in which the probabilities and observations in the net are consistent. For more detail, the interested reader is referred to the literature on Bayesian networks.

Recall from Section 2.2 that the feature variables are continuous. We split each feature variable into two bins, corresponding to "present" and "not present", using a threshold. This threshold is determined individually for each feature node such as to maximize its discriminative power between its own class and one or more other classes. These other classes are determined by the learning procedure (see Section 4.1 below) and remain fixed over the lifetime of the feature node. The discriminative power of a feature variable given a threshold is measured in terms of the Kolmogorov-Smirnoff distance (KSD). The KSD between two conditional distributions of a random variable is the difference between the cumulative probabilities at a given value of this variable under the two conditions. Maximizing the KSD separates the instances of the two conditions optimally, in the Bayesian sense, using a single cutpoint.

3.2 Recognition

Recognition of a visual context can be performed in the conventional way by first measuring the strength of each feature in the image, setting the feature nodes of the Bayes nets to the corresponding values, and computing the posterior probability of the presence of each class. In this case, the absence of a feature is meaningful to the system. Alternatively, robustness to occlusion can be built into the system by setting only feature nodes corresponding to found features, and leaving the others unspecified. In this case, the posterior probability of these features being present (but occluded) can be easily computed. Incorporating evidence into the net will monotonically increase the posterior probability of a class. In particular, the posterior probability is always greater than or equal to the prior. In this paper, however, negative evidence is incorporated into the net.

Since each class is represented by its own Bayes net, the possibility of multiple objects or contexts present in a scene is built into the system. The presence of each class is determined independently of all others. Those classes with a posterior probability greater than 0.5 are said to be *recognized*. A class is called *true* if it is present in the scene, and *false* otherwise. In the experiments discussed in this paper there is always exactly one true class. Thus, each recognition falls into exactly one of the following categories:

correct: Exactly the true class is recognized.
wrong: Exactly one class is recognized, and this is not the true class.
ambiguous: More than one class is recognized, including the true class. The other recognized classes are false.
confused: More than one class is recognized, but they do not include the true class.
ignorant: No class is recognized.

Instead of computing all feature values at the outset, we compute them one by one and update the Bayesian networks after incorporating each feature. One can quit as soon as confidence in the recognition result exceeds some threshold, e.g. based on the posterior probabilities. Features are processed in decreasing order of informativeness. The informativeness of a feature is defined by the maximum mutual information between a feature F and a given class variable C:

$$I(C,F) = \sum_c \sum_f BEL(c,f) \log \frac{BEL(c,f)}{BEL(c)BEL(f)}$$

The summations range over all possible (discrete) values of the two random variables. $I(C,F)$ expresses the potential of an observation of F to reduce the uncertainty of C, i.e. the entropy of its posterior probabilities. It is computed as described by Pearl (1988) and by Rimey and Brown (1992). As a result, only a fraction of all features are measured in an image during a typical recognition process, even if no confidence threshold on the recognition result is used, because the entropies in the class nodes vanish before all features have been queried.

The feature values measured during each recognition procedure, along with the true class label, are grouped in an *instance vector*. Missing values corresponding to features that were not evaluated are represented by a special 'unknown' value. All instance vectors are stored in an *instance list*. This list is used to update the conditional probability tables of the Bayes nets. Also, presentations of novel training images are counted to estimate the class priors.

4. Feature Learning

In many realistic interactive tasks, it is difficult – if not impossible – to define and acquire representative training data a priori for off-line learning. Therefore, practical learning in the real world should be on-line and incremental. In our assumed scenario, the visual learning agent interacts with the environment in the following ways:

- The agent can acquire images of a scene. Objects present in an image may be of relevance to the agent, which for our purposes is represented as a class label.
- The agent can inquire of the environment what the relevance is of a given image. Presently, this is returned in the form of a supervisory class label. Therefore we here refer to the environment also as the *teacher*. In realistic applications, this feedback could be the result of extensive experience (McCallum, 1995).
- The agent can acquire new images of specific relevance, or request them of the teacher. This is realistic in many applications. For example, an infant can pick up a known object and view it from various viewpoints; or a child receives various examples of letters of the alphabet from a teacher. A robot may aim its camera at known locations, or may pick up a known object from a known location.

These assumptions permit incremental supervised learning and statistical evaluation of new features. The following two sections describe our learning procedure in its basic form, and an extension for developing visual expertise.

4.1 Basic Feature Learning and Evaluation

The agent receives training images one by one. Once a new image is available, it is run through the recognition procedure as described in Section 3.2. Initially, the agent does not know about any classes or features. When it is presented with the first image, it simply remembers the correct answer given by the teacher. When it is shown the second image, it will guess the only class it has seen before.

When the recognition procedure does not produce a correct answer, the system first updates all conditional probability tables according to the instance list. If recognition still fails, a new feature is derived to discriminate the true class from the *mistaken* class (or classes), i.e., those false classes that were recognized as true.

Two cases must be distinguished. In the first case, the true class is not among the recognized classes (wrong, confused, or ignorant recognition). The agent needs to find a new feature in this image of the true class – the *sample image* – that, if detected, will cause this class to be recognized. In the second case, some false classes were recognized in addition to the true class. Here, the agent needs to find a feature in an image of a false class that is not present in the currently misrecognized image, and that, if not detected, will prevent this false class from being recognized. Again, this image is called the sample image. The following procedure describes the first case; the second case is handled analogously.

1. A set of *evaluation images* is retrieved from the environment. This set contains k random views of the true class, and k views of the mistaken class (or classes) with highest posterior probability. Alternatively, one could use all mistaken classes or even all false classes. However, in general it will be much easier to find features that discriminate well between pairs of classes, than features that discriminate between one class and a large number of other classes. The precise value of k is unimportant, but involves a tradeoff that will be discussed below.
2. A new feature is generated, generally by sampling from the sample image. The details of this step will be presented in Section 4.2 below.

3. A new node representing this feature is added to the Bayesian net that models the class of the sample image, along with links from the class node and any links required to or from any other feature nodes, as described in Section 3.1.
4. The values of this feature, as well as any existing features represented by nodes linked to or from the new feature node in the Bayes net, are measured within each evaluation image. The resulting instance vectors are added to the instance list.
5. The new feature node is discretized (as outlined in Section 3.1) such that the cutpoint maximizes the KSD between the true and mistaken classes. Since we want to find features that are characteristic of this class, we require that the value of the new feature is above the cutpoint. If this is not the case, the procedure continues at Step 2.
6. The conditional probability tables of the affected nodes are (re)initialized by counting instances in the instance list.
7. The recognition procedure is run on the sample image, using the added feature. If this image is again misrecognized, the new feature node is removed from the Bayes net. The structural description of the new feature, however, is retained for the duration of this feature search, which proceeds with Step 2. If the image is now recognized correctly, this feature search terminates successfully. After a given number of failed iterations, the search terminates without success.

Note that this feature search procedure requires that each new feature individually solves an existing recognition problem. This results in a strong tendency to learn few but informative features.

The value of k determines how accurately the conditional probabilities associated with the new feature are estimated in Step 6. A large value yields accurate estimates, but is expensive to evaluate. The danger of using too small a value of k is that many good features are discarded because of overly pessimistic probability estimates. Optimistic estimates result in the addition of features that later turn out to be of little value. As a result, these will cease to be used as they are superseded by more discriminative features. Thus, using a small k will result in more feature searches, each of which will take less time than when using a large k. Currently, the system uses $k = 5$.

4.2 Constructive Feature Generation

We now describe how a feature is generated in Step 2 above. There are six ways to generate a new feature that differ in the structural complexity of the resulting feature. In accord with *Occam's razor*, simple features are preferred over complex features. Also, they are more likely to be reusable and computed more rapidly than complex features. We implement a bias toward simple features by applying the following six methods in increasing order of structural complexity:

(a) Pick a random feature from some other Bayes net (corresponding to some other class) that is not yet part of this Bayes net (corresponding to the class of the sample image). This promotes the reuse of general features that are characteristic of more than one class.

(b) Sample a new feature directly from the sample image by either picking two points and turning them into a geometric compound of two edgels, or by picking one point and measuring a texel feature vector. (Individual edgels do not carry any information because all representations are invariant to rotation.)

(c) Pick a random existing feature, i.e. one that was learned earlier, or one that was generated by a previous, unsuccessful iteration of this feature search, find a location in the sample image where this feature occurs most strongly, and expand it geometrically by adding a randomly chosen edgel or texel nearby.

(d) Pick two random existing features and combine them topologically.

(e) Pick two random existing features and combine them into a conjunctive feature.

(f) Pick two random existing features and combine them into a disjunctive feature.

The first up to m executions of Step 2 in the procedure above apply the first method, the next m executions apply the second method, etc. This continues until either a suitable feature is found, or until the last method has been applied m times, at which point the current feature search terminates unsuccessfully. The parameter m in effect controls the bias toward simple features. A large value will spend more effort using each method, while a small value will give up sooner and move on to the next method. The current implementation uses $m = 10$.

4.3 Expert Feature Learning

As learning proceeds, the views sampled from the environment become increasingly representative, and the conditional probability tables in the Bayesian network classifiers increasingly reflect the true probabilities. Thus, learning should converge to a point where few misrecognitions occur, and hence few new features are learned. However, learning does not have to cease at this point. It is still possible to learn better features by increasing the minimum discriminative power required of features. When, after a correct recognition, there exist residual posterior probabil-

ities strictly greater than zero and less than one, a new feature is sought that will reduce or remove this uncertainty. This is accomplished by treating a correct recognition with non-zero entropy in a manner similar to a misrecognition.

Specifically, we call the class with the highest entropy (posterior probability closest to 0.5) the *uncertain class*. If this is the true class, we then find the maximum KSD of each feature of the uncertain class versus each of the false classes. The class that minimizes this maximum KSD is most poorly distinguishable from the uncertain class, based on any existing individual feature of the uncertain class. If this minimum KSD is below a threshold t, the same procedure (Section 4.1) is used to learn a new feature for the uncertain class that distinguishes it from this other class. Of the new feature, we require a KSD $\geq t$. Note that such a feature, if found, is guaranteed to be used by the recognition procedure. It will reduce the nonzero entropy that remained without this feature, as, by construction, it is present in the sample image. If the uncertain class is a false class, a new feature is learned that distinguishes it from the true class. The procedure is analogous. Again, the new feature is guaranteed to be used, and it will reduce the corresponding posterior probability.

Over time, the minimum KSD t required of a new feature can be raised, in search of ever more discriminative features. This procedure will terminate when all recognitions result in zero entropy. Unfortunately, this procedure does not guarantee anything about the discriminative power of surviving features from the basic learning stage. Theoretically, it is possible to achieve zero entropy using poor features with low KSDs. However, such features do not generalize well and tend to yield lower mutual information than better features. Therefore, in practice they are mostly eliminated during expert learning.

Alternatively, one could systematically search for features that discriminate between all pairs of classes and achieve a certain KSD. However, this is impractical for even moderate numbers of classes, and it would likely generate a large number of features that are never going to be used in practice. We prefer to allow actual experience to drive the formation of better features and to stop when there is no measurable need for better features.

As we will show in the following section, this procedure generates increasingly better features, obsoleting many features that were learned previously. Features that cease to be used can be deleted ("forgotten"). Moreover, the average number of features used per recognition is reduced, the number of correct recognitions on independent test images increased, and the number of wrong recognitions is reduced. In analogy to humans developing visual expertise with extensive practice (Gauthier & Tarr, 1997), we call this scheme *expert learning*.

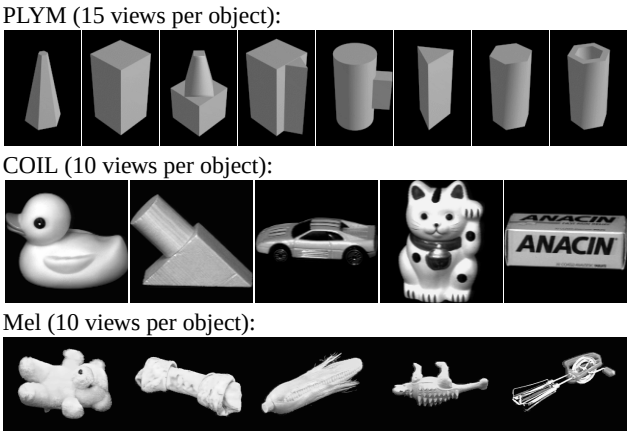

Figure 3. Example views of the three image databases.

5. Experiments

To avoid the difficulties associated with interactive robot learning environments as described previously, we performed our experiments in a conventional supervised-learning scenario using several object recognition data sets. Training images were presented to the agent one by one, cycling through the training set in random order. Evaluation images were chosen at random from the training set. Training was performed in stages. In the first stage, the basic learning strategy was applied as described in Section 4.1. The following stages performed expert learning (Section 4.3). At each stage, up to 20 iterations through the training set were performed, unless the training set was learned perfectly (zero entropy) or no progress was made at all during an entire iteration. The training set was reshuffled before each iteration. Up to 10 expert learning stages were performed, as long as new features were learned.

In the expert learning stages, the minimum required $\mathrm{KSD}_{\mathrm{target}}$ was updated according to the rule

$$\mathrm{KSD}_{\mathrm{target}} = 1 - \frac{1 - \min\{\mathrm{KSD}_{\mathrm{worst}}, \mathrm{KSD}_{\mathrm{target}}\}}{2},$$

where $\mathrm{KSD}_{\mathrm{worst}}$ is the worst KSD recorded at the previous stage. This rule seeks to increase $\mathrm{KSD}_{\mathrm{target}}$ exponentially, asymptoting at $\mathrm{KSD} = 1$, as long as the previous stage succeeded in achieving $\mathrm{KSD}_{\mathrm{worst}} \geq \mathrm{KSD}_{\mathrm{target}}$.

Experiments were performed using three data sets (Figure 3). The *PLYM* data set consisted of eight geometric objects on 15 artificially rendered images each, covering a small section of the viewing sphere[1]. We ran a two-fold cross-validation on the PLYM data based on two randomly-formed equally-sized subsets, each containing 7 or 8 images of each class. In the *COIL* task, 20 images of the first five objects from the COIL-20 database (Nene et al.,

[1] http://www.cis.plym.ac.uk/cis/levi/UoP_CIS_3D_Archive/8obj_set.tar

1996) were split into two disjoint subsets such that neighboring viewpoints were assigned to different subsets. The resulting image subsets each contained 10 images, spaced ten degrees apart on the viewing sphere, at constant elevation. Again, we performed a two-fold cross-validation on these two subsets. The *Mel* data set comprised 10 selected images of each of five classes from B. Mel's SEEMORE study (Mel, 1997). Here we performed a single run using 8 randomly chosen images of each class as the training set, and the remaining images as the test set.

Quantitative results are summarized in Figure 4. While the recognition results fall short of the best currently available machine recognition technology, they were achieved by an uncommitted visual system with a strong bias toward few and simple features that had access only to a small number of random training views at any given time during an incremental training procedure, not knowing that there was exactly one class present in each image. Most of these properties are contrary to current computer vision technology, but are characteristic of biological vision systems and would benefit many realistic computer vision applications. We are not aware of any comparable artificial systems.

In accord with our biased search strategy, most learned features were isolated texels and simple geometric compounds of edgels and/or texels. Smaller numbers of the other compound types of features were also found.

At the basic learning stage, there is no pressure on features to generalize. In principle, basic learning could generate features that memorize specific unique features in individual training images. As the minimum KSD required of a feature is increased during expert learning, all experiments show clear improvements: Correct recognitions increase, and most other recognition results tend to become less common. The number of features queried per recognition decreases, as does the number of features present in the system (sometimes after a transient increase). However, this behavior is not monotonic. For instance, COIL fold 2 at stage 2 exhibits a substantial drop in performance (see Figure 4). Notably, performance on the training set was also unusually poor at this stage (not shown in the figure). This observation, further supported by earlier pilot studies (Piater & Grupen, 2000), indicates that expert training can be assumed to improve performance on independent test data, unless performance on the training data is poor.

Notably, it is the learning agent itself that judges its own performance based on training-set performance and residual entropy. Our results demonstrate that this judgment is an excellent predictor of test-set performance and recognition speed, measured by the number of features queried. In particular, when training-set performance was perfect, i.e. no misrecognitions and no residual entropy, performance was always at a maximum in our experiments. This is precisely the stopping criterion for expert learning. In other words, an agent is well advised to spend its spare time doing expert learning, as this is very likely to pay off.

A dramatic case is made by the Mel task, which proved hard to learn for our system due to the small number of training images with relatively large viewpoint variations. Test-set performance after the basic learning stage was essentially at chance level, despite perfect results on the training set. The expert learning procedure achieved reasonable accuracy with no misclassifications, and a drastically reduced number of features.

6. Conclusions

We presented a framework for learning of features for visual recognition. The learning method contains few adjustable parameters, none of which is critical. Learning is incremental in two ways: First, training images are considered sequentially; second, continued "expert" learning produces increasingly better features and reduces any overfitting effects, resulting in improved test-set performance and a reduced number of features.

The objective of the system is to learn *discriminative* features. In the worst case, discrimination between all pairs of classes must explicitly be learned, which is only practical for a small number of classes. Since learned features are sampled directly from misrecognized images, they likely serve for more than their designated distinction. This reduces the number of features required in practice.

The sequential nature of the learning process is both a strength and a weakness. Learning any new image can cause the system to misclassify previously learned images, requiring extensive training until convergence. This is the price our system pays for being able to learn any distinctions expressible by an infinite feature space. This distinguishes it from almost all other visual recognition system based on local features. Future research will investigate how to combine the advantages of constructive feature learning with those of conventional methods; notably, feature-based indexing and the avoidance of unlearning. Progress in these areas will lead toward practical systems for learning large, open-ended visual tasks.

Acknowledgments

This work was supported in part by the National Science Foundation under grants CISE/CDA-9703217, IRI-9704530 and IRI-9503687, and by the Air Force Research Labs, IFTD (via DARPA) under grant F30602-97-2-0032. We also thank Bartlett Mel for providing image data, and Hugin Expert A/S for a low-cost PhD license of their Bayesian network library.

References

Amit, Y., Geman, D., & Wilder, K. (1997). Joint induction of shape features and tree classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 19*, 1300–1305.

Freeman, W. T., & Adelson, E. H. (1991). The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 13*, 891–906.

Gauthier, I., & Tarr, M. J. (1997). Becoming a "Greeble" expert: Exploring mechanisms for face recognition. *Vision Research, 37*, 1673–1682.

John, G. H., Kohavi, R., & Pfleger, K. (1994). Irrelevant features and the subset selction problem. *Proceedings of the Eleventh International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

McCallum, A. K. (1995). *Reinforcement learning with selective perception and hidden state*. Doctoral dissertation, Department of Computer Science, University of Rochester, Rochester, NY. Revised 1996.

Mel, B. W. (1997). Combining color, shape, and texture histogramming in a neurally-inspired approach to visual object recognition. *Neural Computation, 9*, 777–804.

Nene, S. A., Nayar, S. K., & Murase, H. (1996). *Columbia object image library (COIL-20)* (Technical Report CUCS-005-96). Columbia University, New York.

Pearl, J. (1988). *Probabilistic reasoning in intelligent systems: Networks of plausible inference*. San Francisco: Morgan Kaufmann.

Piater, J. H., & Grupen, R. A. (1999). Toward learning visual discrimination strategies. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition* (pp. 410–415).

Piater, J. H., & Grupen, R. A. (2000). Distinctive features should be learned. *Proceedings of the IEEE International Workshop on Biologically Motivated Computer Vision*. Springer-Verlag.

Precup, D., & Utgoff, P. E. (1998). Classification using Phi-machines and constructive function approximation. *Proceedings of the Fifteenth International Conference on Machine Learning*. San Francisco: Morgan Kaufmann.

Rao, R. P. N., & Ballard, D. H. (1995). An active vision architecture based on iconic representations. *Artificial Intelligence, 78*, 461–505.

Rimey, R. D., & Brown, C. M. (1992). Task-oriented vision with multiple Bayes nets. In A. Blake & A. Yuille (Eds.), *Active vision*, 217–236. Cambridge, Massachusetts: MIT Press.

Tanaka, J. W., & Taylor, M. (1991). Object categories and expertise: Is the basic level in the eye of the beholder? *Cognitive Psychology, 23*, 457–482.

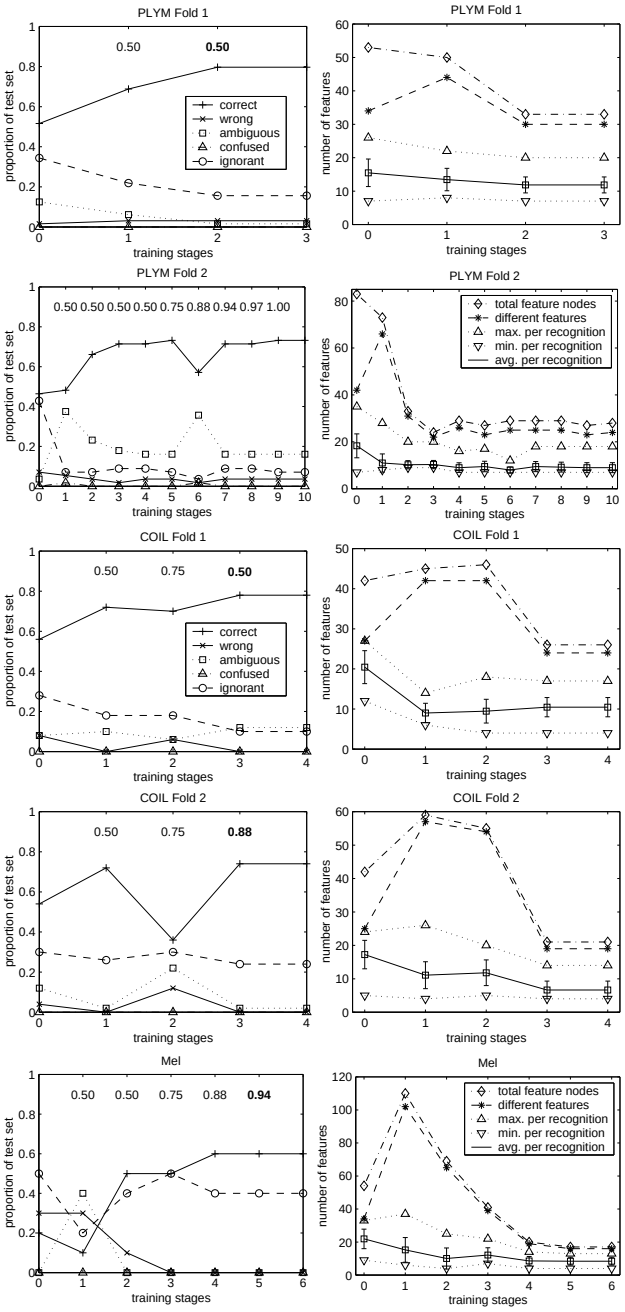

Figure 4. Experimental results. Stage 0 is the basic training stage, stages 1 and up perform iterative expert training. *Left column:* Recognition results. The numbers in the top of each graph give $\mathrm{KSD}_{\mathrm{target}}$, in bold type if there were no misclassifications on the training set. This was always the case for Stage 0. *Right column:* Feature statistics. The top two curves in each graph show the total number of feature nodes in all Bayes nets, and the number of different features in all Bayes nets, respectively. The difference is due to features that are shared between Bayes nets. The remaining curves show how many features are used in a single recognition run; error bars indicate one standard deviation.