To appear in: Soft-computing and image processing, *S.K. Pal and A. Ghosh, and M. K. Kundu, Eds., Springer-Verlag, 1999.*

Knowledge Reuse Mechanisms for Categorizing Related Image Sets

Kurt D. Bollacker and Joydeep Ghosh

Department of Electrical and Computer Engineering,
University of Texas at Austin,
Austin, TX 78712
{kdb,ghosh}@lans.ece.utexas.edu

Abstract. This chapter introduces the concept of *classifier knowledge reuse* as a means of exploiting domain knowledge taken from old, previously created, relevant classifiers to assist in a new classification task. Knowledge reuse helps in constructing better generalizing classifiers given few training examples and for evaluating images for search in an image database. In particular, we discuss a knowledge reuse framework in which a *supra-classifier* improves the performance of the *target classifier* using information from existing *support classifiers*. Soft computing methods can be used for all three types of classifiers involved. We explore supra-classifier design issues and introduce several types of supra-classifiers, comparing their relative strengths and weaknesses. Empirical examples on real world image data sets are used to demonstrate the effectiveness of the supra-classifier framework for classification and retrieval/search in image databases.

Keywords: knowledge reuse, image classification, image database, curse of dimensionality, soft classifiers

1 Introduction

1.1 *A Priori* Knowledge for Image Classification

The development of computer vision systems that can perform as well as humans has proven to be an extremely difficult task. One of the reasons often cited for this is the difficulty in giving artificial vision systems enough domain knowledge to handle the complexity of real-world image understanding tasks. One of the most important image understanding problems that suffers from this drawback is image classification, i.e. the task of building a system that can distinguish one category of images from another. As an example, consider the problem of distinguishing images of males from images of females. Considerations of physiology, customs of clothing design, grooming habits, and other less tangible concepts are often leveraged by humans making such a decision. Understanding what relevant knowledge is available and how it can be included in an image classification system is still an open problem.

Exemplar based inductive image classifiers try to generalize from a given training set. They can utilize two types of knowledge sources: raw image data and *a priori* knowledge about the image data set. The raw image data may consist of an array of pixel intensity values (for grayscale images) or color intensity information (for color images). Images represented in this or a similar fashion potentially contain a very large amount of information ("A picture is worth a thousand words"), but this information is difficult to handle without other a priori knowledge. Generally, using pixel value information directly for image classification is extremely difficult, if not impossible due to the extremely high dimensional input space.

A priori knowledge about the image data set is simply information about the data set that is external to the data itself. This information can be used in several capacities in the construction of an image classifier. One of the most important uses of a priori knowledge is for feature extraction. For example, the knowledge that a set of images are from indoor office scenes might influence to what degree edges would be considered germane features since they tend to occur more commonly in man-made objects. A priori knowledge can also be used to choose image sets or learner architectures, or even modify the learning process itself. For example, Bayesian approaches to image classification (e.g. [21]) use a priori knowledge in the form of prior class probabilities and prior distributions assumed for the model parameters. Some classifier architectures use the structure and value of model parameters to represent a priori knowledge (e.g. the discriminant function in statistical classifiers [12], size and order of features in decision trees [23], and the type and number of hidden units, amount and form of regularization in feed-forward neural networks [13]). Such approaches can work very well if the resulting inductive bias matches the problem very closely. However, in practice it may be quite difficult to use this type of knowledge to select and tune a proper model. Also, standard assumptions used (independence among variables, Gaussian distributions, etc.) to make the problem tractable often result in a loss of accuracy [16,21].

1.2 Classifier Knowledge Reuse

Besides training data and a priori knowledge, previously constructed classifiers or labelers of images are a third type of knowledge source to consider for image classifier construction. This source is essentially a product of the first two. Image labels may have been present when the image database was originally created, or subsequently determined during later classification tasks. Alternatively, labels may result from a partitioning of the input image space induced by a different data set/categorization combination. In all three cases, the labels contain knowledge derived from both previous image data sets and a priori information. If this knowledge is relevant to the current classification task, then it can be used to built a better classifier. In Figure 1 salient features (e.g. size, color, shape) have been extracted from some unknown and

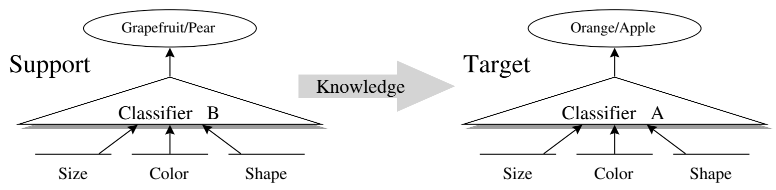

Fig. 1. Knowledge transfer between related tasks.

perhaps currently unavailable image data set, and an artificial classifier has been built to discriminate images of grapefruit from images of pears. We refer to a previously constructed classifier as a *support classifier*. We wish to construct a new *target classifier* to discriminate images of oranges from images of apples. Given that the same features are available, we can present images of apples and oranges to the grapefruit/pear classifier and observe its resulting behavior. In this case, since apples have similarity to pears and oranges have similarity to grapefruit, we expect that the grapefruit/pear classifier should be able to provide some indication as to whether we are showing it a image of an apple or an orange.

1.3 Characteristics of Classifier Knowledge Reuse

Classifier Knowledge Reuse is the idea that knowledge embedded in a previously created set of classifiers can be used to build a new classifier that performs better than one which simply uses its current training data and any available a priori knowledge for the current task. This is most effective when there is insufficient information in the current training set and a priori sources, and thus knowledge from classifier reuse can supplement existing knowledge. For example, if there are too few or noisy training images, then statistics over this training set may be difficult to estimate, resulting in poor learning. If there is too little a priori information, then the feature space for artificial learners may become too noisy or too large (high dimensional) to be searched effectively.

Besides assisting in new classification problems, classifier knowledge reuse has another, more interesting application. In traditional image classification problems, the goal is to build classifiers that generalize well to new, unseen images the classifier may encounter. Thus, the classification task is static but the image set of interest is dynamic. Consider the converse to this; a static image set and a changing set of classification tasks corresponding to newer uses of or studies on the same data. The goal in this application is to be able to create a knowledge base for future understanding and search in a fixed set of images.

As an example, consider the Mars Pathfinder images gathered recently by NASA. At the end of the mission, the set of available images does not grow or change. However, as science progresses and further analyses are performed, knowledge about this set in the form of classifications of the images may increase over time. The set of previous classifications can function as a "knowledge profile" about each of the images. When a researcher wants to find all of the images that fit a particular profile, he/she could manually classify some of the images as positive and negative examples of the concept being searched for. Knowledge from the previous classifications could be used to build a new classifier that can make decisions on the image set and retrieve the images that have been classified as positive examples. If the image set is static, then the previous classifiers no longer need to be available, as only the class labels that they generated are important. Thus, humans, automated classifiers with a limited lifespan (e.g. the Pathfinder probe), and other temporary types of classifiers can be used. If the image set is not static and the previous classifiers are still available, then new images can also be searched. An example of this type of knowledge reuse is presented later in this chapter.

2 Methods of Classifier Knowledge Reuse

We first briefly survey some existing research into architecture specific classifier knowledge reuse. Most of this work has focused on knowledge transfer mechanisms that use multilayer perceptron (MLP) neural network classifiers and has explored two main mechanisms; (i) knowledge re-representation and (ii) sharing internal state information. The benefits and limitations of these existing approaches are discussed, and then a broader framework for general classifier reuse is introduced.

2.1 Knowledge Re-representation

In the context of knowledge reuse, knowledge re-representation is the concept that knowledge about a classification task is extracted from a classifier in some new representation that is suitable for insertion into later classifiers. There has been much work on knowledge intensive learning focusing on symbolic rules extracted from and used in the creation of neural classifiers (e.g. [11,14,22,31,34]). If knowledge can be represented as rules, then these rules may be inserted into other neural networks. Often, these rules are used to initialize or adjust the structure or weights in multilayer perceptron neural networks. These approaches have resulted in better performing classifiers and/or classifiers which can be trained more quickly. However, these rule extraction approaches cannot reuse easily from non-MLP classifiers and have not demonstrated scalability to the cases where the number of relevant rules is very large.

Although less popular, there have been other re-representation schemes investigated. In [32], a neural network is used to recognize previously learned concepts in order to estimate the probability of an old class being presented as input when training a new classifier. The explanation-based neural network algorithm (EBNN) is used to train the classifier for the current classification task by using the target function derivative information (the re-represented knowledge) to augment the learning process. In another, unrelated work [33], a scaling vector for a nearest neighbor classifier is learned for one classification task and reused for another, related task.

2.2 Internal State Sharing

Instead of re-representing knowledge, some knowledge reuse research has focused on reusing internal state information, namely weight values in MLP style neural networks. Under the belief that related classification tasks may benefit from common internal features, Caruana [6] has created an MLP based multiple classifier system that is trained *simultaneously* to perform several related classification tasks. In this work on two layer neural networks, the first layer is shared by several related classification tasks. The premise is that related tasks have similar connectionist representations in the weight space, and that by training on more and a wider variety of samples (because there are multiple training sets), these representations can be better learned. The second layer of this neural network is separated and independent for each classification task. Improved classification performance has been demonstrated in some cases. Baxter [2] has developed a rigorous analysis of a similar type of architecture, showing that as the number of simultaneously trained tasks increases, the number of examples needed per task for good generalization decreases. These knowledge sharing methods are not knowledge reuse by our previous definition since all of the classification tasks must be created simultaneously, but share many of its qualities. More closely matching the knowledge reuse definition is work by Pratt [25], in which some of the trained weights from one MLP network trained for a single task are used to initialize weights in an MLP to be trained for a later, related task. Improved training speed has been shown for this reuse method.

2.3 Supra-Classifier Knowledge Reuse

We now describe a general framework for classifier knowledge reuse recently introduced in [3,4]. The *Supra-classifier* knowledge reuse framework is a simple two layer structure which allows the reuse of knowledge from any type and quantity of previously created classifiers. These classifiers ultimately share the same input domain as the new classification task of interest, although they may operate on different features extracted from the images. The supra-classifier knowledge reuse process is to present the images to all available previously trained classifiers and then use the resulting output vector of classification labels as the input for a second stage supra-classifier[1]. This supra-classifier then makes the final classification decision for the current target classification task ($\tilde{c}_\tau(\cdot)$ in Figure 2.) Previously trained classifiers

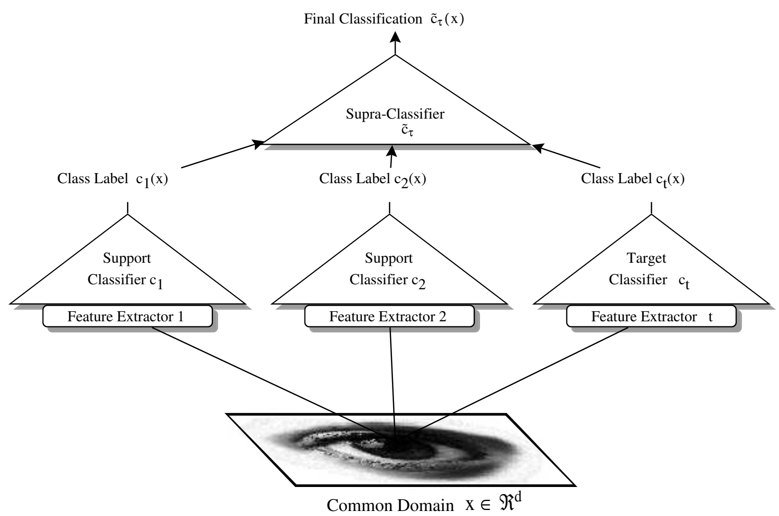

Fig. 2. A *Supra-Classifier* based knowledge reuse framework.

are termed *support* classifiers. Support classifiers are generally (but not always) designed for tasks other than the current *target* classification task of interest. In Figure 2, two of the three support classifiers are for different tasks, and one has been constructed for the current classification task of interest using only the training set.

While Figure 2 may appear to bear a superficial resemblance to recent popular approaches such as stacking [35], committees, ensembles [15,17,27], and mixtures of experts[19,20,26], the supra-classifier is fundamentally different from these "combiner" approaches. The supra-classifier is a generalization on combining where the support classifier could be designed for different tasks, and are immutable, having been trained previously. Support classifiers for ensembles/combiners try to solve the *same* classification task (though they may differentiated by input regions or feature selection) and are not previously created classifiers. Techniques like combining and stacking are simply good methods of decomposing a classification task into simpler tasks and generally do not reuse previous knowledge.

There is a simple probabilistic intuition to explain why the supra-classifier can effectively reuse knowledge from previously constructed relevant classi-

[1] This restriction allows one to use any type of support classifier since internal information is not needed.

fiers. Suppose that each image for a new classification task is represented as a point in a two-dimensional feature space. Let there be two target classes of images, X and O, in this space, and let a distribution of image samples be represented in Figure 3. Suppose there is a previously trained (support)

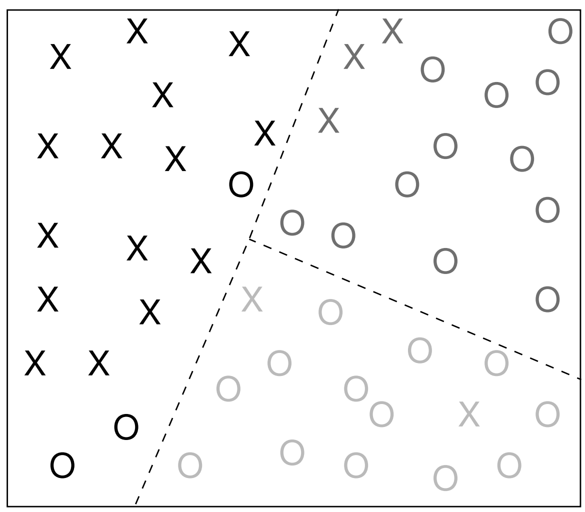

Fig. 3. Knowing the support classifier labels (indicated by the grey levels) helps to guess the target class.

classifier that divides the feature space into three regions; black, dark gray, and light gray. In Figure 3 these regions are separated by dotted lines and the X and O points in these regions are colored appropriately. In the example here, knowing that the support classifier label is black for a particular image gives a good indication that the target class for that image is probably X. Thus, knowing the support classifier label has helped guess the target class label correctly with greater probability. A formal treatment of this result is presented in [5].

3 Supra-Classifier Design

The supra-classifier design process is dependent on the specifics of the training image set and support classifiers since it should be able to use both of these knowledge sources effectively to maximize classification accuracy. We now discuss the criteria of size of the training set and number of support classifiers to guide the construction of the supra-classifier and compare different supra-classifier approaches in the context of these criteria.

3.1 Space of Knowledge Sources

Supra-classifiers make classification decisions on a vector of categorical (class label) values. Just like any normal classifier, they use the target training samples (and a priori information if available) to make a decision. In a normal classifier, typically the feature set is static, and to improve classification performance more and/or better training samples are needed. In contrast to this, the premise of the supra-classifier framework is that knowledge can also be added by increasing the number of relevant support classifiers (input features). Thus, although the design of a supra-classifier is closely tied to that of making a normal classifier with a discrete input space, there is the additional design goal of being able to perform better when more features are available, especially in the scenario of a static training set size. Consider the "knowledge space" of Figure 4. Points in this space qualitatively repre-

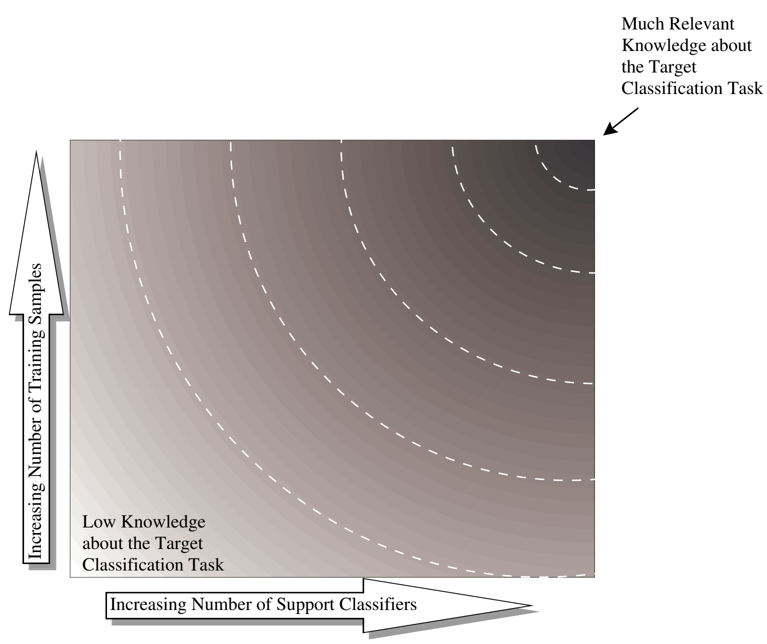

Fig. 4. Hypothetical space of available knowledge about the target classification task.

sent the amount of knowledge that is available in a target training sample set to a supra-classifier. With more "good" samples or support classifiers, the amount of knowledge increases. Goodness depends on certain desirable conditions such as independence and random sampling. The hypothetical greyscale shown has contours where the knowledge relevant to the target classification task (as reflected by the ideal achievable error rate) is equal in quantity. The supra-classifier designer must know where in this knowledge space he/she is working, and choose a supra-classifier that functions well in that part of the space.

We now set up the mathematical framework for supra-classifiers, describe several potential supra-classifier architectures, and discuss where in the above knowledge space they are the most appropriate. We also discuss some techniques that expand the region of usefulness in the knowledge space for some of these classifiers.

3.2 Definitions

Let the target classification task be τ, and let τ have discrete range $\mathcal{S}_\tau$ and d dimensional input domain space $\Re^d$. Let $\{x, y\}_\tau : x \in \Re^d, y \in \mathcal{S}_\tau$ be the set of training examples for task τ. We assume that $\{x, y\}_\tau$ is a sample set from the true distribution for task τ having associated random variable $(X_\tau, Y_\tau) \in (\Re^d, \mathcal{S}_\tau)$. Our goal is to find the most likely value of the conditional marginal $Y_\tau|(X_\tau = x)$ and define this maximum likelihood function to be $t(x) = \arg\max_y P(Y_\tau = y|X_\tau = x)$. Thus, $t(\cdot) : t(\cdot) \in \mathcal{S}_\tau$ is the target function that we would like to approximate using the information in $\{x, y\}_\tau$. Let $\mathcal{B}$ be a set of support classification tasks which have the same input domain space $\Re^d$ as task τ. Let $\{c_b(\cdot)\} : b \in \mathcal{B}$ be the corresponding set of classifiers where each $c_b(\cdot)$ maps $\Re^d \mapsto \mathcal{S}_b : b \in \mathcal{B}$.[2] Let $\hat{X}_\tau$ be the random variable associated with the input values of training sample set $\{x, y\}_\tau$. Let $T_\tau : T_\tau = t_\tau(\hat{X}_\tau)$ be defined as the random variable associated with the target function of $\hat{X}_\tau$. Similarly, let $C_b : C_b = c_b(\hat{X}_\tau)$ be the random variables resulting from the application of $\hat{X}_\tau$ to the support classifiers.

3.3 Probability Estimate Based Supra-Classifiers

One of the most common and compelling approaches to constructing a supra-classifier is to perform probability estimates on the discrete feature space of support classifier labels. An ideal probability based supra-classifier $c^*_\tau(x)$ will always choose the most likely class of the $y \in \mathcal{S}_\tau$ given the class labels $\{c_b(x)\} : b \in \mathcal{B}$ (maximum posterior probability). More specifically, for any given set of values $\{z_b : z_b \in \mathcal{S}_b\} : b \in \mathcal{B}$ we can define the maximum probability function $m_\tau(\cdot)$ as

$$m_\tau(\{z_b : z_b \in \mathcal{S}_b\} : b \in \mathcal{B}) = \arg\max_y P(T_\tau = y|\{C_b = z_b\} : b \in \mathcal{B}). \quad (1)$$

We can then define an ideal classifier based on this maximum probability function as

$$c^*_\tau(x) = m_\tau(\{c_b(x) : c_b(x) \in \mathcal{S}_b\} : b \in \mathcal{B}). \quad (2)$$

[2] Although some of the support classifiers may have been trained for task τ directly, in general $b \neq \tau$ and $\mathcal{S}_\tau \neq \mathcal{S}_b$, as the tasks are different.

where $c^*_\tau(\cdot)$ has associated random variable $C^*_\tau : C^*_\tau = c^*_\tau(\hat{X}_\tau)$. If the number of support classifiers is small and the number of target training samples is large (the upper left corner of Figure 3), then this ideal supra-classifier can be built by estimating probabilities directly from the sample probabilities. However, if the number of support classifiers is quite large, Equation 2 is not directly scalable due to the *curse of dimensionality* [10]. One aspect of this "curse" is the fact that in order to maintain a constant confidence in sample based probability estimates as the dimensionality (number of support classifiers) goes up, one must have an exponentially increasing number of samples.

Thus, in practice, approximating approaches to Equation 2 are usually required. Most of these approximations use assumptions on and/or a priori knowledge about the structure of dependencies between support classifiers. Adding knowledge in this manner can reduce the dimensionality of the probabilities to be estimated. Examples of this include belief networks[24] and log-linear modeling[7]. While this structuring generally requires a priori knowledge specific to a particular classification task, some common assumptions such as independence among support classifier labels conditional on the target classification task are often made.

The Naive Bayes Classifier. If the independence assumption truly holds, then probability based supra-classifier types restricted to the upper left corner of Figure 4 can move rightward (toward more support classifiers) and downward (toward less samples) to some degree without compromising classification performance. The best known (and possibly simplest) classifier that takes advantage of this is the *Naive Bayes* classifier[23]. Bayes rule states that

$$P(T_\tau | \{C_b\} : b \in \mathcal{B}) = \frac{P(\{C_b\} : b \in \mathcal{B} | T_\tau) P(T_\tau)}{P(\{C_b\} : b \in \mathcal{B})}. \tag{3}$$

Given the conditional independence assumption, the conditional term of the numerator in Equation 3 can be calculated as

$$P(\{C_b\} : b \in \mathcal{B} | T_\tau) = \prod_{b \in \mathcal{B}} P(C_b | T_\tau).$$

The term $P(T_\tau)$ can be assumed constant (equal priors), estimated from the samples, or estimated from a priori information. The denominator of Equation 3 can be assumed to be constant (equal prior support class probabilities), but is often calculated as

$$P(\{C_b\} : b \in \mathcal{B}) = \sum_{y \in \mathcal{S}_\tau} P(T_\tau = y) \prod_{b \in \mathcal{B}} P(C_b | T_\tau = y),$$

where the conditional independence assumption is made once again. The probabilities in the RHS of Equation 3 can be estimated from the training samples and be substituted into Equation 1. The ideal classifier of Equation 2 can then be calculated from these estimates. This supra-classifier is equivalent to the ideal classifier if the conditional independence assumption holds.

Bayes classifier with Feature Selection. If there are many training samples and few support classifiers, then Equation 2 can be estimated directly. However, if there are many support classifiers, but most of them are irrelevant for to the target classification task, then a process of *feature selection* can be used to eliminate the less useful features. This corresponds to moving the target classification task leftward in Figure 4, making direct probability estimates easier. Feature selection requires making a judgement on which subset of the support classifiers of a given size is optimal for supra-classifier accuracy. In general, this is a well studied problem, and finding the best feature selection method often depends on the target classification task.

Bayes Classifier with Smoothing. Rather than assuming independence or hoping that most of the support classifiers are irrelevant and can be excluded, it is also possible to use *smoothing* to make better probability estimates of the target classes conditional on the support class labels. The kernel method for probability smoothing introduced in [1] allows estimation of the joint target and support classifier label probabilities $P(T_\tau = y, \{C_b = x_b\} : b \in \mathcal{B})$, which is proportional to $P(T_\tau = y | \{C_b = x_b\} : b \in \mathcal{B})$ (the conditional target class probabilities). Suppose there are $|\mathcal{B}|$ support classifiers and n images as training samples. The kernel smoothing function can be written as:

$$P(T_\tau = y, \{C_b = x_b\} : b \in \mathcal{B}) = \frac{1}{n} \sum_{i=1}^{n} \prod_{b=1}^{|\mathcal{B}|} K(i, b, \lambda), \tag{4}$$

where $K(i, b, \lambda)$ is a kernel function and λ is a smoothing factor. The sum is over all the set of training images $x_s : s = 1 \dots n$ and the product is over all $|\mathcal{B}|$ support classifiers. The kernel for a test support classifier label vector x_{test} is defined as:

$$\begin{aligned} K(i, b, \lambda) &= \lambda, \quad c_b(x_s) = c_b(x_{test}) \\ &= \tfrac{1-\lambda}{|\mathcal{S}_b|-1}, \; c_b(x_s) \neq c_b(x_{test}) \end{aligned} \tag{5}$$

where x_s is the sth training image and $|\mathcal{S}_b|$ is the number of different class labels for support classifier b. λ is defined only on $\max_b \frac{1}{|\mathcal{S}_b|} \leq \lambda \leq 1$. The case of $\lambda = 1$ means there is no smoothing, and with a large number of support classifiers (lower right corner of Figure 4), would mean that most of the probability estimates would almost certainly be zero. Since we are interested in the left hand side of

$$P(T_\tau = y | \{C_b = x_b\} : b \in \mathcal{B}) = \frac{P(T_\tau = y, \{C_b = x_b\} : b \in \mathcal{B})}{P(\{C_b = x_b\} : b \in \mathcal{B})}, \tag{6}$$

and the denominator of the right hand side of Equation 6 is constant for a given image, if we simply calculate the numerator of Equation 6 for all possible target values and take the largest, we are performing a direct estimation of the ideal probabilistic supra-classifier Equation in 2.

3.4 Combiner Based Supra-Classifiers

Combiner or ensemble based classifiers are systems which use the classification decisions of many simultaneous *target* classifiers and "combine" their decisions into a final decision. Combiners have been extensively researched (see [29] for a survey), and we so we only introduce a simple application to the supra-classifier framework here. Consider the ideal classifier of Equation 2 constructed using only a single support classifier b and call this a "voting" classifier $c_b^{vote}(x)$ as defined by

$$c_b^{vote}(x) = m_\tau(\{c_b(x)\} : b \in \mathcal{B}), \tag{7}$$

where τ is the target task. The voting classifier makes a guess at the most likely target class based only on the information from one support classifier; in essence this is the support classifier's "vote" for the correct target class. The supra-classifier consists of tallying all $|\mathcal{B}|$ votes and choosing the target class with the most votes. While we avoid the problems of making high dimensional probability estimates, this supra-classifier is sensitive to noisy voters. If a few "good voters" make correct choices most of the time, they may be overwhelmed by those voters which are essentially guessing randomly, or always choosing the target class with the highest prior probability. Thus, it may be desirable to weight voters by their accuracy to favor the better voters. Some weighting schemes are discussed in section 3.8.

3.5 Tree Based Supra-Classifiers

In a traditional decision tree classifier, the strategy is to divide an input space into "hyper-rectangular" target class regions of high class "purity". Branching decisions are based on how much each input feature increases the class "purity" of examples in resulting subregions. The supra-classifier framework is a very intuitive application of tree based classifiers in that it shares the goal of creating target class regions of high class purity, but the regions are not simple "hyper-rectangles" as would be found in a real valued input space. Instead, target class regions are defined by how the support classifiers partition the image feature input space.

Consider that any single support classifier $c_b(\cdot)$ partitions the input space $\mathcal{R}^d$ into regions labeled for the classes of classification task b. Recall from the above discussion on the intuition of the supra-classifier architecture that if $c_b(\cdot)$ has (even a small amount of) relevant knowledge to contribute to the target classification task and a good relevance measure is chosen, then it should partition the input space into subregions of greater purity (of the target classes) than would a random partitioning.

Consider the extension to the set of $|\mathcal{B}|$ support classifiers that define a set of $|\mathcal{B}|$ overlapping partitions of the input space. The overall result is a partitioning of the input space consisting of $|\mathcal{B}|$-way "intersection regions". It is easy to see that as $|\mathcal{B}|$ increases, the average size of these intersection regions will decrease as their number increases. If each of the $|\mathcal{B}|$ support classifiers has contributed some amount of unique knowledge, then the premise of tree classifiers is that the average class purity of the intersection regions will also increase. A hypothetical example can be seen in Figure 5 where a two di-

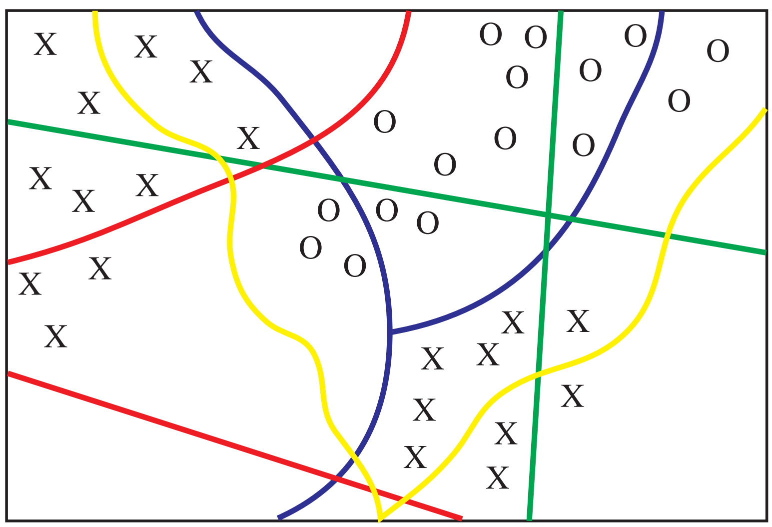

Fig. 5. A hypothetical 2-D feature space that has been partitioned by 4 different support classifiers, identified by the different grayscales of the partition boundaries.

mensional feature space of two target classes "X" and "O" (similar to figure 3) is partitioned by different four support classifiers, the boundaries of which are represented by the four line grey levels. Here it can be seen that class purity of each intersection region us higher when more support classifiers are considered.

A difficulty with decision trees is that at each additional partitioning, the number of image samples in each region intersection tends to drop. Thus, after only a relatively few features, there may no longer be sufficient samples to make probability estimated upon which further branching is based. Most practical decision tree classifiers order the branching by decreasing relevance, conditional on the branches already taken. A commonly used relevance measure is mutual information of each support classifier with the target. Previous work [5] has given empirical evidence that this relevance measure is also effective in a supra-classifier framework.

3.6 Similarity Based Supra-Classifiers

Probability based supra-classifiers tend to have the same strengths and weaknesses. If the set of support classifiers for the target classification task is large and cannot be reduced by feature selection or compensated for by independence assumptions or smoothing, then it may not be appropriate to use these techniques. Instead, it may be better to use *similarity based* supra-classifier techniques instead. These techniques consist of defining distance measures between images in the space of support classifier labels, and then making a final target classification based on these distances. Ideally, images that are similar would have a small distance between them, while dissimilar images would be far from each other. For the purposes of classification, an optimal distance measure $d(x, y)$ between two images, x and y, for target classification function $t(\cdot)$ would have the property:

$$d(x, y|t(x) \neq t(y)) > d(x, y|t(x) = t(y)). \tag{8}$$

for all value pairs of x and y in the set of images. Equation 8 states that if two images are of the same target class, the distance between them will always be less than if they are of are differing target classes. The challenge then, is to find a good distance measure and build an appropriate supra-classifier to achieve satisfaction of Equation 8.

Hamming Nearest Neighbor Supra-Classifier. The Hamming Nearest Neighbor (HNN) is a simple classifier for discrete features (e.g. support classifier labels) similar to a traditional nearest neighbor which operates in a Euclidean space. If $I(\cdot)$ is the indicator function, then the (Hamming) distance between two samples x_{train} and x_{test} can be calculated as

$$D_{|\{\mathcal{B}\}|}(x_{train}, x_{test}) = \sum_{b:b=1\ldots|\{\mathcal{B}\}|} I(c_b(x_{train}) \neq c_b(x_{test})).$$

For each test sample, the Hamming Nearest Neighbor (HNN) supra-classifier will choose the class label of the training sample with the smallest Hamming distance from it. There is no need to estimate probabilities as in the probability based supra-classifiers.

Recent analysis gives indication that the Hamming distance as used in the HNN classifier approaches the optimal distance measure [4,5]. One result of this analysis can be summarized in the following theorem:

Theorem:
If the support classifiers $\{c_b(\cdot)\} : b = 1 \ldots |\{\mathcal{B}\}|$ are independent of each

other conditionally on the target class $t(\cdot)$, we are given three images x_α, x_β, and x_γ, chosen randomly and independently from some distribution, $t(x_\alpha) = t(x_\gamma) \neq t(x_\beta)$, and the priors for the target classes are equal, then

$$\lim_{|\{\mathcal{B}\}| \to \infty} P(D_{|\{\mathcal{B}\}|}(x_\beta, x_\gamma) > D_{|\{\mathcal{B}\}|}(x_\alpha, x_\gamma)) = 1.$$

Proof of this theorem is described in [5]. This theorem states that in the limit as more relevant, independent support classifiers become available, the probability that the Hamming distance between a training and test sample of different target classes will be greater than the distance between the test sample and a training sample of the same class approaches 1. It should also be noted that this theorem holds even if there is only one training sample of each target class and even if all of the support classifiers are only very weakly relevant to the target classification task. Noise from totally irrelevant classifiers will tend to cancel itself out as long as the independence assumption holds.

Thus, the HNN supra-classifiers is useful even in the extreme bottom right corner of the knowledge space described in Figure 4. Furthermore, a simple application of the Hoeffding inequality [18] is able to place an exponential upper bound on the convergence rate of the HNN supra-classifier as function of the average relevance of the support classifiers.

Despite this potentially powerful result, a few caveats are in order. First, the existence of an infinite number of independent, relevant support classifiers is only possible if the classification problem has zero Bayes error. Also, the HNN may not perform well if a few strong features can be selected for the target task (effectively described in the upper left corner of Figure 4), since it assumes a more uniform distribution of relevant knowledge among the support classifiers. It is possible to weigh the indicator functions in the Hamming distance by the relevance of the support classifiers to compensate for a non-uniform distribution of knowledge, thus moving its applicability leftward in the knowledge space.

3.7 Use of Other Supra-Classifier Types

The problem of building a supra-classifier is simply the problem of building a classifier that can effectively use (perhaps a large number of) categorical inputs features. Many classifiers, even if not specifically designed for categorical input, may be used if an appropriate representation for the support classifier labels is made. For example, the Multilayer Perceptron (MLP) and Radial Basis Function (RBF) classifiers expect their input to be a vector of real values, so a simple "1-of-M" encoding of the support classifier output labels can be used.

3.8 Relevance Measures and Their Uses

In many cases the support classifiers will vary widely in their usefulness in assisting a supra-classifier. Many of the supra-classifiers can benefit from (and some even require) knowing the relevance of the support classifiers to build a practical system. Relevance of a support classifier or set of support classifiers is a measure of its ability to improve the classification accuracy of a supra-classifier. In general, attempts to make relevance measures on sets of many support classifiers fall prey to the same curse of dimensionality that the ideal probabilistic supra-classifier does because many, if not all, of these measures depend on probability estimates. Thus, we will make independence assumptions as needed so that we may only consider relevance measures of single support classifiers. Depending on the specific type of measure, relevance can be used to weight support classifiers for purposes such as ranking of support classifiers for feature selection or weighting of terms in a Hamming distance for the HNN supra-classifier.

Mutual Information. The information theoretic measure mutual information is a measure of "shared knowledge" between two random variables. A standard definition of mutual information between random variables U and V in bits is

$$I(U,V) = \sum_{u,v} P(U=u, V=v) \log_2 \frac{P(U=u, V=v)}{P(U=u)P(V=v)}. \tag{9}$$

If $I(T_\tau, C_b) > I(T_A, C'_b)$ where $C_b = c_b(\hat{X}_\tau)$, $C_b = c_b(\hat{X}_\tau)$, and $T_\tau = t(\hat{X}_\tau)$, then we say that c_b "knows" more about t_τ than does c'_b. From Fano's inequality[9], we also know that in this case, an information theoretic upper bound on performance of a classifier built to perform classification task τ using only the information from c_b is higher than for one built using only the information from c'_b. As mentioned earlier, this is a commonly used relevance measure in decision tree construction, where the measure is made conditionally on all of the previously taken branches.

A Value Distance Metric. Stanfill and Waltz [30] introduced a Value Distance Metric (VDM) to measure the distance between two discretely valued vectors for instance based learning (IBL) methods, making is applicable to similarity based supra-classifiers such as the HNN. This metric considers the differences between the frequencies of each target class occurring over the target training set conditional on the value of each feature (support classifier label), summed over all of the support classifiers. Consider the labels $c_b(x)$ and $c_b(y)$ of a single support classifier b for two images x and y. In a supra-classifier framework, the VDM defines the distance between these two values to be:

$$d_b(c_b(x), c_b(y)) = \sum_{t=1}^{|\{\mathcal{S}_\tau\}|} |F(T = t|C_b = c_b(x)) - F(T = t|C_b = c_b(y))|^k . \quad (10)$$

Where the $F(\cdot)$ are the sample probabilities of each event over the image training set. Often, the constant $k = 1$ is used. Thus, for every support classifier with $|\{\mathcal{S}_\tau\}|$ possible labels, there is a $|\{\mathcal{S}_\tau\}| \times |\{\mathcal{S}_\tau\}|$ matrix of distance values $d_b(\cdot, \cdot)$. Stanfill and Waltz also included a weighting term w_f^g which made $d_b(\cdot, \cdot)$ asymetric. A priori knowledge is required to use this weight effectively, and its exclusion keeps $d_b(\cdot, \cdot)$ symmetric.

Using the above metric calculated for all of the support classifiers over the image training allows the total distance between two images to be:

$$D(c_b(x), c_b(y)) = \sum_{b=1}^{|\{\mathcal{B}\}|} w_x w_y d_b(c_b(x), c_b(y))^r \quad (11)$$

where w_x and w_y are weights on the images themselves and r determines a norm (e.g. $r = 2$ means Euclidean distance). In some IBL methods, these weights can be used to favor those images that help discriminate the target classes better. In [8] a modified VDM (MVDM) demonstrates empirically the usefulness of the VDM with the HNN classifier in an IBL context.

4 Experiments

In order to demonstrate knowledge reuse in the supra-classifier framework, we have chosen two classification tasks, one each for the supra-classifier framework's two major application areas. The first application is the enhancement of classification performance of a new classifier related to previously constructed classifiers. For this, a collection of binary classifiers of images of military vehicles is used to aid in the creation of a similar such classifier. Second, previous classification labeling of images in a database by a human user are used to predict current classifications of interest to that person on the same database. These predictions could then be used to recall specific images.

4.1 Target Recognition

The goal here is to build a classifier to discriminate between two classes of military vehicles which are labeled HMMWV and 2S1. The sources of knowledge available are a training set of second generation FLIR images of outdoor scenes containing these two types of vehicles and a collection of ten previously built vehicle discriminators. The images were segmented to extract only the immediate region around the vehicles and each such sample is then represented by 47 scalar features, including 23 Zernike moments, 7 standard moments, 6 normalized/central moments, and other assorted features such as average intensity, height, width, etc [28]. The training set for the new classifier consists of 20 examples of HMMWV and 75 examples of 2S1.

All of the support classifiers are multilayer perceptron (MLP) two-class neural networks that have been constructed to discriminate between the following pairs of vehicle types; “M35 and HMMWV”, “M35 and M60”, “M35 and ZSU”, “M35 and M730”, “HMMWV and M60”, “HMMWV and ZSU”, “HMMWV and M730”, “M60 and ZSU”, “M60 and M730”, and “ZSU and M730”. Figure 4.1 shows sample images from the five classes. The M35 is a truck, the M60, 2S1, and M730 are tanks, the HMMWV is a “hummer” transport, and the ZSU is a Soviet anti-aircraft launcher. Note that, while some of these discriminators include HMMWV as a class, none include the 2S1.

Fig. 6. Examples of preprocessed second generation FLIR images used for the target recognition problem: From top, left to right: 2S1, HMMWV, M35, M60 and M730.

Careful choosing and parameter hand tuning of a simple MLP classifier allows a classification rate of about 98.5% using all of the training examples as a knowledge source. As mentioned earlier, classifier knowledge reuse is most useful when there is a dearth of knowledge from the training examples. Thus, we created an experimental setup that purposefully held back some of the training examples to see if the knowledge from the previously constructed classifiers could compensate for the loss of training set information. The number of available training examples ranged from 4 to 32, evenly distributed among the two target classes. We trained three traditional, unaided target classifiers for comparison: an MLP, a traditional single nearest neighbor classifier, and a C4.5 decision tree.

Eleven support classifiers were available; the ten previous constructed classifiers and an unaided MLP target classifier, chosen because it was a good performer in informal testing. Several supra-classifiers types were constructed including C4.5, MLP, the combiner based (VOTE), naive Bayes (BAYES), and Hamming nearest neighbor (HNN) classifiers. The target class examples were randomly divided into training and test examples. The supra-classifiers and unaided classifiers were constructed using the training examples and tested on the rest. This was iterated for 500 trials for each quantity of training examples considered. Figure 4.1 shows the classification rate of the various

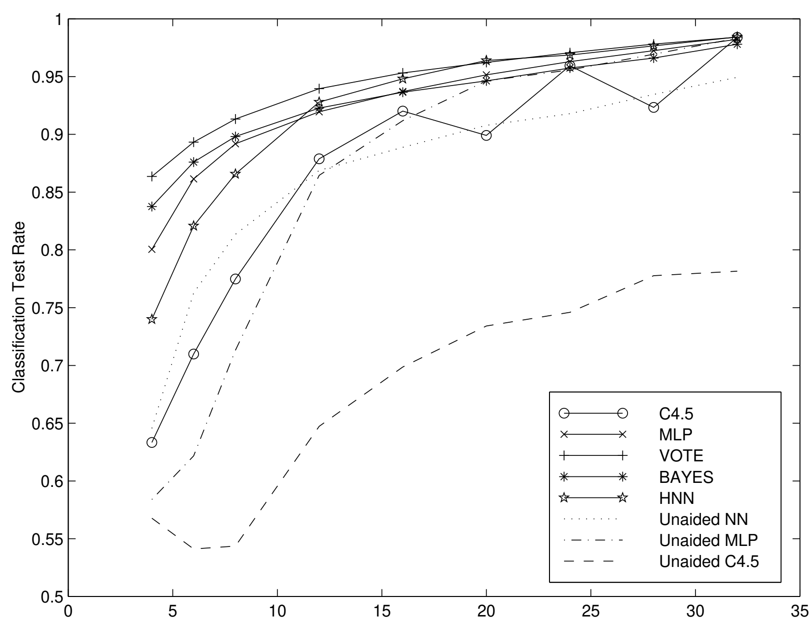

Fig. 7. Classification rate of several supra-classifiers and unaided classifiers versus the number of target training examples, for the target recognition problem.

supra and unaided classifiers on the test set versus the the number of target training examples available. For very few training examples, all but the C4.5 supra-classifiers provided a substantial performance improvement over all of the unaided classifiers with the combiner (VOTE) followed by the naive Bayes supra-classifiers demonstrating the highest overall performance. The unaided MLP classifier was a superior performer to the unaided nearest neighbor and C4.5 classifiers. As more training examples became available, unsurprisingly the benefit of knowledge reuse diminished, since there was more knowledge available from the training set. The results shown are statistically significant, but for neatness, the error bars are not shown. These results give evidence that when an inadequate target training set results in poor classification performance, knowledge reuse can help.

4.2 Building An Image Knowledge Base

Although the supra-classifier knowledge reuse framework can help in the construction of new, related classifiers, perhaps its best application is to the construction of a knowledge base for a (possibly fixed) set of images. Consider a fixed set of images such as works of art for which multiple classifications have been made by human experts, artificial classifiers, and other types of systems. For a particular image, the set of classification labels can act as a powerful description that can be used in understanding it. Suppose one knew the answer to dozens or perhaps even hundreds of categorizations of a photograph (e.g. indoor or outdoor, natural or man-made, whether it contains people, etc.). An "internal representation" as to what the photograph was about could be formed and perhaps one could even correctly answer novel questions about it, all without ever actually having seen the image. This novel desired classification could then be used to recall images from the database.

The supra-classifier knowledge reuse framework provides some of the tools to construct a system with such an ability. Given an image database where each image is annotated with a large number of classifications, a user could manually classify a few positive and negative examples of some novel concept of interest, which would become a training set for a supra-classifier. The major challenge of building a supra-classifier for a database of this sort is designing one that can effectively use a large number of support classifiers with only a small number of training samples. This case of a small number of high dimensional training samples corresponds to the lower right hand corner of the knowledge space described in Figure 4.

In order to demonstrate how a supra-classifier framework can be used as part of an image knowledge base system, a data set of 30 color images (primarily photographs) from the authors' personal collection and from a commercial CD-ROM was assembled. These images were chosen to be (subjectively) as diverse as possible, and some of them can be seen in Figure 4.2. We defined 71 sets of potential categorizations for these images that represented mutually exclusive concepts. Examples include "Big vs. Small", "Clean vs. Dirty", "Busy vs. Calm", and "Solid vs. Liquid vs. Gas". A web site (http://www.lans.ece.utexas.edu/cgibin/cgiwrap/kdb/top.pl) was created to present the 30 images to six human users, who were asked to classify the

Fig. 8. Nine of the 30 images in the data set.

images in each of the 71 ways. These classifications constitute a "personal profile" of knowledge about the images for each user, essentially making the users become their own "support classifiers". A 72nd classification was also made by each user for each image to act as a test "target" for novel classifications. For this target, the users were asked to decide whether they "liked" each image more than the "average" image in their judgement or not. A supra-classifier would use these target classifications as a training set and function as the judge of which images to recall based on whether the user would "like" each image. Although for demonstration purposes, the six users were asked to make a target classification for all of the 30 images, in a real system, hopefully only a few of such classifications should be needed if there is already enough information from a large number of support classifiers.

First Experiment - Number of Training Samples. The 30 photographs were randomly split into training set and test sets of varying sizes so that the dependency of supra-classifier performance on the number of training examples could be explored. The training set size ranged from 5 to 25 images with the rest held as test images. We then used several types of supra-classifiers to predict whether a subject would "like" each test image. These included a multilayer perceptron neural network (MLP), a C4.5 decision tree classifier

(C4.5), a Naive Bayes classifier (BAYES), the combiner (VOTE), a Hamming nearest neighbor (HNN) and a simple baseline classifier that always guessed the most common class in the training set (MCC). 500 trials (random splits of the image sets) for each training set size and for each of the six users were performed. The average classification test rate over all 500 trials for each type of supra-classifier versus the number of available training examples is shown in Figure 9. Here we can see that the HNN classifier performs better than other classifiers when there are very few training samples, with this margin of superiority waning as the number of training samples grows. As more training images became available, performance of all of supra-classifiers improved, although to varying degrees.

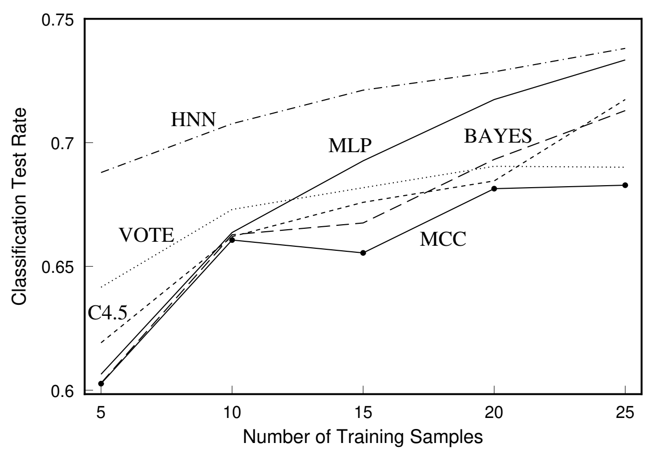

Fig. 9. Test rate versus number of training samples for each of the classifiers.

Second Experiment - Number of Support Classifiers. To study scenarios where only a few training examples but a very large number of support classifiers are available, we used a similar setup as in the first experiment but with the number of training samples held at five. The number of support classifiers was varied to measure supra-classifier scalability with increasing input dimensionality. 200 trials of random training/test set splits for each of several quantities of support classifiers ranging from 4 to 71 for one of the users was performed. The average classification test rate over the trials for the "like/don't like" labeling versus the number of available support classifiers is shown in Figure 10. Beyond 36 support classifiers, only the HNN supra-classifier continued to show improvement and most of the other supra-classifiers were not scalable beyond a small number of support classifiers. The naive Bayes classifier actually got worse, probably due to failed independence assumptions. All but the naive Bayes and MLP supra-classifiers

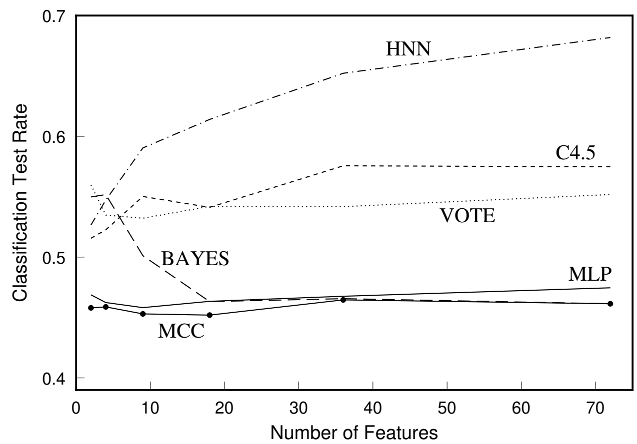

Fig. 10. Test rate versus number of input features for each of the classifiers.

however, had statistically significantly better performance than the baseline MCC classifier.

With a very large number of support classifiers, one would hope that by only classifying a small number of training images, other images of the desired class could be retrieved. This experiment is germane to such practical applications of supra-classifier knowledge reuse because even though most of the image classifications in this experiment were quite subjective in nature, a supra-classifier was able to use knowledge implicit in these subjective classifications to classify on a novel concept much better than random guessing.

5 Summary and Recommendations

The problem of insufficient domain knowledge poses a challenge in many image classification problems. Classifier knowledge reuse is discussed as a possible additional source of domain knowledge beyond traditional training set and a priori knowledge sources. It provides a more automated process for the inclusion of large amounts of high level domain knowledge that are implicit in existing classifiers. The supra-classifier framework is proposed as an approach to practical classifier knowledge reuse. Several issues of supra-classifier design and potential supra-classifier architectures are discussed, including the Hamming nearest neighbor classifier which demonstrates scalability to large amounts of classifier domain knowledge. Experiments showing two types of applications of supra-classifier knowledge reuse are presented. The first shows how to enhance a novel classifier's performance by reusing knowledge and the second examines how the supra-classifier framework can be used to estimate human subjective classifications of images in an image database of fixed composition. The experiments indicate that there is no single ideal supra-classifier architecture, although the Hamming nearest neighbor did demonstrate excellent performance in the traditionally difficult to handle case of high dimensionality and low training sample size. This motivates further investigation of the HNN architecture.

Currently, when novel image classifiers are built, most previous classifiers that may be relevant to the new task are ignored or are simply unavailable. The construction of a "database of image classifiers" would be a means for those who have built and those who need to build image classifiers to implicitly collaborate by using existing image classifiers from the database and contributing newly created ones.

Acknowledgments: This research was supported in part by ARO contracts DAAG55-98-1-0230 and 04-95-10494. We are thankful to Prof. Shishir Shah for providing images and data for the target recognition experiments.

References

1. Aitken, C. G. G. (1983). Kernel methods for the estimation of discrete distributions. *Journal of Statistical Computation and Simulation*, 16:189–200.
2. Baxter, J. (1994). *Learning Internal Representations.* PhD thesis, The Flinders University of South Australia.
3. Bollacker, K. D. and Ghosh, J. (1997). Knowledge reuse in multiple classifier systems. *Pattern Recognition Letters*, 18(11-13):1385–1390.
4. Bollacker, K. D. and Ghosh, J. (1998a). On the design of supra-classifiers for knowledge reuse. In *Proceedings of the 1998 International Joint Conference on Neural Networks*, pages 1404–1409.
5. Bollacker, K. D. and Ghosh, J. (1998b). A supra-classifier architecture for scalable knowledge reuse. University of Texas Technical Report UT-CVIS-TR-98-001.
6. Caruana, R. (1995). Learning many related tasks at the same time with backpropagation. In *Advances in Neural Information Processing Systems 7*, pages 657–664.
7. Christensen, R. (1997). *Log-Linear Models and Logistic Regression.* Springer, New York.
8. Cost, S. and Salzberg, S. (1993). A weighted nearest neighbor algorithm for learning with symbolic features. *Machine Learning*, 10:57–78.
9. Cover, T. M. and Thomas, J. A. (1991). *Elements of Information Theory.* John Wiley and Sons, Inc., New York.
10. Friedman, J. H. (1994). An overview of predictive learning and function approximation. In Cherkassky, V., Friedman, J. H., and Wechsler, H., editors, *From Statistics to Neural Networks, Proc. NATO/ASI Workshop*, pages 1–61. Springer Verlag.
11. Fu, L. M. (1993). Knowledge-based connectionism for revising domain theories. *IEEE Transactions on Systems, Man, and Cybernetics*, 23:173–182.
12. Fukunaga, K. (1990). *Introduction to Statistical Pattern Recognition.* Academic Press, San Diego, CA.
13. Ghosh, J. and Tumer, K. (1994). Structural adaptation and generalization in supervised feedforward networks. *Journal of Artificial Neural Networks*, 1(4):431–458.

14. Giles, C. L. and Omlin, C. W. (1994). Extraction and insertion of symbolic information in recurrent neural networks. In Honavar, V. and Uhr, L., editors, *Artificial Intelligence and Neural Networks: Steps toward Principled Integration*, pages 271–299. Academic Press.
15. Hashem, S. (1993). *Optimal Linear Combinations of Neural Networks*. PhD thesis, Purdue University.
16. Heckerman, D. (1997). Bayesian networks for data mining. *Data Mining and Knowledge Discovery*, 1(1):79–120.
17. Ho, T. K. (1992). *A Theory of Multiple Classifier Systems and its Application to Visual Word Recognition*. PhD thesis, State Universtiy of New York at Buffalo.
18. Hofri, M. (1995). *Analysis of Algorithms*. Oxford University Press, New York.
19. Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. (1991). Adaptive mixtures of local experts. *Neural Computation*, 3:78–88.
20. Jordan, M. I. and Jacobs, R. A. (1994). Hierarchical mixture of experts and the EM algorithm. *Neural Computation*, 7:181–214.
21. Mackay, D. J. C. (1995). Probable networks and plausible predictions - a review of practical Bayesian methods for supervised neural networks. *Network: Computation in Neural Systems*, 6(3):469–505.
22. Mahoney, J. J. and Mooney, R. J. (1993). Combining connectionist and symbolic learning to refine certainty factor rule bases. *Connection Science*, 5:339–364. Nos. 3 and 4.
23. Mitchell, T. M. (1997). *Machine Learning*. McGraw hill, New York.
24. Pearl, J. (1988). *Probabilistic Reasoning In Intelligent Systems*. Morgan Kaufmann Publishers.
25. Pratt, L. Y. (1994). Experiments on the transfer of knowledge between neural networks. In Hanson, S., Drastal, G., and Rivest, R., editors, *Computational Learning Theory and Natural Learning Systems, Constraints and Prospects*, chapter 19, pages 523–560. MIT Press.
26. Ramamurthi, V. and Ghosh, J. On the use of localized gating in mixtures of experts networks. In *(invited paper), SPIE Conf. on Applications and Science of Computational Intelligence*.
27. Rogova, G. (1994). Combining the results of several neural network classifiers. *Neural Networks*, 7(5):777–781.
28. Shah, S. K. (1998). *Probabilistic Multifeature/Multisensor Integration for Automatic Object Recognition*. PhD thesis, University of Texas at Austin, Austin, Texas.
29. Sharkey, A. (1996). On combining artificial neural networks. *Connection Science*, 8(3/4):299–314.
30. Stanfill, C. and Waltz, D. (1986). Toward memory-based reasoning. *Communications of the ACM*, 29:1213–1228.
31. Taha, I. and Ghosh, J. (1996). Three techniques for extracting rules from feedforward networks. In *Intelligent Engineering Systems Through Artificial Neural Networks ANNIE*, volume 6. ASME Press.
32. Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Advances in Neural Information Processing Systems 8*, pages 640–646.
33. Thrun, S. and O'Sullivan, J. (1996). Discovering structure in multiple learning tasks: The TC alogorithm. In *The 13th International Conference on Machine Learning*.
34. Towell, G. and Shavlik, J. (1994). Knowledge-based artificial neural networks. *Artificial Intelligence*, 70(1-2):119–165.

35. Wolpert, D. H. (1992). Stacked generalization. *Neural Networks*, 5:241–259.