Shape Quantization and Recognition with Randomized Trees

Yali Amit * and Donald Geman †

August 1996

*Department of Statistics, University of Chicago, Chicago, IL, 60637; Email: amit@galton.uchicago.edu.
Supported in part by the ARO under grant DAAL-03-92-G-0322.

†Department of Mathematics and Statistics, University of Massachusetts, Amherst, MA 01003; Email:geman@math.umass.edu. Supported in part by the NSF under grant DMS-9217655, ONR under contract N00014-91-J-1021, and ARPA contract MDA972-93-1-0012.

Abstract

We explore a new approach to shape recognition based on a virtually infinite family of binary features ("queries") of the image data, designed to accommodate prior information about shape invariance and regularity. Each query corresponds to a spatial arrangement of several local topographic codes ("tags") which are in themselves too primitive and common to be informative about shape. All the discriminating power derives from relative angles and distances among the tags. The important attributes of the queries are (i) a natural *partial ordering* corresponding to increasing structure and complexity; (ii) *semi-invariance*, meaning that most shapes of a given class will answer the same way to two queries which are successive in the ordering; and (iii) *stability*, since the queries are not based on distinguished points and substructures.

No classifier based on the full feature set can be evaluated and it is impossible to determine *a priori* which arrangements are informative. Our approach is to select informative features and build tree classifiers *at the same time* by inductive learning. In effect, each tree provides an approximation to the full posterior where the features chosen depend on the branch which is traversed. Due to the number and nature of the queries, standard decision tree construction based on a fixed length feature vector is not feasible. Instead we entertain only a small random sample of queries at each node, constrain their complexity to increase with tree depth, and grow multiple trees. The terminal nodes are labeled by estimates of the corresponding posterior distribution over shape classes. An image is classified by sending it down every tree and aggregating the resulting distributions.

The method is applied to classifying handwritten digits and synthetic linear and nonlinear deformations of three hundred Latex symbols. State-of-the-art error rates are achieved on the NIST database of digits. The principal goal of the experiments on Latex symbols is to analyze invariance, generalization error and related issues, and a comparison with ANN methods is presented in this context.

Contents

1 Introduction **5**

2 Invariant Recognition **8**

3 Shape Queries **13**

3.1 Tags 14

3.2 Tag Arrangements 16

4 The Posterior Distribution and Tree-Based Approximations **19**

5 Tree-Structured Shape Quantization **21**

5.1 Exploring Shape Space 21

5.2 Randomization 23

5.3 A Structural Description 24

5.4 Semi-Invariance 25

5.5 Unsupervised Learning 27

6 Multiple Trees **28**

6.1 Aggregation 29

6.2 Dependence on the Training Set 30

6.3 Relative Error Rates 31

6.4 Parameter Estimation 32

7 Performance Bounds **33**

7.1 Individual Trees: Twenty Questions 33

7.2 Multiple Trees: Weak Dependence 36

8 Generalization **38**

8.1 Interpolation 40

8.2 Extrapolation 41

8.3 Note 42

9 Incremental Learning and Universal Trees **43**

10 Fast Indexing **44**

11 Handwritten Digit Recognition **44**

12 Comparison with ANN's **47**

12.1 The Posterior Distribution and Generalization Error 47

12.2 Invariance and the Visual System 48

13 Conclusion **50**

1 Introduction

We explore a new approach to shape recognition based on the joint induction of shape features and tree classifiers. The data are binary images of two-dimensional shapes of varying sizes. The number of shape classes may reach into the hundreds (see Figure 1) and there may be considerable within-class variation, as with handwritten digits. The fundamental problem is how to design a practical classification algorithm which incorporates the prior knowledge that the shape classes remain invariant under certain transformations. The proposed framework is analyzed within the context of invariance, generalization error and other methods based on inductive learning, principally artificial neural networks (ANN).

Classification is based on a large, in fact virtually infinite, family of binary features of the image data which are constructed from local topographic codes ("tags"). A large sample of small subimages of fixed size is recursively partitioned based on individual pixel values. The tags are simply labels for the cells of each successive partition and each pixel in the image is assigned all the labels of the subimage centered there. As a result, the tags do *not* involve detecting distinguished points along curves, special topological structures, or any other complex attributes whose very definition can be problematic due to locally ambiguous data. In fact, the tags are too primitive and numerous to classify the shapes.

Although the mere existence of a tag conveys very little information, one can begin discriminating among shape classes by investigating just a few spatial relationships among the tags, for example asking whether there is a tag of one type "north" of a tag of another type. Relationships are specified by coarse constraints on the angles of the vectors connecting pairs of tags and on the relative distances among triples of tags. No absolute location or scale constraints are involved. An image may contain one or more instances of an arrangement, with significant variations in location, distances, angles, etc. There is one binary feature ("query") for each such spatial arrangement; the response is positive if a collection of tags consistent with the associated constraints is present anywhere in the image. Hence a query involves an extensive disjunction (ORing) operation.

Two images which answer the same to *every* query must have very similar shapes. In fact, it is reasonable to assume that the shape class is determined by the full feature set, i.e., the theoretical Bayes error rate is zero. But no classifier based on the full feature set can

be evaluated and it is impossible to determine *a priori* which arrangements are informative. Our approach is to select informative features and build tree classifiers (Breiman, Friedman, Olshen & Stone (1984), Casey & Nagy (1984), Quinlan (1986)) *at the same time* by inductive learning. In effect, each tree provides an approximation to the full posterior where the features chosen depend on the branch which is traversed.

There is a natural partial ordering on the queries which results from regarding each tag arrangement as a labeled graph: Vertex labels correspond to the tag types and edge labels to angle and distance constraints; see Figures 6,7. In this ways the features are ordered according to increasing structure and complexity. A related attribute is *semi-invariance*, which means that a large fraction of those images of a given class which answer the same way to a given query will also answer the same way to any query immediately succeeding it in the ordering. This leads to nearly invariant classification with respect to many of the transformations which preserve shape, such as scaling, translation, skew, and small non-linear deformations of the type shown in Figure 2.

Due to the partial ordering, tree construction with an infinite-dimensional feature set is computationally efficient. During training multiple trees (Breiman (1994), Dietterich & Bakiri (1995), Shlien (1990)) are grown and a form of randomization is used to reduce the statistical dependence from tree to tree; weak dependence is verified experimentally. Simple queries are used at the top of the trees and the complexity of the queries increases with tree depth. In this way semi-invariance is exploited and the space of shapes is systematically explored by calculating only a tiny fraction of the answers.

Each tree is regarded as a random variable on image space whose values are the terminal nodes. In order to *recognize* shapes, each terminal node of each tree is labeled by an estimate of the conditional distribution over the shape classes given that an image reaches that terminal node. The estimates are simply relative frequencies based on training data and require no optimization. A new data point is classified by dropping it down each of the trees, averaging over the resulting terminal distributions, and taking the mode of this aggregate distribution. Due to averaging and weak dependence, considerable errors in these estimates can be tolerated. Moreover, since tree-growing (i.e., question selection) and parameter estimation can be separated, the estimates can be refined indefinitely *without reconstructing the*

trees, simply by updating a counter in each tree for each new data point.

The separation between tree-making and parameter estimation, and the possibility of using different training samples for each phase, opens the way to selecting the queries based on either unlabeled samples (i.e., unsupervised learning) or based only on samples from some of the shape classes. Both of these perform surprisingly well compared with ordinary supervised learning.

Our recognition strategy differs from those based on true invariants (algebraic, differential, etc.) or "structural features" (holes, endings, etc.). These methods certainly introduce prior knowledge about shape and structure and we share that emphasis. However, invariant features usually require image normalization and/or boundary extraction, and are generally sensitive to shape distortion and image degradation. Similarly, structural features can be difficult to express as well-defined functions of the image (as opposed to model) data. In contrast, our queries are stable and primitive, precisely because they are *not* truly invariant and are *not* based on distinguished points or sub-structures.

A popular approach to multi-class learning problems in pattern recognition is based on ANNs, such as feedforward, multilayer perceptrons (Dietterich & Bakiri (1995), Fukushima & Miyake (1982), Knerr, Personnaz & Dreyfus (1992), Martin & Pitman (1991)). For example the best rates on handwritten digits are reported in LeCun, Boser, Denker, Henderson, Howard, Hubbard & Jackel (1990). Classification trees and neural networks certainly have aspects in common; for example, both rely on training data, are fast on-line, and require little storage; see Brown, Corruble & Pittard (1993), Gelfand & Delp (1991). However, our approach to invariance and generalization is, by comparison, more direct in that certain properties are acquired by "hardwiring" rather than depending on learning or image normalization. With ANNs, the emphasis is on parallel and local processing and a limited degree of disjunction, in large part due to assumptions regarding the operation of the visual system. However only a limited degree of invariance can be achieved with such models. In contrast, the features here involve extensive disjunction and more global processing, thus achieving a greater degree of invariance. This comparison is pursued in §12.

The paper is organized as follows. Other approaches to invariant shape recognition are reviewed in §2; synthesized random deformations of 293 basic Latex symbols (Figures 1,2)

provide a controlled experimental setting for an empirical analysis of invariance in a high dimensional shape space. The basic building blocks of the algorithm, namely the tags and the tag arrangements, are described in §3. In §4 we address the fundamental question of how to exploit the discriminating power of the feature set; we attempt to motivate the use of multiple decision trees in the context of the ideal Bayes classifier and the tradeoff between approximation error and estimation error. In §5 we explain the roles of the partial ordering and randomization for both supervised and unsupervised tree construction; we also discuss and quantify semi-invariance. Multiple decision trees and the full classification algorithm are presented in §6, together with an analysis of the dependence on the training set. In §7 we calculate some rough performance bounds, both for individual and multiple trees. Generalization experiments, where the training and test samples represent different populations, are presented in §8; and incremental learning is addressed in §9. "Fast indexing," another possible role for shape quantization, is considered in §10. We then apply the method in §11 to a real problem - classifying handwritten digits - using the NIST database for training and testing, achieving state-of-the-art error rates. In §12 we develop the comparison with ANNs in terms of invariance, generalization error, and connections to observed functions in the visual system. We conclude in §13 by assessing extensions to other visual recognition problems.

2 Invariant Recognition

Invariance is perhaps the fundamental issue in shape recognition, at least for isolated shapes. Some basic approaches are reviewed within the following framework. Let $\mathbf{X}$ denote a space of digital images and let $\mathcal{C}$ denote a set of shape classes. Let us assume that each image $\mathbf{x} \in \mathbf{X}$ has a true class label $Y(\mathbf{x}) \in \mathcal{C} = \{1, 2, ..., K\}$. Of course we cannot directly observe Y. In addition, there is a probability distribution P on $\mathbf{X}$. Our goal is to construct a classifier $\hat{Y} : \mathbf{X} \rightarrow \mathcal{C}$ such that $P(\hat{Y} \neq Y)$ is small.

In the literature on statistical pattern recognition it is common to address some variation by preprocessing or normalization. Given $\mathbf{x}$, and before estimating the shape class, one estimates a transformation ψ such that $\psi(\mathbf{x})$ represents a "standardized" image. Finding

ψ involves a sequence of procedures which brings all images to the same size and then corrects for translation, slant and rotation by one of a variety of methods. There may also be some morphological operations to standardize stroke thickness; Bottou, Cortes, Denker, Drucker, Guyon, Jackel, LeCun, Muller, Sackinger, Simard & Vapnik (1994),Hastie, Buja & Tibshirani (1995). The resulting image is then classified by one of the standard procedures (discriminant analysis, multilayer neural network, nearest-neighbors, etc.), in some cases essentially ignoring the global spatial properties of shape classes. Difficulties in generalization are often encountered because the normalization is not robust or does not accommodate nonlinear deformations. This deficiency can only be ameliorated with very large training sets; see the discussions in Hussain & Kabuka (1994), Raudys & Jain (1991), Simard, LeCun & Denker (1994), Werbos (1991) in the context of neural networks. Still, it is clear that robust normalization methods which reduce variability and yet preserve information can lead to improved performance of any classifier; we shall see an example of this in regard to "slant correction" for handwritten digits.

Template-matching is another approach. One estimates a transformation from $\mathbf{x}$ for *each* of the prototypes in the library. Classification is then based on the collection of estimated transformations. This requires explicit modeling of the prototypes, extensive computation at the estimation stage (usually involving relaxation methods) and appears impractical with large numbers of shape classes.

A third approach, closer in spirit to ours, is to search for invariant functions $\Phi(\mathbf{x})$, meaning that $P\left(\Phi(\mathbf{x}) = \phi_c | Y = c\right) = 1$ for some constants ϕ_c, $c = 1, ..., K$. The discriminating power of Φ depends on the extent to which the values ϕ_c are distinct. Many invariants for planar objects (based on single views) and for non-planar objects (based on multiple views) have been discovered and proposed for recognition; see Reiss (1993) and the references therein. Some invariants are based on Fourier descriptors and image moments; for example, the magnitude of Zernike moments (Khotanzad & Lu (1991)) are invariant to rotation. Most invariants require computing tangents from estimates of the shape boundaries (Forsyth, Mundy, Zisserman, Coelho, Heller & Rothwell (1991), Sabourin & Mitiche (1992)). Examples of such invariants include inflexions and discontinuities in curvature. In general the mathematical level of this work is advanced, borrowing ideas from projective, algebraic

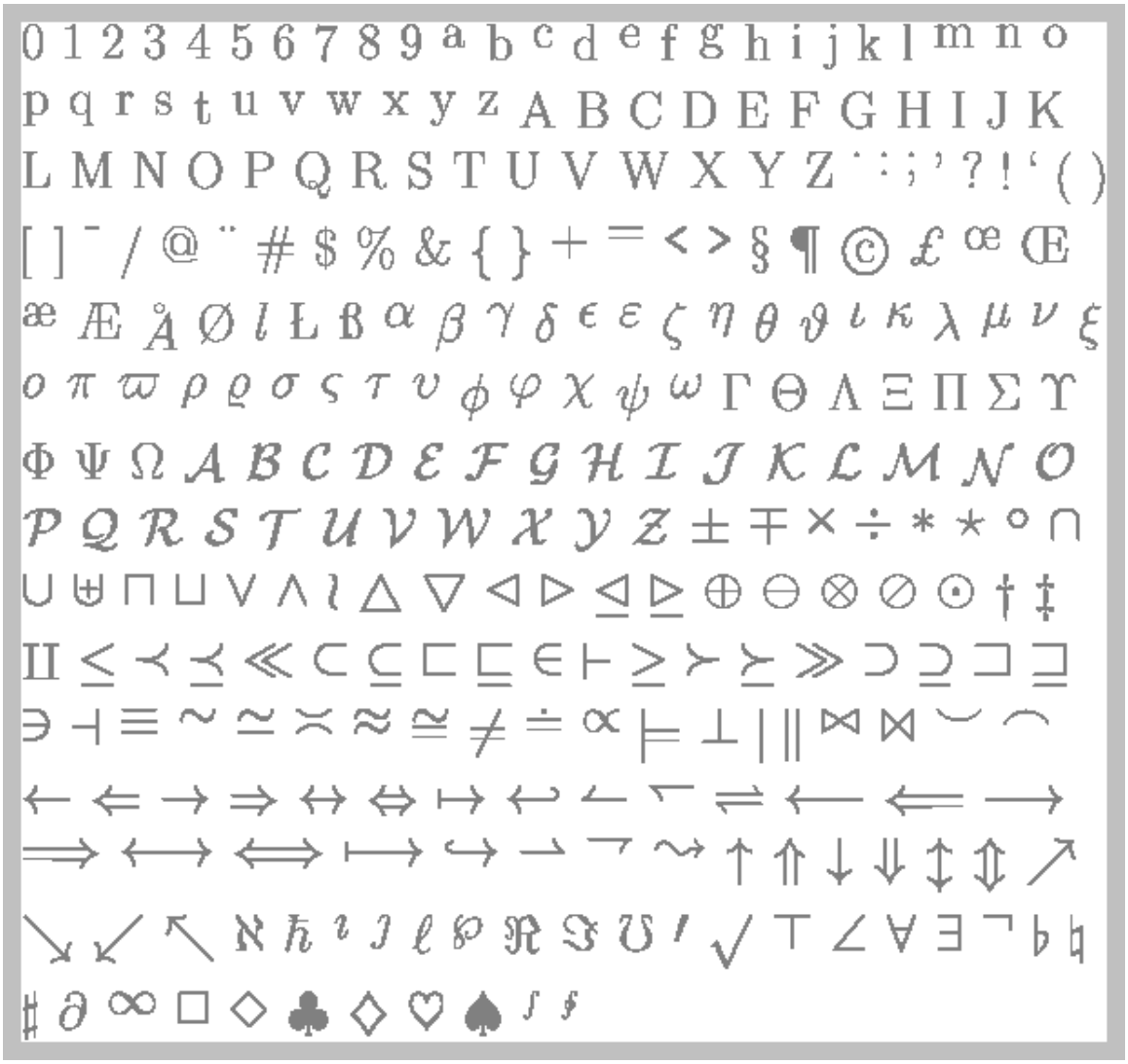

Figure 1: Latex symbols.

and differential geometry (Mundy & Zisserman (1992)).

Other successful treatments of invariance include geometric hashing Lamdan, Schwartz & Wolfson (1988) and nearest-neighbor classifiers based on affine invariant metrics (Simard et al. (1994)). Similarly, structural features involving topological shape attributes (such as junctions, endings, loops, etc.) or distinguished boundary points (such as points of high curvature) have some invariance properties, and many authors (e.g., Lee, Srihari & Gaborski (1991)) report much better results with such features than with standardized raw data.

In our view true invariant features of the form above might not be sufficiently stable for intensity-based recognition because the data structures are often too crude to analyze with continuum-based methods. In particular, such features are not invariant to non-linear

deformations and depend heavily on preprocessing steps such as normalization and boundary extraction. Unless the data is of very high quality, these steps may result in a lack of robustness to distortions of the shapes, due for example to digitization, noise, blur, and other degrading factors; see the discussion in Reiss (1993). Structural features are difficult to model and to extract from the data in a stable fashion. Indeed, it may be more difficult to recognize a "hole" than to recognize an "8". (Similar doubts about hand-crafted features and distinguished points are expressed in Jung & Nagy (1995).) In addition, if one could recognize the components of objects without recognizing the objects themselves, then the choice of classifier would likely be secondary.

Our features are not invariant. However, they are *semi-invariant* in an appropriate sense, and might be regarded as coarse substitutes for some of the true geometric, point-based invariants in the literature cited above. In this sense, we share at least the outlook expressed in recent, model-based work on "quasi-invariants" (Binford & Levitt (1993),Burns, Weiss & Riseman (1993)), where strict invariance is relaxed; however the functionals we compute are entirely different.

The invariance properties of the queries are related to the partial ordering and the manner in which they are selected during recursive partitioning. Roughly speaking, the complexity of the queries is proportional to the depth in the tree, i.e., to the number of questions asked. For elementary queries at the bottom of the ordering, we would expect that for each class c, either $P(Q = 1|Y = c) >> .5$ or $P(Q = 0|Y = c) >> .5$; however this collection of elementary queries would have low discriminatory power. (These statements will be amplified later on.) Queries higher up in the ordering have much higher discriminatory power, and maintain semi-invariance *relative* to subpopulations determined by the answers to queries preceding them in the ordering. Thus if $\tilde{Q}$ is a query immediately preceding Q in the ordering, then $P(Q = 1|\tilde{Q} = 1, Y = c) >> .5$ or $P(Q = 0|\tilde{Q} = 1, Y = c) >> .5$ for each class c. This will be defined more precisely in §5 and verified empirically.

Experiments on invariant recognition are scattered throughout the paper. Some involve real data - handwritten digits. Most employ synthetic data, in which case the data model involves a prototype $\mathbf{x}_c^*$ for each shape class $c \in \mathcal{C}$ (see Figure 1) together with a space Θ of image-to-image transformations. We assume that the class label of the prototype is

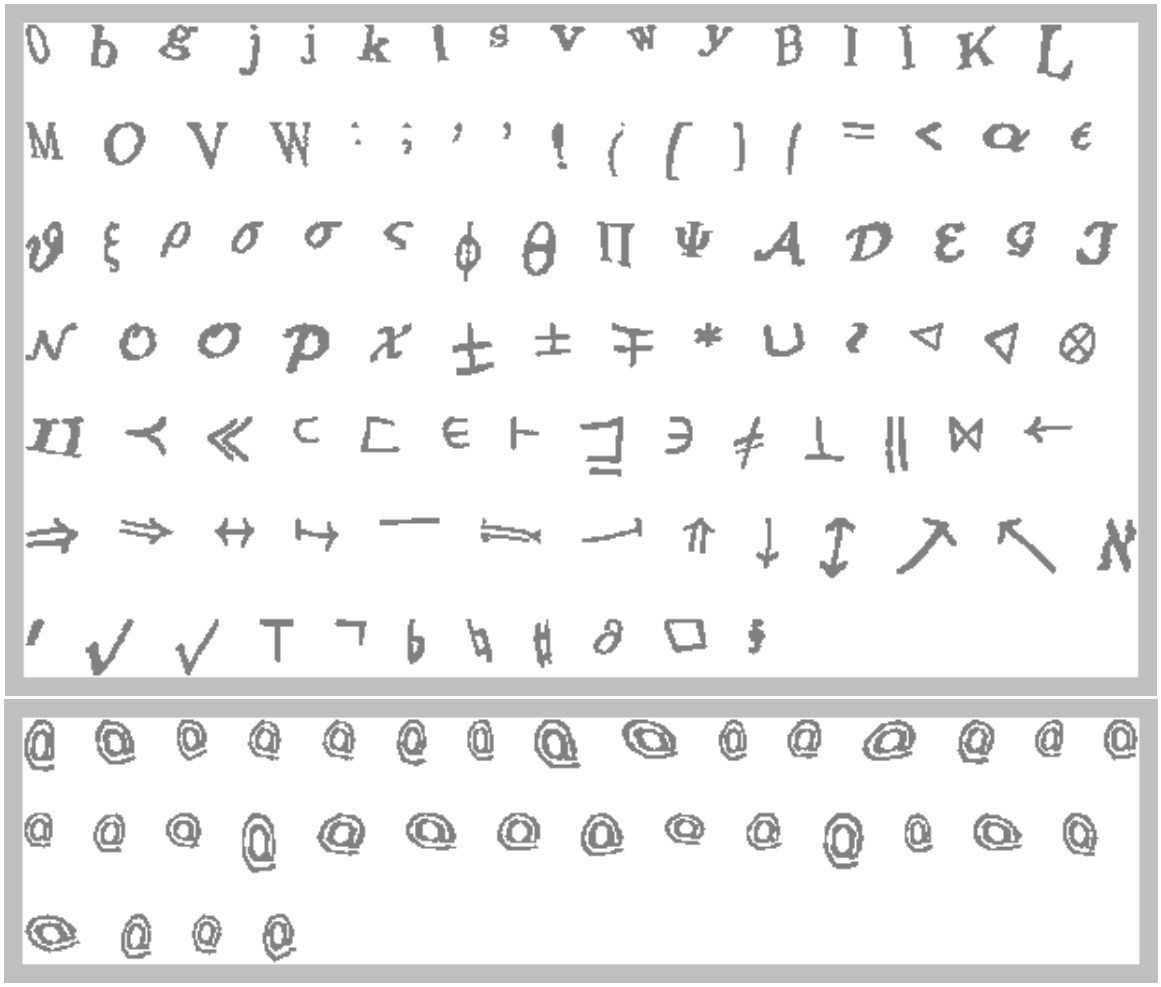

Figure 2: Top: Perturbed Latex symbols. Bottom: Training data for one symbol.

preserved under all transformation in Θ, namely $c = Y(\theta(\mathbf{x}_c^*))$ for all $\theta \in \Theta$, and that no two distinct prototypes can be transformed to the same image. We use "transformations" in a rather broad sense, referring to both affine maps, which alter the "pose" of the shapes, and to nonlinear maps, which "deform" the shapes. (We shall use "degradation" for noise, blur, etc.) Basically, Θ consists of *perturbations of the identity*. In particular, we are not considering the entire "pose space" but rather only perturbations of a reference pose, corresponding to the identity.

The probability measure P on $\mathbf{X}$ is derived from a probability measure $\nu(d\theta)$ on the space of transformations as follows: for any $D \subset \mathbf{X}$,

$$P(D) = \sum_c P(D|Y = c)\pi(c) = \sum_c \nu\{\theta : \theta(\mathbf{x}_c^*) \in D\}\pi(c)$$

where π is a prior distribution on $\mathcal{C}$, which we will always take to be uniform. Thus P

is concentrated on the space of images $\{\theta(\mathbf{x}_c^*)\}_{\theta,c}$. Needless to say, the situation is more complex in many actual visual recognition problems, for example in unrestricted 3D object recognition under standard projection models. Still, invariance is already challenging in the above context.

It is important to emphasize that this model is not used explicitly in the classification algorithm. Knowledge of the prototypes is not assumed, nor is θ estimated as in template approaches. The purpose of the model is to generate samples for training and testing.

The images in Figure 2 were made by random sampling from a particular distribution ν on a space Θ containing both linear (scale, rotation, skew) and nonlinear transformations. Specifically, the log-scale is drawn uniformly between $-1/6$ to $1/6$; the rotation angle is drawn uniformly from $+/-10$ degrees; and the log-ratio of the axes in the skew is drawn uniformly from $-1/3$ to $+1/3$. The nonlinear part is a smooth random deformation field constructed by creating independent random horizontal and vertical displacements, each of which is generated by random trigonometric series with only low frequency terms and with Gaussian coefficients. All images are 32×32 but the actual size of the object in the image varies significantly, both from symbol to symbol and within symbol classes due to random scaling.

3 Shape Queries

We first illustrate a shape query in the context of curves and tangents in an idealized, continuum setting. The example is purely motivational. In practice we are not dealing with one dimensional curves in the continuum but rather with a finite pixel lattice, strokes of variable width, corrupted data, etc. The types of queries we actually use are described in §3.1 and §3.2.

Observe the three versions of the digit "3" in Figure 3 (left) ; they are obtained by spline interpolation of the center points of the segments shown in Figure 3 (middle) in such a way that the segments represent the direction of the tangent at those points. All three segment arrangements satisfy the geometric relations indicated in Figure 3 (right): there is a vertical tangent northeast of a horizontal tangent, which in turn is south of another

Figure 3: Left: Three curves corresponding to the digit '3'. Middle: Three tangent configurations determining these shapes via spline interpolation. Right: Graphical description of relations between locations of derivatives consistent with all three configurations.

horizontal tangent, and so forth. Notice that the directional relations between the points are satisfied to within rather coarse tolerances. Not all curves of a "3" contain five points whose tangents satisfy all these relations. Put differently, some "3"s answer "no" to the query "*Is there a vertical tangent northeast of a ...?*". However rather substantial transformations of each of the versions below will answer "yes." Moreover among those "3"s which answer "no", it is possible to choose a small number of alternative arrangements in such a way that the entire space of "3"s is covered.

3.1 Tags

We employ primitive local features called tags which provide a coarse description of the local topography of the intensity surface in the neighborhood of a pixel. Instead of trying to manually characterize local configurations of interest, for example trying to define local operators to identify gradients in the various directions, we adopt an information-theoretic approach and "code" a microworld of subimages by a process very similar to tree-structured vector quantization. In this way we sidestep the issues of boundary detection and gradients in the discrete world, and allow for other forms of local topographies. This approach has been extended to grey level images in Jedynak & Fleuret (1996).

The basic idea is to re-assign symbolic values to each pixel based on examining a few pixels in its immediate vicinity; the symbolic values are the tag types and represent labels

for the local topography. The neighborhood we choose is the 4×4 subimage containing the pixel at the upper left corner. We cluster the subimages with binary splits corresponding to adaptively choosing the five most informative locations of the sixteen sites of the subimage.

Note that the size of the subimages used must depend on the resolution at which the shapes are imaged. The 4×4 subimages are appropriate for a certain range of resolutions, roughly 10×10 through 70×70 in our experience. The size must be adjusted for higher resolution data and the ultimate performance of the classifier will suffer if the resolution of the test data is not approximately the same as that of the training data. The best approach would be one that is multi-resolution, something we have not done in this paper (except for some preliminary experiments in §11), but which is carried out in Jedynak & Fleuret (1996) in the context of grey-level images and 3D objects.

A large sample of 4×4 subimages are randomly extracted from the training data. The corresponding shape classes are irrelevant and are not retained. The reason is that the purpose of the sample is to provide a representative database of micro-images and to discover the "biases" at that scale; the statistics of that world is largely independent of global image attributes, such as symbolic labels. This family of subimages is then recursively partitioned with binary splits. There are $4 \times 4 = 16$ possible "questions": *"Is site* (i, j) *black?"* for $i, j = 1, 2, 3, 4$. The criterion for choosing a question at a node t is dividing the subimages $\mathbf{U}_t$ at the node as equally as possible into two groups. This corresponds to reducing as much as possible the entropy of the empirical distribution on the 2^{16} possible binary configurations for the sample $\mathbf{U}_t$. There is a tag type for each node of the resulting tree, except for the root. Thus, if three questions are asked there are $2 + 4 + 8 = 14$ tags and if five questions are asked there are 62 tags. Depth five tags correspond to a more detailed description of the local topography than depth three tags, although eleven of the sixteen pixels still remain unexamined. Observe also that tags corresponding to internal nodes of the tree represent unions of those associated with deeper ones. At each pixel we assign all the tags encountered by the corresponding 4×4 subimage as it proceeds down the tree. Unless otherwise stated all experiments below use 62 tags.

At the first level every site splits the population with nearly the same frequencies. However, at the second level some sites are more informative than others, and by level four and

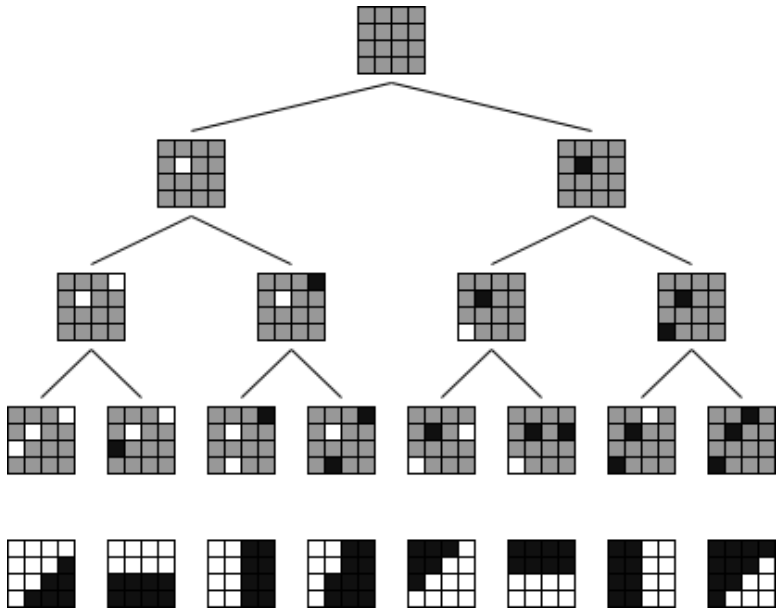

Figure 4: First three tag levels with most common configurations.

five there is usually one site which partitions the remaining subpopulation much better than all others. In this way, the world of micro-images is efficiently coded. For efficiency, the population is restricted to subimages containing at least one black and one white site within the center four, which then obviously concentrates the processing in the neighborhood of boundaries. In the grey-level context it is also useful to consider more general tags, allowing for example for variations on the concept of local homogeneity.

The first three levels of the tree are shown in Figure 4 together with the most common configuration found at each of the eight level three nodes. Notice that the "level one" tag alone (i.e., the first bit in the code) determines the original image, so this "transform" is invertible and redundant. In Figure 5 we show all the 2 bit tags and 3 bit tags appearing in an image.

3.2 Tag Arrangements

The queries involve geometric arrangements of the tags. A query $Q_{\mathbf{A}}$ asks whether a specific geometric arrangement $\mathbf{A}$ of tags of certain types is present ($Q_{\mathbf{A}}(\mathbf{x}) = 1$) or is not present ($Q_{\mathbf{A}}(\mathbf{x}) = 0$) in the image. Figure 6 shows several Latex symbols which contain a specific

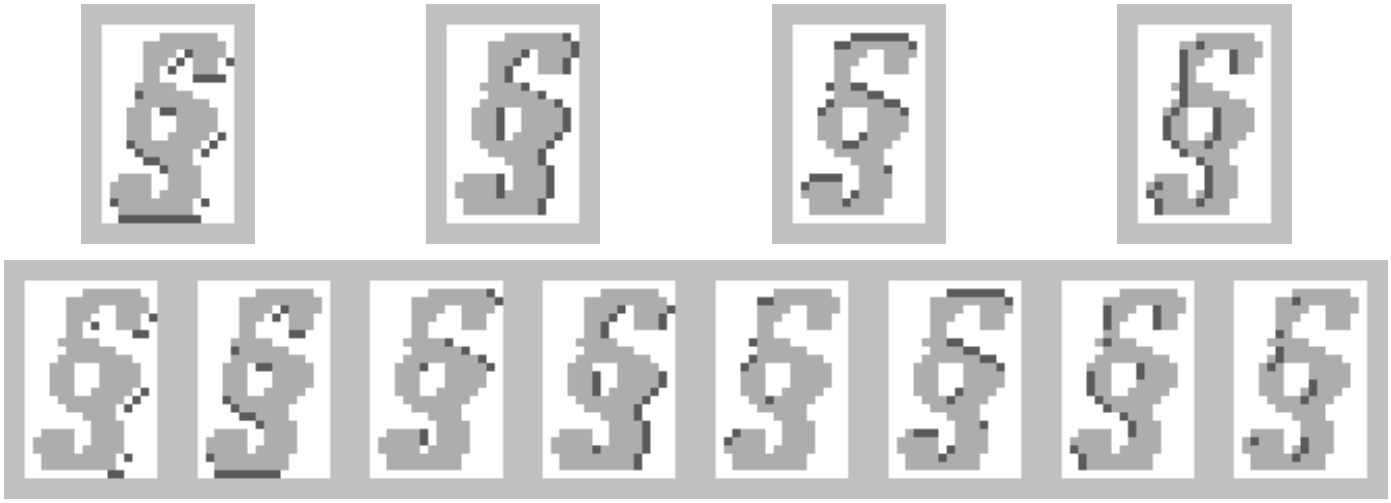

Figure 5: Top: All instances of the four 2 bit tags. Bottom: All instances of the eight 3 bit tags.

geometric arrangement of tags: *tag 16 northeast of tag 53 which is northwest of tag 19.* Notice that there are no fixed locations in this description whereas the tags in any specific image do carry locations. "Present in the image" means there is at least one set of tags in $\mathbf{x}$ of the prescribed types whose locations satisfy the indicated relationships. In Figure 6, notice for example how different instances of the digit 0 still contain the arrangement. Tag 16 is a depth four tag; the corresponding four questions in the subimage are indicated by the following mask: $\begin{pmatrix} n & n & n & 1 \\ 0 & n & n & n \\ n & 0 & 0 & n \\ n & n & n & n \end{pmatrix}$ where 0 corresponds to background, 1 to object, and n to "not asked." These neighborhoods are loosely described by "background to lower left, object to upper right." Similar interpretations can be made for tags 53 and 19.

Restricted to the first ten symbol classes (the ten digits), the conditional distribution $P(Y = c|Q_{\mathbf{A}} = 1)$ on classes given the existence of this arrangement in the image is given in Table 1. Already this simple query contains significant information about shape.

To complete the construction of the feature set we need to define a set of allowable *relationships* among image locations. These are binary functions of pairs, triples, etc. of planar points which depend only on their relative coordinates. An arrangement $\mathbf{A}$ is then a labeled (hyper)graph. Each vertex is labeled with a type of tag and each edge (or "superedge") is labeled with a type of relation. The graph in Figure 6, for example, has only binary rela-

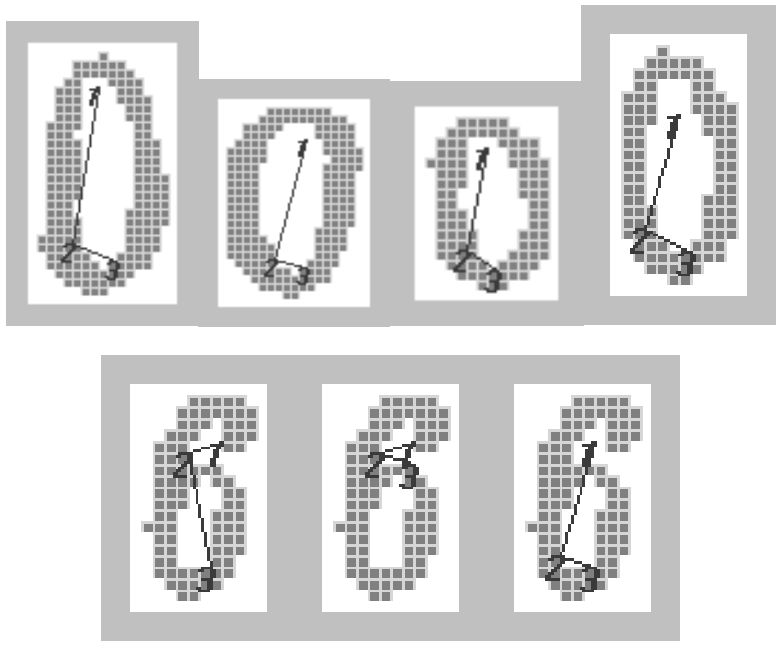

Figure 6: Top: Instances of a geometric arrangement in several 0's. Bottom: Several instances of the geometric arrangement in one 6

0	1	2	3	4	5	6	7	8	9
.13	.003	.03	.08	.04	.07	.23	0	.26	.16

Table 1: Conditional distribution on digit classes given the arrangement of Figure 6.

tions. In fact, all the experiments on the Latex symbols are restricted to this setting. The experiments on handwritten digits also use a ternary relationship of the metric type - see below.

There are eight binary relations between any two locations u and v corresponding to the eight compass headings "north," "northeast," "east," etc. For example, u is "north" of v if the angle of the vector $u - v$ is between $\pi/4$ and $3\pi/4$. More generally, the two points satisfy relation k ($k = 1, ..., 8$) if the angle of the vector $u - v$ is within $\pi/4$ of $k * \pi/4$. Let $\mathcal{A}$ denote the set of all possible arrangements and let $\mathbf{Q} = \{Q_{\mathbf{A}} : \mathbf{A} \in \mathcal{A}\}$, our feature set.

There are many other binary and ternary relations that have discriminating power. For example, there is an entire family of "metric" relationships that are, like the "directional" relationships above, completely scale and translation invariant. Given points u, v, w, z, one example of a ternary relation is $\| u - v \| < \| u - w \|$, which inquires whether or not u closer is to v than to w. With four points we might ask if $\| u - v \| < \| w - z \|$.

4 The Posterior Distribution and Tree-Based Approximations

For simplicity, and in order to facilitate comparisons with other methods, we restrict ourselves to queries $Q_{\mathbf{A}}$ of bounded complexity. For example, consider arrangements $\mathbf{A}$ with at most twenty tags and twenty relations; this limit is never exceeded in any of the experiments. Enumerating these arrangements in some fashion, let $\mathbf{Q} = (Q_1, ..., Q_M)$ be the corresponding feature vector assuming values in $\{0, 1\}^M$. Each image $\mathbf{x}$ then generates a bit string of length M which contains all the information which is available for estimating $Y(\mathbf{x})$. Of course M is enormous. Nonetheless, it is not evident how we might determine *a priori* which features are informative and thereby reduce M to manageable size.

Evidently these bit strings partition $\mathbf{X}$. Two images which generate the same bit string or "atom" need not be identical. Indeed, due to the invariance properties of the queries, the two corresponding symbols may vary considerably in scale, location and skew, and are not even affine-equivalent in general. Nonetheless, two such images will have *very* similar shapes. As a result, it is reasonable to expect that $H(Y|\mathbf{Q})$ is very small, in which case we can in principle obtain high classification rates using $\mathbf{Q}$.

To simplify things further, at least conceptually, we will assume that $H(Y|\mathbf{Q}) = 0$; as indicated above, this is not an unreasonable assumption for large M. An equivalent assumption is that the shape class Y is determined by $\mathbf{Q}$ and the error rate of the Bayes classifier

$$\hat{Y}_B = \arg\max_c P(Y = c|\mathbf{Q})$$

is zero. Needless to say, perfect classification cannot actually be realized: Due to the size of M, the full posterior cannot be computed and the classifier $\hat{Y}_B$ is only hypothetical.

Suppose we examine *some* of the features by constructing a single binary tree T based on entropy-driven recursive partitioning and randomization and that T is uniformly of depth D so that D of the M features are examined for each image $\mathbf{x}$. The exact procedure is described in the following section and the details are not important for the moment. Suffice it to say that a feature Q_m is assigned to each interior node of T and the set of features $Q_{\pi_1}, ..., Q_{\pi_D}$ along each branch from root to leaf is chosen sequentially and based on the

current information content given the observed values of the previously chosen features. The classifier based on T is then

$$\begin{aligned}\hat{Y}_T &= \arg\max_c P(Y=c|T) \\ &= \arg\max_c P(Y=c|Q_{\pi_1},...,Q_{\pi_D}).\end{aligned}$$

Since $D \ll M$, $\hat{Y}_T$ is not the Bayes classifier. However, even for values of D on the order of hundreds or thousands we can expect that

$$P(Y=c|T) \approx P(Y=c|\mathbf{Q})$$

We shall refer to the difference between these distributions (in some appropriate norm) as the "approximation error" ("AE"). This is one of the sources of error in replacing $\mathbf{Q}$ by a subset of features. Of course we cannot actually compute a tree of such depth since at least several hundred features are needed to achieve good classification; we shall return to this point shortly.

Regardless of the depth D, in reality we do not actually know the posterior distribution $P(Y=c|T)$. Rather it must be estimated from a training set $\mathcal{L} = \{(\mathbf{x}_1, Y(\mathbf{x}_1)), ..., (\mathbf{x}_m, Y(\mathbf{x}_m))\}$, where $\mathbf{x}_1, ..., \mathbf{x}_m$ is a random sample from P. (The training set is also used to estimate the entropy values during recursive partitioning.) Let $\hat{P}_{\mathcal{L}}(Y=c|T)$ denote the *estimated* distribution, obtained by simply counting the number of training images of each class c which land at each terminal node of T. If $\mathcal{L}$ is sufficiently large then

$$\hat{P}_{\mathcal{L}}(Y=c|T) \approx P(Y=c|T)$$

We call the difference "estimation error" ("AE"), which of course only vanishes as $|\mathcal{L}| \to \infty$.

The purpose of multiple trees (§6) is to solve the approximation error problem and the estimation error problem at the same time. Even if we could compute and store a very deep tree there would still be too many probabilities (specifically $K2^D$) to estimate with a practical training set $\mathcal{L}$. Our approach is to build multiple trees $T_1, ..., T_N$ of modest depth. In this way tree construction is practical and

$$\hat{P}_{\mathcal{L}}(Y=c|T_n) \approx P(Y=c|T_n), \;\; n=1,...,N.$$

Moreover, the total number of features examined is sufficiently large to control the approximation error. The classifier we propose is

$$\hat{Y}_S = \arg\max_c \frac{1}{N} \sum_{n=1}^{N} \hat{P}_{\mathcal{L}}(Y = c|T_n).$$

An explanation for this particular way of aggregating the information from multiple trees is provided in §6.1. In principle, a better way to combine the trees would be to classify based on the mode of $P(Y = c|T_1, ..., T_N)$. However this is impractical for reasonably-sized training sets for the same reasons that a single deep tree is impractical; see §6.4 for some numerical experiments. The tradeoff between "AE" and "EE" is related to the tradeoff between "bias" and "variance," which is discussed in §6.2 and the relative error rates among all these classifiers is analyzed in more detail in §6.4 in the context of parameter estimation.

5 Tree-Structured Shape Quantization

Standard decision tree construction (Breiman et al. (1984), Quinlan (1986)) is based on a scalar-valued "feature" or "attribute" vector $\mathbf{z} = (z_1, ..., z_k)$ where k is generally about $10 - 100$. Of course in pattern recognition the raw data are images and finding the "right" attributes is widely regarded as the main issue. Standard splitting rules are based on functions of this vector, usually involving a single component z_j (e.g., applying a threshold) but occasionally involving multivariate functions or "trans-generated features" (Friedman (1973), Gelfand & Delp (1991), Guo & Gelfand (1992), Sethi (1991)). In our case, the queries $\{Q_{\mathbf{A}}\}$ are the candidates for splitting rules. We now describe the manner in which the queries are used to construct a tree.

5.1 Exploring Shape Space

Since the set of queries $\mathbf{Q}$ is indexed by graphs there is a natural partial ordering under which a graph precedes any of its extensions. The partial ordering corresponds to a hierarchy of structure. Small arrangements with few tags produce coarse splits of shape space. As the arrangements increase in size (say the number of tags plus relations), they contain more and more information about the images which contain them. However, fewer and fewer

images contain such an instance; that is, $P(Q = 1) \approx 0$ for a query Q based on a complex arrangement.

One straightforward way to exploit this hierarchy is to build a decision tree using the collection $\mathbf{Q}$ as candidates for splitting rules, with the complexity of the queries increasing with tree depth (distance from the root). In order to begin to make this computationally feasible, we define a *minimal extension* of an arrangement $\mathbf{A}$ to mean the addition of exactly one relation between existing tags, or the addition of exactly one tag and one relation binding the new tag to an existing one. By a *binary arrangement* we mean one with two tags and one relation; the collection of associated queries is denoted $\mathbf{B} \subset \mathbf{Q}$.

Now build a tree as follows. At the root search through $\mathbf{B}$ and choose the query $Q \in \mathbf{B}$ which leads to the greatest reduction in the mean uncertainty about Y given Q. This is the standard criterion for recursive partitioning in machine learning and other fields. Denote the chosen query $Q_{\mathbf{A}_0}$. Those data points for which $Q_{\mathbf{A}_0} = 0$ are in the "no" child node and we search again through $\mathbf{B}$. Those data points for which $Q_{\mathbf{A}_0} = 1$ are in the "yes" child node and have one or more instances of $\mathbf{A}_0$, the "pending arrangement." Now search among minimal extensions of $\mathbf{A}_0$ and choose the one which leads to the greatest reduction in uncertainty about Y *given the existence of* $\mathbf{A}_0$. The digits in Figure 6 were taken from a depth 2 ("yes/yes") node of such a tree.

We measure uncertainty by Shannon entropy. The expected uncertainty in Y given a random variable Z is

$$H(Y|Z) = -\sum_z P(Z = z) \sum_c P(Y = c|Z = z) \log_2 P(Y = c|Z = z).$$

Define $H(Y|Z, B)$ for an event $B \subset \mathbf{X}$ in the same way, except that P is replaced by the conditional probability measure $P(.|B)$.

Given we are at a node t of depth $k > 0$ in the tree, let the "history" be $B_t = \{Q_{\mathbf{A}_0} = q_0, ..., Q_{\mathbf{A}_{k-1}} = q_{k-1}\}$, meaning that $Q_{\mathbf{A}_1}$ is the second query chosen given that $q_0 \in \{0, 1\}$ is the answer to the first, $Q_{\mathbf{A}_2}$ is the third query chosen given the answers to the first two are q_0 and q_1, and so forth. The pending arrangement, say $\mathbf{A}_j$, is the deepest arrangement along the path from root to t for which $q_j = 1$, so that $q_i = 0, i = j + 1, ..., k - 1$. Then $Q_{\mathbf{A}_k}$ minimizes $H(Y|Q_{\mathbf{A}}, B_t)$ among minimal extensions of $\mathbf{A}_j$. Continue in this fashion until a

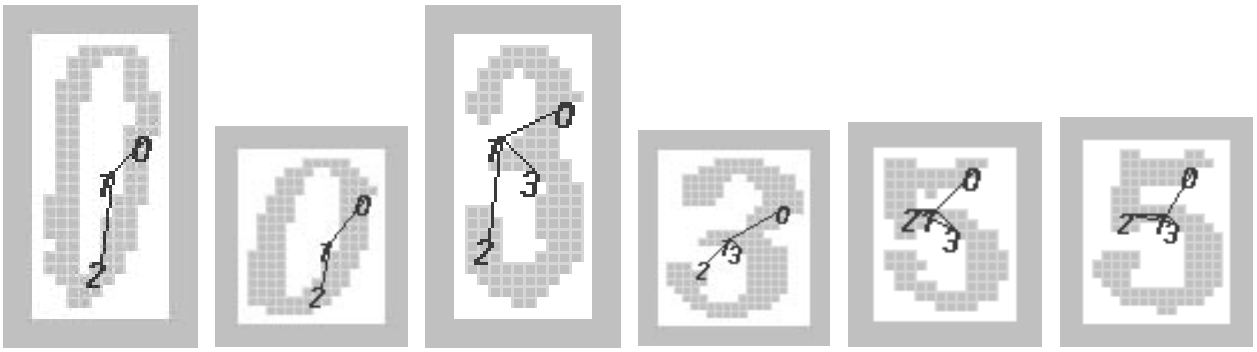

Figure 7: Examples of node spitting. All six images lie in the same node and have a pending arrangement with three vertices. The 0's are separated from the 3's and 5's by asking for the presence of a new tag and then the 3's and 5's are separated by asking a question about the relative angle between two existing vertices. The particular tags associated with these vertices are not indicated.

stopping criterion is satisfied, e.g., the number of data points at every terminal node falls below a threshold. Each tree may then be regarded as a discrete random variable T on $\mathbf{X}$; each terminal node corresponds to a different value of T.

In practice, we cannot compute these expected entropies; we can only estimate them from a training set $\mathcal{L}$. Then P is replaced by the empirical distribution $\hat{P}_{\mathcal{L}}$ on $\{\mathbf{x}_1, ..., \mathbf{x}_m\}$ in computing the entropy values.

5.2 Randomization

Despite the growth restrictions, the procedure above is still not practical; the number of binary arrangements is very large and there are too many minimal extensions of more complex arrangements. In addition, if more than one tree is made, even with a fresh sample of data points per tree, there might be very little difference among the trees. The solution is simple: Instead of searching among *all* the admissible queries at each node, we restrict the search to a *small random subset*.

5.3 A Structural Description

Notice that only *connected* arrangements can be selected, meaning either every two tags are "neighbors" (participate in a relation) or are connected by a sequence of neighboring tags. As a result, training is more complex than standard recursive partitioning. At each node, a list must be assigned to each data point consisting of all instances of the pending arrangement, including the coordinates of each participating tag. If a data point passes to the "yes" child, then only those instances which can be incremented are maintained and updated; the rest are deleted. The more data points the more bookkeeping.

Another, far simpler, possibility is sampling exclusively from $\mathbf{B}$, the binary arrangements (i.e., two vertices and one relation) listed in some order. In fact, we can imagine evaluating *all* the queries in $\mathbf{B}$ for each data point. This vector could then be used with a variety of standard classifiers, including decision trees built in the standard fashion. In the latter case the pending arrangements are unions of binary graphs, each one disconnected from all the others. This approach is much simpler and faster to implement and preserves the semi-invariance.

However the price is dear: we lose the common, global characterization of shape in terms of a large connected graph. Here we are referring to the pending arrangements at the terminal nodes (except at the end of the all "no" branch); by definition, this graph is found in all the shapes at the node. This is what we mean by a *structural description.* The difference between one connected graph and a union of binary graphs can be illustrated as follows. Relative to the *entire* population $\mathbf{X}$, a random selection in $\mathbf{B}$ is quite likely to carry some information about Y, measured say by the mutual information $I(Y,Q) = H(Y) - H(Y|Q)$. On the other hand, a random choice among all queries with, say, five tags will most likely have no information because nearly all data points $\mathbf{x}$ will answer "no." In other words, it makes sense to at least *start* with binary arrangements.

Assume, however, we are restricted to a subset $\{Q_{\mathbf{A}} = 1\} \subset \mathbf{X}$ determined by an arrangement $\mathbf{A}$ of moderate complexity. (In general, the subsets at the nodes are determined the "no" answers as well, but the situation is virtually the same.) *On this small subset a randomly sampled binary arrangement will be less likely to yield a significant drop in uncertainty than a randomly sampled query among minimal extensions of* $\mathbf{A}$. These observations

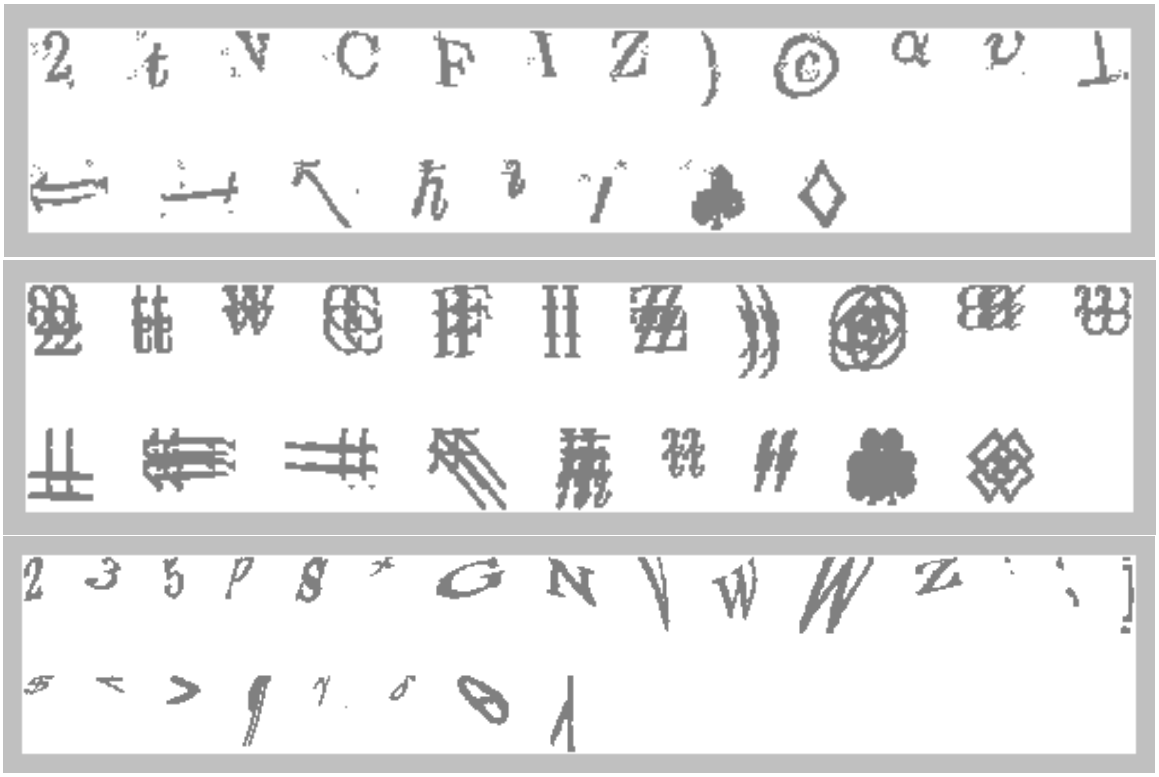

Figure 8: Samples from data sets. Top: Spot noise. Middle: Duplication. Bottom: Severe perturbations.

have been verified experimentally and we omit the details.

This distinction becomes more pronounced if the images are noisy (top panel of Figure 8) or contain structured backgrounds (bottom panel of Figure 11) because there will be many "false positives" for arrangements with only two tags. However, the chance of finding complex arrangements utilizing "noise tags" or "background tags" is much smaller. Put differently a structural description is more robust than a list of attributes. The situation is the same for more complex shapes; see for example the middle panel of Figure 8 where the shapes were created by duplicating each symbol four times with some shifts. Again, a random choice among minimal extensions carries much more information than a random choice in $\mathbf{B}$.

5.4 Semi-Invariance

Another benefit of the structural description is what we refer to as semi-invariance. Given a node t, let B_t be the history and let $\mathbf{A_j}$ be the pending arrangement. For any minimal

extension $\mathbf{A}$ of $\mathbf{A_j}$, and for any shape class c, we want

$$\max\left(P(Q_{\mathbf{A}} = 0|Y = c, B_t),\ P(Q_{\mathbf{A}} = 1|Y = c, B_t)\right) >> .5.$$

In other words, most of the images in B_t of the same class should answer the same way to query $Q_{\mathbf{A}}$. In terms of entropy, semi-invariance is equivalent to relatively small values of $H(Q_{\mathbf{A}}|Y = c, B_t)$ for all c. Averaging over classes, this in turn is equivalent to small values of $H(Q_{\mathbf{A}}|Y, B_t)$ at each node t.

In order to verify this property we created ten trees of depth five using the data set described in §2 with 32 samples per symbol class. At each non-terminal node t of each tree, the average value of $H(Q_{\mathbf{A}}|Y, B_t)$ was calculated over 20 randomly sampled minimal extensions. Over all nodes, the mean entropy was $m = .33$; this is the entropy of the distribution $(.06, .94)$. The standard deviation over all nodes and queries was $\sigma = .08$. Moreover there was a clear decrease in average entropy (i.e., increase in the degree of invariance) as the depth of the node increases.

We also estimated the entropy for more severe deformations. On a more variable data set with approximately double the range of rotations, log-scale, and log-skew (relative to the values in §2), and the same non-linear deformations, the corresponding numbers were $m = .38, \sigma = .09$. Finally for rotations sampled from $(-30, 30)$ degrees, log-scale from $(-.5, .5)$, log-skew from $(-1, 1)$, and doubling the variance of the random non-linear deformation (see bottom panel of Figure 8), the corresponding mean entropy was $m = .44$ ($\sigma = .11$), corresponding to a $(.1, .9)$ split. In other words, on average, ninety percent of the images in the same shape class still answer the same way to a new query.

Notice that invariance property is independent of the *discriminating power* of the query, i.e., the extent to which the distribution $P(Y = c|B_t, Q_{\mathbf{A}})$ is more peaked than the distribution $P(Y = c|B_t)$. Due to the symmetry of mutual information,

$$H(Y|B_t) - H(Y|Q_{\mathbf{A}}, B_t) = H(Q_{\mathbf{A}}|B_t) - H(Q_{\mathbf{A}}|Y, B_t).$$

This means that if we seek a question which maximizes the reduction in the conditional entropy of Y, and if we assume the second term on the right is small due to semi-invariance, then we need only find a query which maximizes $H(Q_{\mathbf{A}}|B_t)$. This however does not in-

volve the class variable and hence points to the possibility of unsupervised learning which is discussed in the following section.

5.5 Unsupervised Learning

We outline two ways to construct trees in an *unsupervised* mode, i.e., without using the class labels $Y(\mathbf{x}_j)$ of the samples $\mathbf{x}_j$ in $\mathcal{L}$. Clearly each query Q_m decreases uncertainty about $\mathbf{Q}$, and hence about Y. Indeed, $H(Y|Q_m) \leq H(\mathbf{Q}|Q_m)$ since we are assuming Y is determined by $\mathbf{Q}$. More generally, if T is a tree based on some of the components of $\mathbf{Q}$, and if $H(\mathbf{Q}|T) << H(\mathbf{Q})$, then T should contain considerable information about the shape class. Recall that in the supervised mode the query Q_m chosen at node t minimizes $H(Y|B_t, Q_m)$ (among a random sample of admissible queries), where B_t is the event in $\mathbf{X}$ corresponding to the answers to the previous queries. Notice that typically this is *not* equivalent to simply maximizing the information content of Q_m because $H(Y|B_t, Q_m) = H(Y, Q_m|B_t) - H(Q_m|B_t)$ and *both* terms depend on m. However, in light of the discussion in the preceding section about semi-invariance, the first term can be ignored and we can focus on maximizing the second term. Another way to motivate this criterion is to replace Y by $\mathbf{Q}$, in which case

$$\begin{aligned} H(\mathbf{Q}|B_t, Q_m) &= H(\mathbf{Q}, Q_m|B_t) - H(Q_m|B_t) \\ &= H(\mathbf{Q}|B_t) - H(Q_m|B_t). \end{aligned}$$

Since the first term is independent of m, the query of choice will again be the one maximizing $H(Q_m|B_t)$. Recall that the entropy values are estimated from training data and that Q_m is binary. It follows that growing a tree aimed at reducing uncertainty about $\mathbf{Q}$ is equivalent to finding at each node that query which best splits the data at the node into two equal parts. This results from the fact that maximizing $H(p) = p\log_2(p) + (1-p)\log_2(1-p)$ reduces to minimizing $|p - .5|$.

In this way we generate *shape quantiles* or clusters ignoring the class labels. Still, the tree variable T is highly correlated with the class variable Y. *This would be the case even if the tree were grown from samples representing only some of the shape classes.* In other words, these clustering trees produce a generic quantization of shape space. In fact, the same trees can be used to classify new shapes; see §9.

We have experimented with such trees, using the splitting criterion described above as well as another unsupervised one based on the "question metric"

$$d_{\mathbf{Q}}(\mathbf{x}, \mathbf{x}') = \frac{1}{M} \sum_{m=1}^{M} \delta(Q_m(\mathbf{x}) \neq Q_m(\mathbf{x}')), \quad \mathbf{x}, \mathbf{x}' \in \mathbf{X}$$

where $\delta(...) = 1$ if the statement is true and $\delta(...) = 0$ otherwise. Since $\mathbf{Q}$ leads to Y, it makes sense to divide the data so that each child is as "homogeneous" as possible with respect to $d_{\mathbf{Q}}$; we omit the details. Both clustering methods lead to classification rates which are of course inferior to those obtained with splits determined by separating classes but still surprisingly high; one such experiment is reported in §6.1.

6 Multiple Trees

We have seen that small random subsets of the admissible queries at any node will invariably contain at least one query which is informative about the shape class. What happens if many such trees are constructed using the same training set $\mathcal{L}$? Because the family $\mathbf{Q}$ of queries is so large, and because different queries - tag arrangements - address different aspects of shape, separate trees should provide separate structural descriptions, characterizing the shapes from different "points of view". This is visually illustrated in Figure 9 where the same image is shown with an instance of the pending graph at the terminal node in five different trees. Hence, aggregating the information provided by a family of trees (see §6.1) should yield more accurate and more robust classification. This will be demonstrated in experiments throughout the remainder of the paper.

Generating multiple trees by randomization was proposed in Geman, Amit & Wilder (1996). Previously, other authors had advanced other methods for generating multiple trees. One of the earliest was "weighted voting trees" Casey & Jih (1983); Shlien (1990) uses different splitting criteria; Breiman (1994) uses bootstrap replicates of $\mathcal{L}$; and Dietterich & Bakiri (1995) introduce the novel idea of replacing the multiclass learning problem by a family of 2-class problems, dedicating a tree to each of these. Most of these papers deal with fixed size feature vectors and coordinate-based questions. All authors report gains in accuracy and stability.

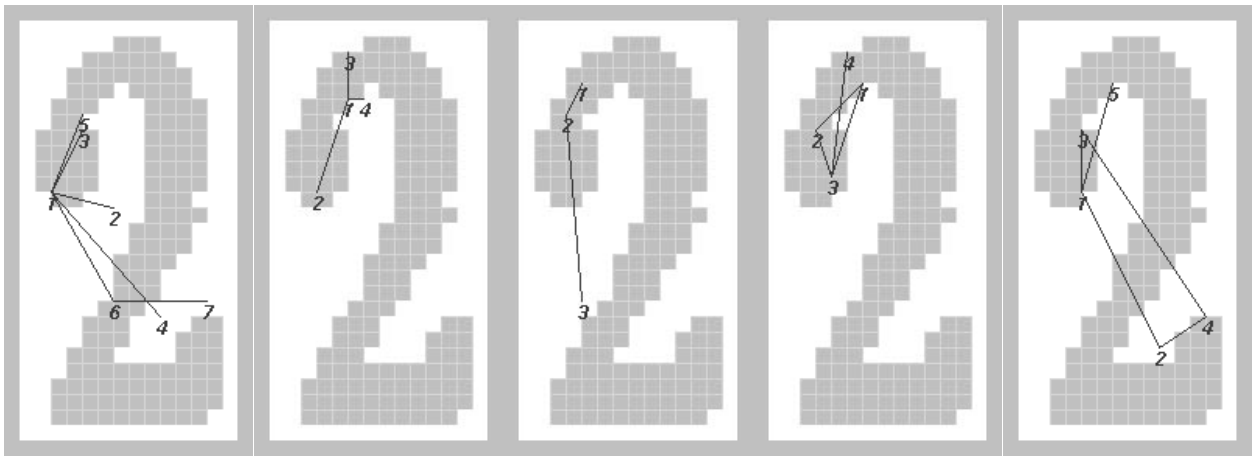

Figure 9: Graphs found in an image at terminal nodes of five different trees

6.1 Aggregation

Suppose we are given a family of trees $T_1, ..., T_N$. The best classifier based on these is

$$\hat{Y}_A = \arg\max_c P(Y = c|T_1, ..., T_N)$$

but this is not feasible (see §6.4). Another option would be to regard the trees as high-dimensional inputs to standard classifiers. We tried that with CART, linear and nonlinear discriminant analysis, k-means clustering and nearest neighbors, all without improvement over simple averaging (see below) for the amount of training data we used.

By "averaging" we mean the following. Let $\mu_{n,\tau}(c)$ denote the posterior distribution $P(Y = c|T_n = \tau), n = 1, \ldots, N, c = 1, \ldots, K$, where τ denotes a terminal node. We write μ_{T_n} for the random variable μ_{n,T_n}. These probabilities are the parameters of the system and the problem of estimating them will be discussed in §6.4. Define

$$\bar{\mu}(\mathbf{x}) = \frac{1}{N}\sum_{n=1}^{N} \mu_{T_n(\mathbf{x})},$$

the *arithmetic average* of the distributions at the leaves reached by $\mathbf{x}$. The mode of $\bar{\mu}(\mathbf{x})$ is the class assigned to the data point $\mathbf{x}$, i.e.

$$\hat{Y}_S = \arg\max_c \bar{\mu}_c.$$

Using a training database of 32 samples per symbol from the distribution described in §2 we grew $N = 100$ trees of average depth $d = 10$, and tested the performance on a test set of 5

samples per symbol. The classification rate was 96%. This experiment was repeated several times with very similar results. On the other hand growing one hundred *unsupervised* trees of average depth 11 and using the labeled data only to estimate the terminal distributions a classification rate of 94.5% was achieved.

6.2 Dependence on the Training Set

The performance of classifiers constructed from training samples can be adversely affected by over-dependence on the particular sample. One way to measure this is to consider the population of *all* training sets $\mathcal{L}$ of a particular size and to compute, for each data point $\mathbf{x}$, the *average* $E_{\mathcal{L}} e_{\mathcal{L}}(\mathbf{x})$, where $e_{\mathcal{L}}$ denotes the error at $\mathbf{x}$ for the classifier made with $\mathcal{L}$. (These averages may then be further averaged over $\mathbf{X}$.) The average error decomposes into two terms, one corresponding to "bias" and the other to "variance" (Geman, Bienenstock & Doursat (1992)). Roughly speaking, the bias term captures the *systematic* errors of the classifier design and the variance term measures the error component due to *random fluctuations* from $\mathcal{L}$ to $\mathcal{L}$. Generally, parsimonious designs (e.g., those based on relatively few unknown parameters) yield low variance but highly biased decision boundaries, whereas complex nonparametric classifiers (e.g., neural networks with many parameters) suffer from high variance, at least without enormous training sets. Good generalization requires striking a balance. See Geman et al. (1992) for a comprehensive treatment of the "bias/variance dilemma"; see also the discussions in Breiman (1994), Kong & Dietterich (1995) and Raudys & Jain (1991).

One simple experiment was carried out to measure the dependence of our classifier $\hat{Y}_S$ on the training sample; we did not systematically explore the decomposition mentioned above. We made 10 sets of 20 trees from ten different training sets, each consisting of 32 samples per symbol. The average classification rate was 85.3%; the standard deviation was .8%. Table 2 shows the number of images in the test set correctly labeled by j of the classifiers, $j = 0, 1, ..., 10$. For example, we see that 88% of the test points are correctly labeled at least six out of ten times. Taking the plurality of the ten classifiers improves the classification rate to 95.5% so there is some pointwise variability among the classifiers. However, the decision boundaries and overall performance are fairly stable with respect to $\mathcal{L}$.

No. of correct classifiers	0	1	2	3	4	5	6	7	8	9	10
No. of points	9	11	20	29	58	42	59	88	149	237	763

Table 2: Number of points as function of number of correct classifiers

We attribute the relatively small variance component to the aggregation of many weakly dependent trees, which in turn results from randomization. The bias issue is more complex, and we have definitely noticed certain types of "structural errors" in our experiments with handwritten digits from the NIST database, for example certain styles of writing are systematically misclassified despite the randomization effects.

6.3 Relative Error Rates

Due to estimation error we favor many trees of modest depth over a few deep ones, even at the expense of *theoretically* higher error rates were perfect estimation possible. In this section we analyze those error rates for some of the alternative classifiers discussed above in the asymptotic case of infinite data and assuming the total number of features examined is held fixed, presumably large enough to guarantee low approximation error. The implications for finite data are outlined in §6.4.

Instead of making N trees $T_1, ..., T_N$ of depth D suppose we made just one tree T^* of depth ND; in both cases we are asking ND questions. Of course this is not practical for the values of D and N mentioned above (e.g., $D = 10$, $N = 20$), but it is still illuminating to compare the hypothetical performance of the two methods. Suppose further that the criterion for selecting T^* is to minimize the error rate over all trees of depth ND:

$$T^* = \arg\max_T E[\max_c P(Y = c|T)],$$

where the maximum is over all trees of depth ND. The error rate of the corresponding classifier $\hat{Y}^* = \arg\max_c P(Y = c|T^*)$ is then $e(\hat{Y}^*) = 1 - E[\max_c P(Y = c|T^*)]$. Notice that finding T^* would require the solution of a *global optimization* problem which is generally intractable, accounting for the nearly universal adoption of *greedy* tree-growing algorithms

based on entropy reduction, such as the one we are using. Notice also that minimizing the entropy $H(Y|T)$ or the error rate $P(Y \neq \hat{Y}(T))$ amounts to basically the same thing.

Let $e(\hat{Y}_A)$ and $e(\hat{Y}_S)$ be the error rates of $\hat{Y}_A$ and $\hat{Y}_S$ respectively. Then it is easy to show that

$$e(\hat{Y}^*) \leq e(\hat{Y}_A) \leq e(\hat{Y}_S).$$

The first inequality results from the observation that the N trees of depth D could be combined into one tree of depth ND simply be grafting T_2 onto each terminal node of T_1, then grafting T_3 onto each terminal node the new tree, and so forth. The error rate of the tree so-constructed is just $e(\hat{Y}_A)$. However, the error rate of T^* is minimal among all trees of depth ND, and hence is lower than $e(\hat{Y}_A)$. Since $\hat{Y}_S$ is a function of $T_1, ..., T_N$, the second inequality follows from a standard argument:

$$\begin{aligned} P(Y \neq \hat{Y}_S) &= E[P(Y \neq \hat{Y}_S | T_1, ..., T_N)] \\ &\geq E[P(Y \neq \arg\max_c P(Y = c | T_1, ..., T_N)) | T_1, ..., T_N)] \\ &= P(Y \neq \hat{Y}_A). \end{aligned}$$

6.4 Parameter Estimation

In terms of tree depth, the limiting factor is *parameter estimation* not computation or storage. The probabilities $P(Y = c|T^*), P(Y = c|T_1, ..., T_N)$ and $P(Y = c|T_n)$ are unknown and must be estimated from training data. In each of the cases $\hat{Y}^*$ and $\hat{Y}_A$ there are $K \times 2^{ND}$ parameters to estimate (recall K is the number of shape classes) whereas for $\hat{Y}_S$ there are $K \times N \times 2^D$ parameters. Moreover, the number of data points in $\mathcal{L}$ available *per* parameter is $\| \mathcal{L} \| / (K2^{ND})$ in the first two cases and $\| \mathcal{L} \| / (K2^D)$ with aggregation.

For example, consider the family of $N = 100$ trees described in §6.1 which were used to classify the $K = 293$ Latex symbols. Since the average depth is $D = 8$, then there are approximately $100 \times 2^8 \times 293 \sim 7.5 \times 10^6$ parameters, although most of these are nearly zero. *Indeed in all experiments reported below only the largest 5 elements of $\mu_{n,\tau}$ are estimated; the rest are set to zero.* It should be emphasized, however, that the parameter estimates can be refined indefinitely using additional samples from $\mathbf{X}$, a form of *incremental learning*; see §9.

For $\hat{Y}_A = \arg\max_c P(Y = c|T_1, ..., T_N)$ the estimation problem is overwhelming, at least without assuming conditional independence or some other model for dependence. This was illustrate when we tried to compare the magnitudes of $e(\hat{Y}_A)$ with $e(\hat{Y}_S)$ in a simple case. We created $N = 4$ trees of depth $D = 5$ to classify just the first $K = 10$ symbols, which are the ten digits. The trees were constructed using a training set $\mathcal{L}$ with 1000 samples per symbol. Using $\hat{Y}_S$, the error rate on $\mathcal{L}$ was just under 6%; on a test set $\mathcal{V}$ of 100 samples per symbol the error rate was 7%.

Unfortunately $\mathcal{L}$ was not large enough to estimate the full posterior given the four trees. Consequently, we tried using $1000, 2000, 4000, 10000$ and 20000 samples per symbol for estimation. With two trees, the error rate was consistent from $\mathcal{L}$ to $\mathcal{V}$, even with 2000 samples per symbol, and was slightly lower than $e(\hat{Y}_S)$. With three trees, there was a significant gap between the (estimated) $e(\hat{Y}_A)$ on $\mathcal{L}$ and $\mathcal{V}$, even with 20000 samples for symbol; the estimated value of $e(\hat{Y}_A)$ on $\mathcal{V}$ was 6% compared with 8% for $e(\hat{Y}_S)$. With four trees, and using 20000 samples per symbol, the estimate of $e(\hat{Y}_A)$ on $\mathcal{V}$ was about 6%, and about 1% on $\mathcal{L}$. It was only 1% better than $e(\hat{Y}_S)$, which was 7% and required only 1000 samples per symbol.

We did not go beyond 20000 samples per symbol. Ultimately $\hat{Y}_A$ will do better, but the amount of data needed to demonstrate this is prohibitive, even for four trees. Evidently, the same problems would be encountered in trying to estimate the error rate for a very deep tree.

7 Performance Bounds

We divide this into two cases: individual trees and multiple trees. Most of the analysis for individual trees concerns a rather ideal case ("twenty questions") in which the shape classes are "atomic"; there is then a natural metric on shape classes and one can obtain bounds on the expected uncertainty after a given number of queries in terms of this metric and an initial distribution over classes. The key issue for multiple trees is weak dependence and the analysis there is focused on the dependence structure among the trees.

7.1 Individual Trees: Twenty Questions

Suppose first that each shape class or hypothesis c is "atomic" - consists of a *single* atom of $\mathbf{Q}$ (as defined in §4 above). In other words each "hypothesis" c has a unique codeword which we denote by $\mathbf{Q}(c) = (Q_1(c), ..., Q_M(c))$, so that $\mathbf{Q}$ is determined by Y. This setting corresponds exactly to a mathematical version of the "twenty questions game." There is also an initial distribution $\nu(c) = P(Y = c)$. For each $c = 1, ..., K$, the binary sequence $(Q_m(1), ..., Q_m(K))$ determines a subset of hypotheses - those which answer "yes" to query Q_m. Since the codewords are distinct, asking enough questions will eventually determine Y. The mathematical problem is to find the ordering of the queries which minimizes the mean number of queries needed to determine Y, or minimizes the mean uncertainty about Y after a fixed number of queries. The best known example is when there is a query for *every* subset of $\{1, ..., K\}$, so that $M = 2^K$. The optimal strategy is given by the Huffman code, in which case the mean number of queries required to determine Y lies in the interval $[H(Y), H(Y) + 1)$; see Cover & Thomas (1991).

Suppose $\pi_1, ..., \pi_k$ represent the indices of the first k queries. The mean residual uncertainty about Y after k queries is then

$$\begin{aligned} H(Y|Q_{\pi_1}, ..., Q_{\pi_k}) &= H(Y, Q_{\pi_1}, ..., Q_{\pi_k}) - H(Q_{\pi_1}, ..., Q_{\pi_k}) \\ &= H(Y) - H(Q_{\pi_1}, ..., Q_{\pi_k}) \\ &= H(Y) - \left(H(Q_{\pi_1}) + H(Q_{\pi_2}|Q_{\pi_1}) + ... + H(Q_{\pi_k}|Q_{\pi_1}, ..., Q_{\pi_{k-1}})\right). \end{aligned}$$

Consequently, if at each stage there is a query which divides the "active" hypotheses into two groups such that the mass of the smaller group is at least β $(0 < \beta \leq .5)$ then $H(Y|Q_{\pi_1}, ..., Q_{\pi_k}) \leq H(Y) - kH(\beta)$. The mean decision time is roughly $H(Y)/H(\beta)$. In all unsupervised trees we produced, we found $H(Q_{\pi_k}|Q_{\pi_1}, ..., Q_{\pi_{k-1}})$ to be greater than .99 (corresponding to $\beta \approx .5$) at 95% of the nodes.

If assumptions are made about the degree of separation among the codewords, one can obtain bounds on mean decision times and on the expected uncertainty after a fixed number of queries, in terms of the prior distribution ν. For these types of calculations, it is easier to work with the "Hellinger" measure of uncertainty than with Shannon entropy. Given a

probability vector $\mathbf{p} = (p_1, ..., p_J)$, define

$$G(\mathbf{p}) = \sum_{j \neq i} \sqrt{p_j}\sqrt{p_i}$$

and define $G(Y), G(Y|B_t)$ and $G(Y|B_t, Q_m)$ the same way as with the entropy function H. (G and H have similar properties; for example, G is minimized on a point mass, maximized on the uniform distribution, and it follows from Jensen's inequality that $H(\mathbf{p}) \leq \log_2[G(\mathbf{p})+1]$.) The initial amount of uncertainty is

$$G(Y) = \sum_{c \neq c'} \nu^{1/2}(c)\nu^{1/2}(c').$$

For any subset $\{m_1, ..., m_k\} \subset \{1, ..., M\}$, using Bayes rule and the fact that $P(\mathbf{Q}|Y)$ is either 0 or 1, we obtain

$$G(Y|Q_{m_1}, ..., Q_{m_k}) = \sum_{c \neq c'} \prod_{i=1}^{k} \delta(Q_{m_i}(c) = Q_{m_i}(c'))\nu^{1/2}(c)\nu^{1/2}(c').$$

Now suppose we average $G(Y|Q_{m_1}, ..., Q_{m_k})$ over *all* subsets $\{m_1, ..., m_k\}$ (allowing repetition). The average is

$$\begin{aligned} M^{-k} \sum_{(m_1,...,m_k)} G(Y|Q_{m_1}, ..., Q_{m_k}) &= \sum_{c \neq c'} M^{-k} \sum_{(m_1,...,m_k)} \prod_{i=1}^{k} \delta(Q_{m_i}(c) = Q_{m_i}(c'))\nu^{1/2}(c)\nu^{1/2}(c') \\ &= \sum_{c \neq c'} (1 - d_{\mathbf{Q}}(c, c'))^k \nu^{1/2}(c)\nu^{1/2}(c') \end{aligned}$$

Consequently, any "better than average" subset of queries satisfies

$$G(Y|Q_{m_1}, ..., Q_{m_k}) \leq \sum_{c \neq c'} (1 - d_{\mathbf{Q}}(c, c'))^k \nu^{1/2}(c)\nu^{1/2}(c') \qquad (*)$$

If $\gamma = \min_{c,c'} d_{\mathbf{Q}}(c, c')$, then the residual uncertainty is at most $(1-\gamma)^k G(Y)$. In order to disambiguate K hypotheses under a uniform starting distribution (in which case $G(Y) = K - 1$) we would need approximately $k \approx -\frac{\log K}{\log(1-\gamma)}$ queries, or $k \approx \log K/\gamma$ for small γ. (This is clear without the general result in (*) since we eliminate a fraction γ of the remaining hypotheses with each new query.) This value of k is too large to be practical for realistic values of γ (due to storage, etc.) but does express the "divide-and-conquer" nature of recursive partitioning in the logarithmic dependence on the number of hypotheses.

Needless to say, the "compound" case is the only realistic one, where the number of atoms in a shape class is a measure of its complexity. (For example, we would expect many more atoms per handwritten digit class than per printed font class.) In the compound case one can obtain results similar to those mentioned above by considering the degree of homogeneity within classes as well as the degree of separation between classes. For example, the index γ must be replaced by one based on both the maximum distance D_{max} between codewords of the same class and the minimum distance D_{min} between codewords from different classes. Again, the bounds obtained call for trees which are too deep to actually be made, and much deeper than those which are empirically demonstrated to obtain good discrimination. We achieve this in practice due to semi-invariance, guaranteeing that D_{max} is small, and the extraordinary richness of the world of spatial relationships, guaranteeing that D_{min} is large.

7.2 Multiple Trees: Weak Dependence

From a statistical perspective, randomization leads to *weak conditional dependence* among the trees. For example, given $Y = c$, the correlation between two trees T_1 and T_2 is small. In other words, given the class of an image, knowing the leaf of T_1 which is reached would not aid us in predicting the leaf reached in T_2.

In this section we analyze the dependence structure among the trees and obtain a crude lower bound on the performance of the classifier $\hat{Y}_S$ for a *fixed* family of trees $T_1, ..., T_N$ constructed from a *fixed* training set $\mathcal{L}$. Thus we are *not* investigating the asymptotic performance of $\hat{Y}_S$ as either $N \to \infty$ or $|\mathcal{L}| \to \infty$. With infinite training data a tree could be made arbitrarily deep, leading to arbitrarily high classification rates since nonparametric classifiers are generally strongly consistent.

Let $E_c\bar{\mu} = (E_c\bar{\mu}(1), ..., E_c\bar{\mu}(K))$ denote the *mean* of $\bar{\mu}$ conditioned on $Y = c$: $E_c\bar{\mu}(d) = \frac{1}{N}\sum_{i=1}^{N} E(\mu_{T_n}(d)|Y = c)$. We make three assumptions about the mean vector, all of which turn out to be true in practice:

1. $\arg\max_d E_c\bar{\mu}(d) = c$;

2. $E_c\bar{\mu}(c) = \alpha_c >> 1/K$;

3. $E_c\bar{\mu}(d) \sim (1 - \alpha_c)/(K - 1)$

The validity of the first two is clear from the table below. The last assumption says that the amount of mass in the *mean* aggregate distribution which is off the true class tends to be uniformly distributed over the other classes.

Let S_K denote the K-dimensional simplex (probability vectors in $\mathbf{R}^K$) and let $U_c = \{\mu : \arg\max_d \mu(d) = c\}$, an open convex subset of S_K. Define ϕ_c to be the (Euclidean) distance from $E_c\bar{\mu}$ to ∂U_c, the boundary of U_c. Clearly $\|\mu - E_c\bar{\mu}\| < \phi_c$ implies that $\arg\max_d \mu(d) = c$, where $\|\cdot\|$ denotes Euclidean norm. This is used below to bound the misclassification rate. First, however, we need to compute ϕ_c. Clearly,

$$\partial U_c = \cup_{d:d\neq c}\{\mu \in U_c : \mu(c) = \mu(d)\}.$$

From symmetry arguments a point in ∂U_c which achieves the minimum distance to $E_c\bar{\mu}$ will lie in each of the sets in the union above. A straightforward computation involving orthogonal projections than yields $\phi_c = (\alpha_c K - 1)/\sqrt{2}(K-1)$.

Using Chebyshev's inequality a crude upper bound on the misclassification rate for class c is obtained as follows.

$$\begin{aligned}
P(\hat{Y}_S \neq c|Y = c) &= P(\mathbf{x} : \arg\max_d \bar{\mu}(\mathbf{x}, d) \neq c|Y = c) \\
&\leq P(\|\bar{\mu} - E_c\bar{\mu}\| > \phi_c|Y = c) \\
&\leq \frac{1}{\phi_c^2} E\|\bar{\mu} - E_c\bar{\mu}\|^2 \\
&= \frac{1}{\phi_c^2 N^2}\sum_{d=1}^{K}[\sum_{n=1}^{N} Var(\mu_{T_n}(d)|Y = c) + \sum_{n\neq m} Cov(\mu_{T_n}(d), \mu_{T_m}(d)|Y = c)].
\end{aligned}$$

Let η_c denote the sum of the conditional variances and let γ_c the sum of the conditional covariances, both averaged over the trees:

$$\frac{1}{N}\sum_{n=1}^{N}\sum_{d=1}^{K} Var(\mu_{T_n}(d)|Y = c) = \eta_c \qquad \frac{1}{N^2}\sum_{n\neq m}\sum_{d=1}^{K} Cov(\mu_{T_n}(d), \mu_{T_m}(d)|Y = c) = \gamma_c.$$

We see that

$$P(\hat{Y}_S \neq c|Y = c) \leq \frac{\gamma_c + \eta_c/N}{\phi_c^2} = \frac{2(\gamma_c + \eta_c/N)(K-1)^2}{(\alpha_c K - 1)^2}.$$

Since η_c/N will be small compared with γ_c, the key parameters are α_c and γ_c. This inequality yields only coarse bounds. However, it is clear that, under the assumptions above, high

Class	0	1	2	3	4	5	6	7	8	9
α_c	0.66	0.86	0.80	0.74	0.74	0.64	0.56	0.86	0.49	0.68
γ_c	0.03	0.01	0.01	0.01	0.03	0.02	0.04	0.01	0.02	0.01
e_c	0.14	0.04	0.03	0.04	0.11	0.13	0.32	0.02	0.23	0.05

Table 3: Estimates of α_c, γ_c and e_c for 10 classes

classification rates are feasible as long as γ_c is sufficiently small and α_c is sufficiently large, *even if the estimates μ_{T_n} are poor.*

Observe that the N trees form a simple random sample from some large population $\mathcal{T}$ of trees under a suitable distribution on $\mathcal{T}$. This is due to the randomization aspect of tree construction. (Recall that at each node the splitting rule is chosen from a small *random sample* of queries.) Both $E_c\bar{\mu}$ and the sum of variances are sample means of functionals on $\mathcal{T}$. The sum of the covariances has the form of a U-statistic. Since the trees are drawn independently, and since the range of the corresponding variables is very small (typically less than 1), standard statistical arguments imply that these sample means are close to the corresponding population means for a moderate number N of trees, say tens or hundreds. In other words, $\alpha_c \sim E_{\mathcal{T}} E_{\mathbf{X}}(\mu_T(c)|Y=c)$ and $\gamma_c \sim E_{\mathcal{T}\times\mathcal{T}} \sum_{d=1}^{K} Cov_{\mathbf{X}}(\mu_{T_1}(d), \mu_{T_2}(d)|Y=c)$. Thus the conditions on α_c and γ_c translate into conditions on the corresponding expectations over $\mathcal{T}$, and the performance variability among the trees can be ignored.

Table 3 shows some estimates of α_c and γ_c and the resulting bound e_c on the misclassification rate $P(\hat{Y}_S \neq c|Y=c)$. Ten pairs of random trees were made on ten classes to estimate γ_c and α_c. Again, the bounds are crude; they could be refined by considering higher-order joint moments of the trees.

8 Generalization

For convenience, we will consider two types of generalization, referred to as "interpolation" and "extrapolation." Our use of these terms may not be standard, and is decidedly *ad hoc*. Interpolation is the easier case, in which both the training and testing samples are randomly

drawn from $(\mathbf{X}, P)$ and the number of training samples is sufficiently large to "cover" the space $\mathbf{X}$. Consequently, for most test points, the classifier is being asked to "interpolate" among "nearby" training points.

By extrapolation we mean situations in which the training samples do not "represent" the space from which the test samples are drawn. Examples of this would be i) training on a very small number of samples per symbol (e.g., one); ii) using different perturbation models to generate the training and test sets, perhaps adding more severe scaling or skewing; iii) degrading the test images with correlated noise or lowering the resolution. Another example of this occurred at the first NIST competition Wilkinson, Geist, Janet, Grother, Gurges, Creecy, Hammond, Hull, Larsen, Vogl, & Wilson (1992); the hand-printed digits in the test set were written by a different population than those in the distributed training set. (Not surprisingly, the distinguishing feature of the winning algorithm was the size and diversity of the actual samples used to train the classifier.) One way to characterize such situations is to regard P as a *mixture distribution* $P = \sum_i \alpha_i P_i$, where the P_i might correspond to writer populations, perturbation models, levels of degradation, etc. In complex visual recognition problems the number of terms might be very large, but the training samples might be drawn from relatively few of the P_i and hence represent a *biased sample* from P.

In order to gauge the difficulty of the problem, we shall consider the performance of two other classifiers, based on k-nearest-neighbor classification with $k = 5$, which was more or less optimal in our setting. (Using nearest-neighbors as a benchmark is common; see e.g., Geman et al. (1992), Khotanzad & Lu (1991)). Let "$NN(raw)$" refer to nearest-neighbor classification based on Hamming distance in (binary) image space, i.e., between bitmaps. This is clearly the *wrong* metric, but helps to calibrate the difficulty of the problem. Of course this metric is entirely "blind" to invariance, but is not entirely unreasonable when the symbols nearly fill the bounding box and the degree of perturbation is limited.

Let "$NN(\mathbf{B})$" refer to nearest-neighbor classification based on the binary tag arrangements. Thus two images $\mathbf{x}$ and $\mathbf{x}'$ are compared by evaluating $Q(\mathbf{x})$ and $Q(\mathbf{x}')$ for all $Q \in \mathbf{B}_0 \subset \mathbf{B}$ and computing the Hamming distance between the corresponding binary sequences. $\mathbf{B}_0$ was chosen as the subset of binary tag arrangements which split $\mathbf{X}$ to within 5% of 50-50. There were 1510 such queries out of the total 15376 binary tag arrangements.

Sample size	Trees	$NN(\mathbf{B})$	$NN(raw)$
1	44	11	5
8	87	57	31
32	96	74	55

Table 4: Classification rates in percent for various training sample sizes compared with nearest neighbor methods.

Due to invariance and other properties, we would expect this metric to work better than Hamming distance in image space, and of course it does (see below).

8.1 Interpolation

One hundred (randomized) trees were constructed from a training data set with 32 samples for each of the $K = 293$ symbols. The average classification rate per tree on a test set $\mathcal{V}$ consisting of 100 samples per symbol is 27%. However the performance of the classifier $\hat{Y}_S$ based on 100 trees is 96%. This clearly demonstrates the weak dependence among randomized trees (as well as the discriminating power of the queries). With the $NN(\mathbf{B})$-classifier, the classification rate was 74%; with $NN(raw)$, the rate is 55%; see Table 4. All of these rates are on the test set.

When the only random perturbations are non-linear (i.e., no scaling, rotation or skew), there is not much standardization that can be done to the raw image; see Figure 10. With 32 samples per symbol, $NN(raw)$ climbs to 76%, whereas the trees reach 98.5%.

8.2 Extrapolation

We also grew trees using *only* the original prototypes $\mathbf{x}_c^*, c = 1, ..., 293$, recursively dividing this group until pure leaves were obtained. Of course the trees are relatively shallow. In this case, only about half the symbols in $\mathbf{X}$ could then be recognized; see Table 4.

The 100 trees grown with 32 samples per symbol were tested on samples which exhibit a

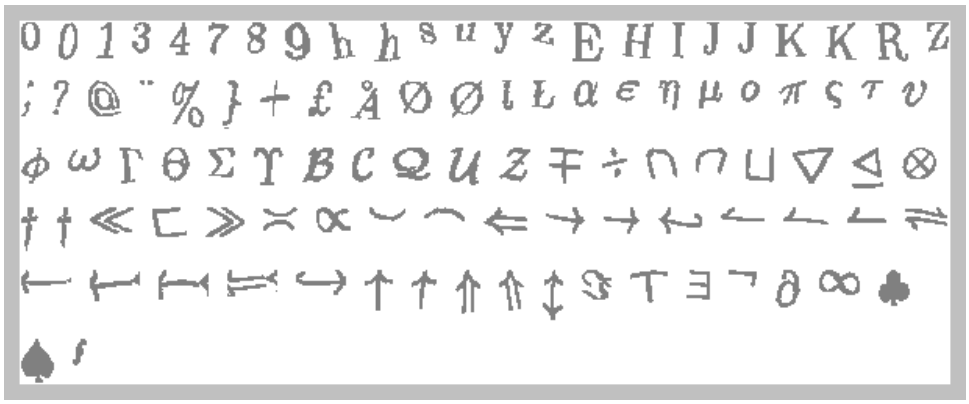

Figure 10: Latex symbols perturbed only with non-linear deformations

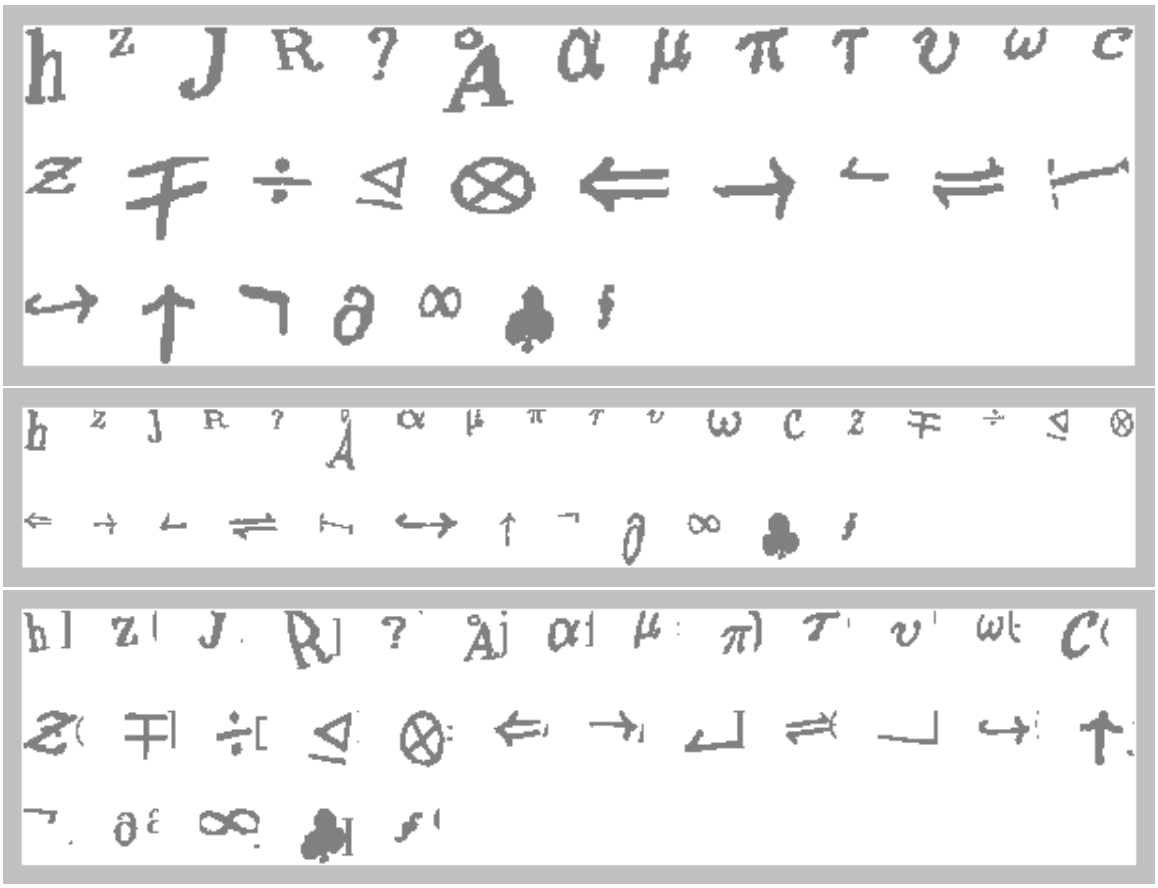

Figure 11: Top: Upscaling. Middle: Downscaling. Bottom: Clutter.

Type of Perturbation	Trees	$NN(\mathbf{B})$	$NN(raw)$
Original	96	74	55
Upscaling	88	57	0
Downscaling	80	52	0
Spot Noise	71	28	57
Clutter	74	27	59

Table 5: Classification rates in percent for various perturbations

greater level of distortion or variability than described up to this point. The results appear in Table 5. "Upscaling" (resp. "downscaling") refers to uniform sampling between the original scale and twice (resp. half) the original scale, as in the top (resp. middle) panel of Figures 11; "spot noise" refers to adding correlated noise (top panel of Figure 8). Clutter (bottom panel of Figure 11) refers to the addition of pieces of other symbols in the image. All of these distortions came in addition to the random nonlinear deformations, skew and rotations. Downscaling creates more confusions due to extreme thinning of the stroke. Notice that the $NN(\mathbf{B})$ classifier falls apart with spot noise. The reason is the number of false positives: tags due to the noise induce random occurrences of simple arrangements. In contrast, complex arrangements $\mathbf{A}$ are far less likely to be found in the image by pure chance; therefore, chance occurrences are "weeded out" deeper in the tree.

8.3 Note

The purpose of all the experiments in this paper is to illustrate various attributes of the recognition strategy. No effort was made to optimize the classification rates. In particular, the same tags and tree-making protocol were used in every experiment. Experiments were repeated several times; the variability was negligible.

One direction which appears promising is explicitly introducing different protocols from tree to tree in order to decrease the dependence. One small experiment was carried out in this direction. All the images were subsampled to half the resolution, e.g. 32×32 images

become 16×16. A tag tree was made with 4×4 subimages from the subsampled data set, and one hundred trees are grown using the subsampled training set. The output of these trees is combined with the output of the original trees on the test data. No change in the classification rate was observed for the original test set. For the test set with spot noise, the two sets of trees each had a classification rate of about 72%. Combined however they yield a rate of 86%. Clearly there is a significant potential for improvement in this direction.

9 Incremental Learning and Universal Trees

As mentioned earlier, the parameters $\mu_{n,\tau}(c) = P(Y = c|T_n = \tau)$ can be incrementally updated with new training samples. Given a set of trees, the actual counts from the training set (instead of the normalized distributions) are kept in the terminal nodes τ. When a new labeled sample is obtained, it can be dropped down each of the trees and the corresponding counters incremented. There is no need to keep the image itself.

This separation between tree construction and parameter estimation is crucial. It provides a mechanism for gradually learning to recognize an increasing number of shapes. Trees originally constructed with training samples from a small number of classes can eventually be updated to accommodate new classes; namely the parameters can be re-estimated. In addition, as more data points are observed the estimates of the terminal distributions can be perpetually refined. Finally, the trees can be *further deepened* as more data becomes available. Each terminal node is assigned a randomly chosen list of minimal extensions of the pending arrangement. The answers to these queries are then calculated and stored for each new labeled sample which reaches that node; again there is no need to keep the sample itself. When sufficiently many samples are accumulated the best query on the list is determined by a simple calculation based on the stored information and the node can then be split.

The adaptivity to additional classes is illustrated in the following experiment. A set of 100 trees was grown with training samples from 50 classes randomly chosen from the full set of 293 classes. The trees were grown to depth 10 just as before (§8). Using the original training set of 32 samples per class for all 293 classes, the terminal distributions were estimated and recorded for each tree. The aggregate classification rate on all 293 classes was about

90%, as compared with about 96% when the full training set is used for both quantization and parameter estimation. Clearly fifty shapes are sufficient to produce a reasonably sharp quantization of the entire shape space.

As for improving the parameter estimates, recall that the 100 trees grown with the pure symbols reached 44% on the test set. The terminal distributions of these trees were then updated using the original training set of 32 samples per symbol. The classification rate on the same test set climbed from 44% to 90%.

10 Fast Indexing

One problem with recognition paradigms such as "hypothesize and test" is determining which particular hypothesis to test. "Indexing" into the shape library is therefore a central issue, especially with methods based on matching image data to model data, and involving large numbers of shape classes. The standard approach in model-based vision is to flag plausible interpretations by searching for "key features" or discriminating "parts" in hierarchical representations.

Indexing efficiency seems to be inversely related to stability with respect to image degradation. Deformable templates are highly robust because they provide a global interpretation for much of the image data. However, a good deal of searching may be necessary to find the "right" template. The method of invariant features lies at the other extreme of this axis: The indexing is "one shot" but there is not much tolerance to distortions of the data.

We have not attempted to formulate this trade-off in a manner susceptible to experimentation. We have noticed, however, that multiple trees appear to offer a reliable mechanism for "fast indexing," at least within the framework of this paper and in terms of narrowing down the number of possible classes. For example, in the original experiment with 96% classification rate, the five highest ranking classes in the aggregate distribution $\bar{\mu}$ contained the true class in all but 4 images in a test set of size 1465 (five samples per class). Even with upscaling, for example, the true label was among the top five in 98% of the cases. These experiments suggest that very high recognition rates could be obtained with "final tests" dedicated to ambiguous cases, as determined, for example, by the mode of the $\bar{\mu}$.

11 Handwritten Digit Recognition

The OCR problem has many variations and the literature is immense; one recent survey is Mori, Suen & Yamamoto (1992). In the area of *handwritten* character recognition, perhaps the most difficult problem is the recognition of unconstrained script; zip codes and hand-drawn checks also present a formidable challenge. The problem we consider is off-line recognition of isolated binary digits. Even this special case has attracted enormous attention, including a competition sponsored by the National Institute of Standards and Technology (NIST) (Wilkinson et al. (1992)), and there is still no solution that matches human performance, nor even one that is commercially viable except in restricted situations. For comparisons among methods see Bottou et al. (1994) and the lucid discussion in Brown et al. (1993). The best reported rates seem to be those obtained by the ATT research group, up to 99.3% by training and testing on composites of the NIST training and test sets (Bottou et al. (1994)).

We present a brief summary of the results of experiments using the tree based shape quantization method to the NIST data base. For a more detailed description see Geman et al. (1996). Our experiments were based on portions of the NIST database, which consists of approximately $223,000$ binary images of isolated digits written by more than 2000 writers. The images vary widely in dimensions, ranging from about twenty to one hundred rows, and also vary in stroke thickness and other attributes. We used $100,000$ for training and $50,000$ for testing. A random sample from the test set is shown in Figure 12.

All results reported in the literature utilize rather sophisticated methods of preprocessing, such as thinning, slant correction and size normalization. For the sake of comparison we did several experiments using a crude form of slant correction and scaling, and no thinning. Twenty-five trees were made. We stopped splitting when the number of data points in the second largest class fell below ten. The depth of the terminal nodes (i.e., number of questions asked per tree) varied widely, the average over trees being 8.8. The average number of terminal nodes was about 600 and the average classification rate (determined by taking the mode of the terminal distribution) was about 91%. The best error rate we achieved with a *single* tree was about 7%.

The classifier was tested in two ways. First, we preprocessed (scaled and slant-corrected)

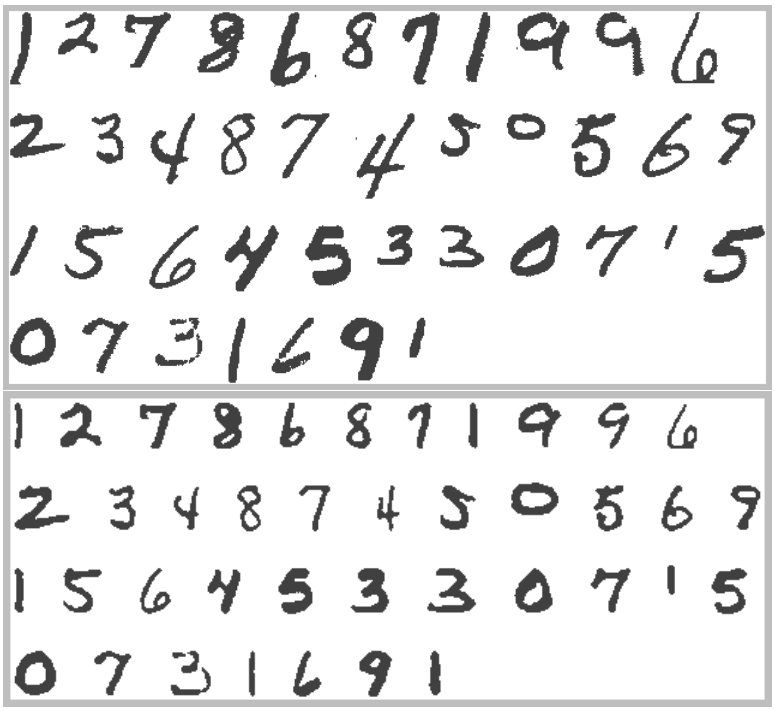

Figure 12: Random sample of test images before (top) and after (bottom) pre-processing.

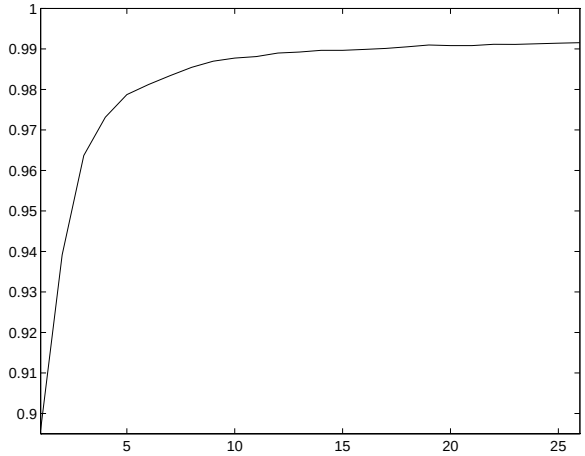

Figure 13: Classification rate vs. Number of trees.

the test set in the same manner as the training set. The resulting classification rate is 99.2% (with no rejection). Figure 13 shows how the classification rate grows with the number of trees. Recall from §6.1 that the estimated class of an image $\mathbf{x}$ is the mode of the aggregate distribution $\bar{\mu}(\mathbf{x})$. A good measure of the confidence in this estimate is the value of $\bar{\mu}(\mathbf{x})$ at the mode, call it $M(\mathbf{x})$. It provides a natural mechanism for rejection by classifying only those images $\mathbf{x}$ for which $M(\mathbf{x}) > m$; no rejection corresponds to $m = 0$. For example, the classification rate is 99.5% with one percent rejection and 99.8% with three percent rejection. Finally, doubling the number of trees makes the classification rates $99.3\%, 99.6\%$ and 99.8% at zero, one and two percent rejection, respectively.

We performed a second experiment in which the test data was not preprocessed in the manner of the training data; in fact, the test images were classified without utilizing the size of the bounding box. This is especially important in the presence of noise and clutter when it is essentially impossible to determine the size of the bounding box. Instead, each test image was classified with the same set of trees at two resolutions (original and halved) and three (fixed) slants. The highest of the resulting six modes determines the classification. The classification rate was 98.9%.

We classify approximately 15 digits per second on a single processor SUN Sparcstation 20 (without special efforts to optimize the code); the time is approximately equally divided between transforming to tags and answering questions. Test data can be dropped down the trees in parallel, in which case classification would become approximately 25 times faster.

12 Comparison with ANN's

The comparison with ANNs is natural in view of their widespread use in pattern recognition (Werbos (1991)) and several common attributes. In particular, neither approach is "model-based" in the sense of utilizing explicit shape models. In addition, both are computationally very efficient in comparison with model-based methods as well as with "memory-based" methods (e.g., nearest neighbors). Finally, in both cases performance is diminished by over-dedication to the training data ("over-fitting") and problems result from deficient approximation capacity and/or parameter estimation. Two key differences are described in

the following two subsections.

12.1 The Posterior Distribution and Generalization Error

It is not clear how a feature vector such as $\mathbf{Q}$ could be accommodated in the ANN framework. Direct use is not likely to be successful since ANNs based on very high dimensional input suffer from poor generalization (Baum & Haussler (1989)) and very large training sets are then necessary in order to approximate complex decision boundaries (Raudys & Jain (1991)).

The role of the posterior distribution $P(Y = c|\mathbf{Q})$ is more explicit in our approach, leading to a somewhat different explanation of the sources of error. In our case the approximation error results from replacing the entire feature set by an adaptive subset (or really many subsets, one per tree). The difference between the full posterior and the tree-based approximations can be thought of as the analog of approximation error in the analysis of the learning capacity of ANNs; see e.g. Niyogi & Girosi (1996). In that case one is interested in the set of functions which can be generated by the family of networks of a fixed architecture. Some work has centered on approximation of the particular function $\mathbf{Q} \rightarrow P(Y = c|\mathbf{Q})$. For example, Lee et al. (1991) consider least-squares approximation to the Bayes rule under conditional independence assumptions on the features; the error vanishes as the number of hidden units goes to infinity.

Estimation error in our case results from an inadequate amount of data to estimate the conditional distributions at the leaves of the trees. The analogous situation for ANNs is well-known: Even for a network of unlimited capacity there is still the problem of estimating the weights from limited data (Niyogi & Girosi (1996)).

Finally, classifiers based on ANNs address classification in a *less* nonparametric fashion. In contrast to our approach, the classifier has an explicit functional (input-output) representation from the feature vector to $\hat{Y}$. Of course a great many parameters may need to be estimated in order to achieve low approximation error, at least for very complex classification problems such as shape recognition. This in turn leads to a relatively large variance component in the bias/variance decomposition. In view of these difficulties, the small error rates achieved by ANNs on handwritten digits and other shape recognition problems is noteworthy.

12.2 Invariance and the Visual System

The structure of ANNs for shape classification is partially motivated by observations about processing in the visual cortex. Thus the emphasis has been on parallel processing and the local integration ("funneling") of information from one layer to the next. Since this local integration is done in a spatially stationary manner, translation invariance is achieved. The most successful example in the context of handwritten digit is the convolution neural net LeNET (LeCun et al. (1990)). Scale, deformation, and other forms of invariance are achieved to a lesser degree, partially from using gray level data and soft thresholds, but mainly by utilizing very large training sets and sophisticated image normalization procedures.

The neocognitron (Fukushima & Miyake (1982), Fukushima & Wake (1991)) is an interesting mechanism to achieve a degree of robustness to shape deformations and scale changes in addition to translation invariance. This is done using hidden layers which carry out a local ORing operation mimicking the operation of the complex neurons in V1. These layers are referred to as "C-type" layers in Fukushima & Miyake (1982) . A dual resolution version (Gochin (1994)) is aimed at achieving scale invariance over a larger range.

Spatial stationarity of the connection weights, the C-layers and the use of multiple resolutions are all forms of ORing aimed at achieving invariance. The complex cell in V1 is indeed the most basic evidence of such an operation in the visual system. There is additional evidence for disjunction at a very global scale in the cells of the inferotemporal cortex with very large receptive fields. These respond to certain shapes at all locations in the receptive field, at a variety of scales but also for a variety of non-linear deformations such as changes in proportions of certain parts (see Ito, Tamura, Fujita & Tanaka (1995), Ito, Fujita, Tamura & Tanaka (1994)).

It would be extremely inefficient to obtain such invariance through ORing of responses to each of the variations of these shapes. It would be preferable to carry out the global ORing at the level of primitive generic features which can serve to identify and discriminate between a large class of shapes. We have demonstrated that *global* features consisting of geometric arrangements of *local* responses (tags) can do just that. The detection of any of the tag arrangements described in the preceding sections can be viewed as an extensive and global *ORing* operation. The "ideal" arrangement, e.g. a vertical edge northwest of a

horizontal edge, is tested at all locations, all scales, and at a large range of angles around the northwest vector. A positive response to *any* of these produces a positive answer to the associated query. This leads to a property we have called semi-invariance and allows our method to perform well without preprocessing and without very large training sets. Good performance extends to transformations or perturbations not encountered during training, e.g., scale changes, spot noise, and clutter.

The local responses we employ are very similar to ones employed at the first level of the convolution neural nets, for example in Fukushima & Wake (1991). However at the next level our geometric arrangements are defined *explicitly* and designed to accommodate *a priori* assumptions about shape regularity and variability. In contrast, the features in convolution neural nets are all *implicitly* derived from local inputs to each layer and from the "slack" obtained by local ORing, or soft thresholding, carried out layer by layer. It is not clear that this is sufficient to obtain the required level of invariance, nor is it clear how portable they are from one problem to another.

Evidence for correlated activities of neurons with distant receptive fields is accumulating, not only in V1 but in LGN and in the retina (Neuenschwander & Singer (1996)). Gilbert, Das, Ito, Kapadia & Westheimer (1996) report increasing evidence for more extensive horizontal connections. Large optical point spreads are observed, where subthreshold neural activation appears in an area covering the size of several receptive fields. When an orientation sensitive cell fires in response to a stimulus in its receptive field, subthreshold activation is observed in cells with the same orientation preference in a wide area outside the receptive field. It appears therefore that the integration mechanisms in V1 are more global and definitely more complex than previously thought. Perhaps the features described in this paper can contribute to bridging the gap between existing computational models and new experimental findings regarding the visual system.

13 Conclusion

We have studied a new approach to shape classification and illustrated its performance in high dimensions, both with respect to the number of shape classes and the degree of variability

within classes. The basic premise is that shapes can be distinguished from one another by a sufficiently robust and invariant form of recursive partitioning. This "quantization" of shape space is based on growing binary classification trees using geometric arrangements among local topographic codes as splitting rules. The arrangements are semi-invariant to linear and nonlinear image transformations. As a result, the method generalizes well to samples not encountered during training. In addition, due to the separation between quantization and estimation, the framework accommodates unsupervised and incremental learning.

The codes are primitive and redundant, and the arrangements involve only simple spatial relationships based on relative angles and distances. It is not necessary to isolate distinguished points along shape boundaries or any other special differential or topological structures, nor to perform any form of grouping, matching or functional optimization. Consequently, a virtually infinite collection of discriminating shape features is generated with elementary computations at the pixel level. Since no classifier based on the full feature set is realizable, and since it is impossible to know apriori which features are informative, we have selected features and built trees at the same time by inductive learning. Another data-driven non-parametric method for shape classification is based on ANN's and a comparison was drawn in terms of invariance, and generalization error, leading to the conclusion that prior information plays a relatively greater role in our approach.

We have experimented with the NIST database of handwritten digits and with a synthetic database constructed from linear and nonlinear deformations of about three hundred Latex symbols. Despite a large degree of within-class variability, the setting is evidently simplified since the images are binary and the shapes are isolated.

Looking ahead, the central question is whether our recognition strategy can be extended to more unconstrained scenarios, involving, for example, multiple or solid objects, general poses and a variety of image formation models. We are aware that our approach differs from the standard one in computer vision, which emphasizes viewing and object models, 3D matching, and advanced geometry. Nonetheless, we are currently modifying the strategy in this paper in order to recognize 3D objects against structured backgrounds (e.g., cluttered desktops) based on grey-level images acquired from ordinary video cameras; see Jedynak & Fleuret (1996). We are also looking at applications to document processing, e.g., omni-font

OCR.

Acknowledgment. The authors would like to thank Ken Wilder for many valuable suggestions and for carrying out the experiments on handwritten digit recognition.

References

Baum, E. B. & Haussler, D. (1989), 'What size net gives valid generalization?', *Neural Comp.* **1**, 151–160.

Binford, T. O. & Levitt, T. S. (1993), Quasi-invariants: Theory and exploitation, *in* 'Proc. Image Understanding Workshop', Washington D.C., pp. 819–828.

Bottou, L., Cortes, C., Denker, J. S., Drucker, H., Guyon, I., Jackel, L. D., LeCun, Y., Muller, U. A., Sackinger, E., Simard, P. & Vapnik, V. (1994), Comparison of classifier methods: a case study in handwritten digit recognition, *in* 'Proc. IEEE Inter. Conf. on Pattern Recognition', pp. 77–82.

Breiman, L. (1994), Bagging predictors, Technical Report 451, Department of Statistics, University of California, Berkeley.

Breiman, L., Friedman, J., Olshen, R. & Stone, C. (1984), *Classification and Regression Trees*, Wadsworth, Belmont, CA.

Brown, D., Corruble, V. & Pittard, C. L. (1993), 'A comparison of decision tree classifiers with backpropagation neural networks for multimodal classification problems', *Pattern Recognition* **26**, 953–961.

Burns, J. B., Weiss, R. S. & Riseman, E. M. (1993), 'View variation of point set and line segment features', *IEEE Trans. PAMI* **15**, 51–68.

Casey, R. G. & Jih, C. R. (1983), 'A processor-based ocr system', *IBM Journal of Research and Development* **27**, 386–399.

Casey, R. G. & Nagy, G. (1984), 'Decision tree design using a probabilistic model', *IEEE Trans. Information Theory* **30**, 93–99.

Cover, T. M. & Thomas, J. A. (1991), *Elements of Information Theory*, John Wiley, New York.

Dietterich, T. G. & Bakiri, G. (1995), 'Solving multiclass learning problems via error-correcting output codes', *J. Artificial Intell. Res.* **2**, 263–286.

Forsyth, D., Mundy, J. L., Zisserman, A., Coelho, C., Heller, A. & Rothwell, C. (1991), 'Invariant descriptors for 3-d object recognition and pose', *IEEE Trans. PAMI* **13**, 971–991.

Friedman, J. H. (1973), 'A recursive partitioning decision rule for nonparametric classification', *IEEE Trans. Comput.* **26**, 404–408.

Fukushima, K. & Miyake, S. (1982), 'Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position', *Pattern Recognition* **15**, 455–469.

Fukushima, K. & Wake, N. (1991), 'Handwritten alphanumeric character recognition by the neocognitron', *IEEE Trans. Neural Networks.*

Gelfand, S. B. & Delp, E. J. (1991), On tree structured classifiers, *in* I. K. Sethi & A. K. Jain, eds, 'Artificial Neural Networks and Statistical Pattern Recognition', North Holland, Amsterdam, pp. 51–70.

Geman, D., Amit, Y. & Wilder, K. (1996), Joint induction of shape features and tree classifiers, Technical report, University of Massachusetts, Amherst.

Geman, S., Bienenstock, E. & Doursat, R. (1992), 'Neural networks and the bias/variance dilemma', *Neural Computation* **4**, 1–58.

Gilbert, C. D., Das, A., Ito, M., Kapadia, M. & Westheimer, G. (1996), 'Spatial integration and cortical dynamics', *Proc. Natl. Acad. Sci.* **93**, 615–622.

Gochin, P. M. (1994), 'Properties of simulated neurons from a model of primate inferior temporal cortex', *Cerebral Cortex* **5**, 532–543.

Guo, H. & Gelfand, S. B. (1992), 'Classification trees with neural network feature extraction', *IEEE Trans. Neural Networks* **3**, 923–933.

Hastie, T., Buja, A. & Tibshirani, R. (1995), 'Penalized discriminant analysis', *Annals of Statistics* **23**, 73–103.

Hussain, B. & Kabuka, M. R. (1994), 'A novel feature recognition neural network and its application to character recognition', *IEEE Trans. PAMI* **16**, 99–106.

Ito, M., Fujita, I., Tamura, H. & Tanaka, K. (1994), 'Processing of contrast polarity of visual images of inferotemporal cortex of macaque monkey', *Cerebral Cortex* **5**, 499–508.

Ito, M., Tamura, H., Fujita, I. & Tanaka, K. (1995), 'Size and position invariance of neuronal response in monkey inferotemporal cortex', *Jour. Neuroscience* **73**(1), 218–226.

Jedynak, B. & Fleuret, F. (1996), 3d object recognition from geometric queries, *in* 'Proc. Image'Com 96', Bordeaux, France.

Jung, D.-M. & Nagy, G. (1995), Joint feature and classifier design for ocr, *in* 'Proc. Third Inter. Conf. Document Analysis and Processing, 2, Montreal', pp. 1115–1118.

Khotanzad, A. & Lu, J.-H. (1991), Shape and texture recognition by a neural network, *in* I. K. Sethi & A. K. Jain, eds, 'Artificial Neural Networks and Statistical Pattern Recognition', North Holland, Amsterdam.

Knerr, S., Personnaz, L. & Dreyfus, G. (1992), 'Handwritten digit recognition by neural networks with single- layer training', *IEEE Trans. Neural Networks* **3**, 962–968.

Kong, E. B. & Dietterich, T. G. (1995), Error- correcting output coding corrects bias and variance, Technical report.

Lamdan, Y., Schwartz, J. T. & Wolfson, H. J. (1988), Object recognition by affine invariant matching, *in* 'IEEE Int. Conf. Computer Vision and Pattern Rec.', pp. 335–344.

LeCun, Y., Boser, B., Denker, J. S., Henderson, D., Howard, R. E., Hubbard, W. & Jackel, L. D. (1990), Handwritten digit recognition with a back-propagation network, *in* D. S. Touresky, ed., 'Advances in Neural Information', Vol. 2, Morgan Kauffman, Denver.

Lee, D.-S., Srihari, S. N. & Gaborski, R. (1991), Bayesian and neural network pattern recognition: A theoretical connection and empirical results with handwritten characters, *in* I. K. Sethi & A. K. Jain, eds, 'Artificial Neural Networks and Statistical Pattern Recognition', North Holland, Amsterdam.

Martin, G. L. & Pitman, J. A. (1991), 'Recognizing hand-printed letters and digits using backpropagation learning', *Neural Computation* **3**, 258–267.

Mori, S., Suen, C. Y. & Yamamoto, K. (1992), Historical review of ocr research and development, *in* 'Proc. IEEE', Vol. 80, pp. 1029–1057.

Mundy, J. L. & Zisserman, A. (1992), *Geometric Invariance in Computer Vision*, MIT Press, Cambridge.

Neuenschwander, S. & Singer, W. (1996), 'Long-range synchronization of oscillatory light responses in cat retina and lateral geniculate nucleus', *NAture* **379**(22), 728–733.

Niyogi, P. & Girosi, F. (1996), 'On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions', *Neural Comp.* **8**, 819–842.

Quinlan, J. R. (1986), 'Induction of decision trees', *MachineLearning* **1**, 81–106.

Raudys, S. & Jain, A. K. (1991), Small sample size problems in designing artificial neural networks, *in* I. K. Sethi & A. K. Jain, eds, 'Artificial Neural Networks and Statistical Pattern Recognition', North Holland, Amsterdam.

Reiss, T. H. (1993), *Recognizing Planar Objects Using Invariant Image Features*, Lecture Notes in Computer Science no. 676, Springer Verlag, Berlin.

Sabourin, M. & Mitiche, A. (1992), 'Optical character recognition by a neural network', *Neural Networks* **5**, 843–852.

Sethi, I. K. (1991), Decision tree performance enhancement using an artificial neural network implementation, *in* I. K. Sethi & A. K. Jain, eds, 'Artificial Neural Networks and Statistical Pattern Recognition', North Holland, Amsterdam.

Shlien, S. (1990), 'Multiple binary decision tree classifiers', *Pattern Recognition* **23**, 757–763.

Simard, P. Y., LeCun, Y. L. & Denker, J. S. (1994), Memory-based character recognition using a transformation invariant metric, *in* 'IEEE Inter. Conf. on Pattern Recog.', pp. 262–267.

Werbos, P. (1991), Links between artificial neural networks (ann) and statistical pattern recognition, *in* I. K. Sethi & A. K. Jain, eds, 'Artificial Neural Networks and Statistical Pattern Recognition', North Holland, Amsterdam.

Wilkinson, R. A., Geist, J., Janet, S., Grother, P., Gurges, C., Creecy, R., Hammond, B., Hull, J., Larsen, N., Vogl, T., & Wilson, C. (1992), The first census optical character recognition system conference, Technical Report NISTIR 4912, Nat. Inst. of Standards and Technol., Gaithersburg, MD.