Optimally Combining a Cascade of Classifiers

Kumar Chellapilla, Michael Shilman, Patrice Simard
Microsoft Research, One Microsoft Way, Redmond, WA, USA 98052
{kumarc, shilman, patrice}@microsoft.com

ABSTRACT

Conventional approaches to combining classifiers improve accuracy at the cost of increased processing. We propose a novel search based approach to automatically combine multiple classifiers in a cascade to obtain the desired tradeoff between classification speed and classification accuracy. The search procedure only updates the rejection thresholds (one for each constituent classier) in the cascade, consequently no new classifiers are added and no training is necessary. A branch-and-bound version of depth-first-search with efficient pruning is proposed for finding the optimal thresholds for the cascade. It produces optimal solutions under arbitrary user specified speed and accuracy constraints. The effectiveness of the approach is demonstrated on handwritten character recognition by finding a) the fastest possible combination given an upper bound on classification error, and also b) the most accurate combination given a lower bound on speed.

Keywords: Combining classifiers, optimal, branch-and-bound, cascade, classification speed, classification accuracy

1. INTRODUCTION

Optical character recognition (OCR) has become a common feature in software systems. This has been possible due to the recent growth in accuracy rates and decreasing computational costs needed to run sophisticated OCR algorithms. Today's mobile electronic devices such as cell phones and digital cameras are capable of acquiring images at a sufficiently high resolution (3 MPix) to facilitate OCR of text in these images. There is a simultaneous explosion of software applications targeting these devices that can read and translate words in documents, traffic signs, restaurant menus, travel guides, etc. The emergence of new technologies such as the World-Wide Web (WWW), digital libraries, video and camera-based OCR has further fueled the ubiquitous need for OCR. As a result, OCR programs are now expected to run on very diverse hardware platforms ranging from high-end servers and desktops to low end PDAs and cell phones.

High-end document servers can support large computational costs and expect very low error rates. In contrast, low end PDAs and cell phones may not be able to support much computational cost (owing to low power requirements) but can live with slightly higher error rates. Though speed is a bottle neck on these mobile devices, memory is more freely available as these devices simultaneously target multimedia applications.

Traditional design of OCR systems has relied on building custom recognizers for each of the above hardware scenarios. In this paper a cascade architecture for combining classifiers is presented and demonstrated to be versatile enough to simultaneously support a wide variety of hardware scenarios. The cascade switches from one mode to another by simply changing its rejection thresholds (a handful of constants). A branch-and-bound variant of depth-first-search (DFS*) with pruning is proposed for finding optimal rejection thresholds in the cascade. Optimality is defined in terms of the desired speed and accuracy.

Two specific scenarios are investigated. In the first scenario, the goal is to find the fastest cascade given a maximum tolerable error (error constraint). In the second scenario, the goal is to find the most accurate classifier given a minimum tolerable speed (speed constraint). Section 2 briefly reviews existing classifier combination approaches for improving accuracy. The new approach and the underlying optimization problem are presented in Section 3 along with the DFS* algorithm for building optimal cascades. Experimental results demonstrating the effectiveness of the new approach are presented in Section 4 and conclusions are offered in Section 5.

2. BACKGROUND

Combining classifiers for improved accuracy is a well studied problem [1-6]. The intuition for why such an approach can work lies in the observation that the classification errors produced by radically different classifiers have low correlation [3]. Diversity among the constituent classifiers (base classifiers) is commonly emphasized to ensure a low error correlation. Though the base classifiers are different they should also be comparable, i.e., their outputs should be represented such that a combining classifier can use them as inputs. Common approaches to generating such a variety of base classifiers include using different initializations, different parameter choices, different architectures, different error criteria, different training sets, and different feature sets [3].

Simple classifier fusion methods such as minimum, maximum, mean, median, and majority voting have recently been studied both theoretically [1] and empirically [3]. Rather than using such simple static rules, a combining classifier can be trained to takes the outputs from two or more classifiers as input and produce a combined output that better models the class posterior probabilities or likelihoods. Such approaches based on learning have greater potential for producing larger improvements in accuracy [2]. Successful applications of classifier combinations include combining fingerprint matches, face and voice recognition and document processing [3-7]. We emphasize that all of these approaches focus on accuracy and invariably result in slower classifiers. The combined classifier is 2-20 times slower with the number of errors dropping by as much as 18% - 63% [4-5].

A preliminary investigation into optimizing a cascade of classifiers for speed can be found in [8]. Three approximate algorithms based on steepest descent, dynamic programming, and simulated annealing showed that significant speedups are possible (without sacrificing the error rate) using such a cascade architecture [8]. In this paper, the investigation is continued with primary emphasis on finding the optimal cascade (comprising a set of base classifiers specified *a priori*) under a variety of error and cost constraints. The proposed search algorithm is very general and can find the optimal cascade given an arbitrary pair of error and speed constraints.

3. COMBINING CLASSIFIERS FOR SPEED

In this paper we use base classifiers that are convolutional neural network (NN) classifiers [7] due to their versatility and success for OCR. Though neural networks are used, the combination architecture and associated optimization approach can be applied to any set of classifiers that return labels and confidences. These networks have two layers of convolutional nodes followed by a hidden layer and an output layer. One such network achieved the best known error rate of 0.4% [7] for handwritten digit recognition on (MNIST). It had 5 nodes in the first conv. layer, 50 nodes in the second conv layer, 100 hidden nodes, and 10 output nodes (one for each digit 0-9) and can process about 250-300 chars/sec on a P4 3GHz machine. The network performs over half a million operations per classification. This network would take over 15 seconds per page for document recognition and is too slow.

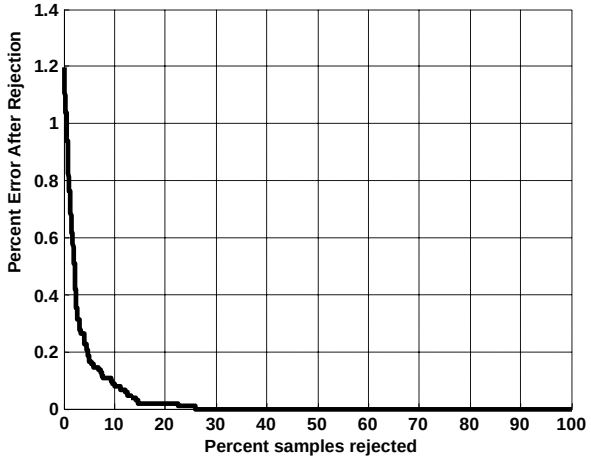

Figure 1. Rejection curve for a classifier.

The factors contributing to the computation in the classifier are a) the input resolution, b) convolutional layers, c) number of hidden nodes, and d) the number of output classes. The last parameter is defined by the problem and usually cannot be changed. For the remaining settings, smaller values produce faster but less accurate networks [7]. Further, it is well known that one observes a rate of diminishing returns in accuracy as complexity is increased.

The convolutional NN is trained using backpropagation and cross-entropy. It learns to predict class conditional probabilities [7]. The output for each class lies in [0,1] and can be used as a confidence measure to reject samples that would be incorrectly classified. Figure 1 presents the rejection curve for an example convolutional NN with 50 hidden nodes trained on MNIST. The rejection curve is monotonically decreasing indicating that the higher the confidence the less likely that the character will be misclassified. Even though the classifier only achieves an error rate of 1.2% on the MNIST data set, we can improve its error rate to 0.1% or even 0% by rejecting 9% or 26% of the data, respectively. This trade-off is the key intuition behind the proposed cascade architecture.

3.1 Cascade of classifiers

Figure 2 presents a cascade architecture for combining classifiers using a sequence of thresholds. Characters are processed by the cascade as follows: each input character image is initially presented to the first stage, S_1. If the classification output exceeds the first stage's threshold, t_1, then it is absorbed by the first stage and processing stops. If not, then the sample is rejected (by the first stage) and is passed on to the next stage. This process is repeated till the sample gets absorbed by some stage or we reach the last stage, S_M, in the cascade. The last stage, S_M, has a threshold of $t_M = 0$ and is designed to absorb all characters that reach it. The label assigned to the input character is that assigned by the absorbing stage. Given a set of thresholds, the expected cost of the cascade is a weighted sum of the constituent cascade costs with the weights being equal to the fraction of the samples absorbed. Note that when a sample gets absorbed at stage *i*, the cascade cost is the sum of the costs for the stages 1 through *i*.

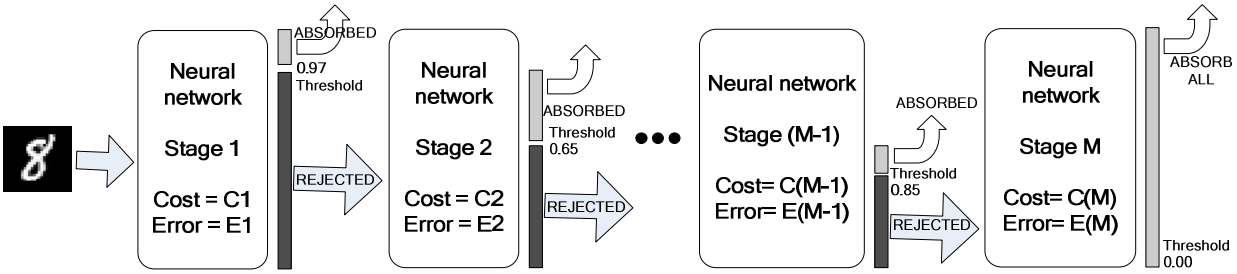

Figure 2. Cascade of classifiers. Samples rejected at each stage are passed on to subsequent stages. Classifiers at the front of the cascade are fast and inaccurate while classifiers towards the end of the cascade are more accurate but slower.

The cascade architecture has several merits for generating error-speed trade-offs. If faster less accurate nets are placed towards the front of the cascade and slower more accurate nets are placed towards the end of the cascade, one can dramatically reduce the expected processing times for input characters. Similarly, one can exploit the diversity in the classifiers to select the thresholds such that error rate decreases. The speedup and error rate of the cascade is determined by the costs, errors, and thresholds of each stage. The cost of each stage is inversely proportional to its speed. Lower cost implies a faster stage.

3.2 Constrained optimization problem

In this paper, we study such cascades with the stages arranged such that costs monotonically increase from left to right. Mathematically, $C_1 \leq C_2 \leq \ldots \leq C_M$ where M is the number of stages and C_i is the computational cost of the i-th stage. Given the ordering, without loss of generality, the costs can be normalized such that $C_1 = 1.0$ (as in Table 1). Unlike the costs, the errors are not expected to be monotonically decreasing as we move through the cascade (see Table 1). The search space of solutions is $S = \{t_1\}\times\{t_2\}\times\ldots\times\{t_M\}$, where $\{t_i\}$ is the set of all thresholds for stage i. Given this architecture, the goal is to find the optimal cascade given a pair of constraints ($e_{\max}$,$C_{\max}$):

a) *error constraint*: The cascade error rate must be below a maximum tolerable error rate, $e(T) \leq e_{\max}$

b) *speed constraint*: The cascade cost must be below a maximum cost, $C(T) \leq C_{\max}$

The optimal threshold vector under these constraints, $T_* = [t_{1*}, t_{2*}, \ldots, t_{M*}]$ is given by

$$T_* = \{T \mid T \in S, e(T) \leq e_{\max} \text{ and } C(T) \leq C_{\max}\} \quad (1)$$

$e(T_*)$ and $C(T_*)$ denote the optimum error and optimum cost, respectively and the speedup obtained is computed as $C_M/C(T_*)$.

Two specific versions (margin cases) of this general problem are of most interest:

a) maximizing speed:

$$T_* = \arg\min \{C(T) \mid T \in S, e(T) \leq e_{\max}\} \quad (2)$$

b) minimizing error:

$$T_* = \arg\min \{e(T) \mid T \in S, C(T) \leq C_{\max}\} \quad (3)$$

During optimization a set of labeled input samples, $\{x_i\}$, is used to evaluate the cost, $C(T)$, and error rate, $e(T)$, of the cascade for each candidate threshold vector, T. If a stage rejects all samples (i.e., absorbs no samples) then it is considered to be pruned from the cascade and adds no cost to the cascade. Note that this problem formulation allows for some of the networks to be completely dropped from the cascade. It is easy to see that $C(T_*)$ can be no lower than C_1. At the other extreme, the maximum possible expected cost per input sample is given by

$$\max C(T) = (NC_1 + (N-1)C_2 + \ldots + (N-M)C_M)/N \quad (4)$$

For $N >> M$, we get

$$\max C(T) = C_1 + C_2 + \ldots + C_M \quad (5)$$

3.3 Branch-and-bound variant of depth first search (DFS*)

A branch-and-bound variant of depth-first-search (referred to as DFS*) is used to exactly optimize rejection thresholds. Each node in the search is a partial configuration of thresholds. The start node corresponds to the state in which all thresholds are left unassigned; each of its child nodes corresponds to a particular setting of thresholds for the first classifier in the cascade. Each of their child nodes, in turn, corresponds to settings of thresholds for the second classifier in the cascade, and so on. The resulting search tree has a maximum depth of M with each leaf node corresponding to a unique solution (T) in the search space.

If at any point the algorithm cannot possibly satisfy the error or speed constraints, the sub-tree is pruned. Several pruning strategies are used to efficiently prune out large sections of the search space. The algorithm reaches a goal state when it achieves a fully assigned configuration. It terminates when it has searched, or safely pruned, the entire space. The pseudo code for DFS* is presented in Figure 3.

The DFS* algorithm is extended to include quantization to exactly optimize quantized rejection thresholds. In quantized DFS*, rather than attempting to split at every single example, we sort the values by confidence and split at every (N/Q)-th example, where N is the total number of examples and Q is the desired quanta. This percentile-based splitting distributes the data evenly amongst the quanta and provides a simple and natural quantization resolution. Quantization is key to ensuring that the optimized cascade generalizes well to unseen samples.

The size of the search space for this optimization problem is $O(Q^M)$, where M is number of stages in the cascade. To make the search scale to large numbers of examples or high quantization, we prune the search using safe heuristics.

To optimize error in the presence of a fixed cost, we have found empirically that a high-to-low cost search is effective. In other words, if we search through the high-cost, low-error stages at the beginning of the search, we are more likely to prune the search space based on cost constraints early on, and find an initial solution quickly. Once we have found an initial solution, the error-based pruning helps accelerate the search.

```
liveSet = { the set of all examples };
DFS*(0, 0, 0, liveSet);
DFS*(inError, inCost, stage, liveSet) {
        if(inError > _maxError || inCost > _minCost || stage > _stageMax) return;

        // first, try to absorb everything in this stage
        cost = inCost + Cost(stage, liveSet);
        error = inError + Error(stage, liveSet);
        if(error < _maxError  && cost < _maxCost) // goal state
                _maxCost = cost; // save thresholds

        // try to absorb some of the examples
        foreach(t in Thresholds(stage)) {
                subSet = Threshold(t, liveSet);
                DFS*(inError + Error(stage, subSet), cost, stage+1, liveSet-subSet);
        }

        // absorb none of the examples
        DFS*(inError, inCost, stage+1, liveSet);
}
```

Figure 3. Pseudo code for DFS* search to find the optimal solution. The Thresholds() function can enumerate either from low to high (e_{max} constraint) or high-to-low (C_{max} constraint).

3.4 Experiments

Experiments were conducted on handwritten digit recognition (MNIST) [7]. The MNIST dataset consists of 60,000 hand written digits uniformly distributed over 0–9. The dataset was split as follows: 80% of the samples were used for training, 10% were used for validation (to determine when to stop training) and the remaining 10% were used for testing. The validation samples were also used to optimize the thresholds using DFS*. The test samples are used to test the base classifiers and optimized cascades.

A total of 18 MNIST classifiers of varying sizes and resolutions were trained using backpropagation and cross-entropy [7]. The parameters being varied were the input image size (5x5, 7x7, 9x9, 11x11, and 29x29), the number of convolution layers (2 layers or 0 layers), and the number of hidden nodes (50 to 300). The speed ratio between the fastest and slowest classifiers is 41.6.

Table 1. MNIST Base Classifier Results: Costs and Errors

SN	Input	Architecture	Cost	Train%	Test%
1	5x5	0,0,50,10	1.00	26.52	27.48
2	5x5	0,0,100,10	1.22	25.06	26.81
3	5x5	0,0,200,10	1.72	25.36	26.76
4	5x5	0,0,300,10	2.24	25.57	27.07
5	7x7	0,0,50,10	1.04	8.82	9.06
6	7x7	0,0,100,10	1.35	8.17	8.38
7	7x7	0,0,200,10	1.99	8.29	8.82
8	7x7	0,0,300,10	2.61	8.04	8.36
9	9x9	0,0,50,10	1.14	4.49	4.65
10	9x9	0,0,100,10	1.53	4.13	4.20
11	9x9	0,0,200,10	2.36	4.03	4.01
12	9x9	0,0,300,10	3.17	3.78	3.89
13	11x11	0,0,50,10	1.26	3.63	3.72
14	11x11	0,0,300,10	3.84	2.35	2.27
15	29x29	0,0,50,10	3.39	3.43	3.36
16	29x29	0,0,300,10	16.3	1.82	1.81
17	29x29	5,50,50,10	23.0	1.20	1.25
18	29x29	5,50,300,10	41.6	1.10	1.19

Cascades were built using 8 trained MNIST networks. The base MNIST classifiers in the cascade were ordered from left-to-right as follows: 1, 13, 4, 15, 14, 16, 17, and 18. Quantized-DFS* was used to optimize the eight thresholds in the cascade using 5000 samples from the validation set. The number of quantization levels was varied in powers of two. Using powers-of-two quantization ensures that the threshold search spaces form exact subsets of each other. Quantization is key to avoiding over training (with small sample sets) and to ensure generalization. The generalization quality of the optimal solutions is tested on the samples from the test set.

Error constraint experiments were conducted with e_{max} = 1.2% (no extra error on test set) and e_{max} = 2.2% (double the training error of the best base classifier) while quantization was varied in powers of two. Cost constraint experiments

were conducted for C_{max} = 2.60, 5.20, 10.40, 20.81, and 41.62, corresponding to a speedup of 16x, 8x, 4x, 2x, and 1x. Note that the largest possible cost for the cascade is about 80.63 (from Eq. 5) which corresponds to about 0.52x.

4. RESULTS

Table 1 presents the base classifier training results for MNIST. The Train% is the error on the validation set (used for stopping training) and the Test% is the error on the test set. The best MNIST network (stage 18) had a validation error rate of 1.1% and a test error of 1.19%. On the other hand, the fastest network which was 41.6 times faster had an error rate of 26.52%!

Figure 4 presents results from error constraint experiments. Quantized DFS* was used to find the optimal thresholds for 8 MNIST classifiers (1, 13, 4, 15, 14, 16, 17, and 18) such that cascade error, $e(T) \leq e_{max}$ and the cascade cost, $C(T)$, is minimized. The training and test costs and errors for the optimal solutions found are shown as a function of the number of quantization levels. Figures 4(a) and 4(b) present the costs and errors for e_{max} = 1.2% while figures 4(c) and 4(d) present the costs and errors for e_{max} = 2.2%. As expected, the optimal cascade cost decreases monotonically as the number of quantized levels increases but quickly saturates. Note that a quantization Q = 32 or 64 achieves approximately the same speedup as the full approximation (equivalent to Q = 5000). The optimum cost for e_{max} = 2.2% is significantly lower than that for e_{max} = 1.2% showing that higher speedups can be obtained when a larger error can be tolerated. Large speed gains are observed. The optimized cascade for e_{max} = 1.2% is 10.4 times faster than the last stage in the cascade (with an error rate of 1.1%). For 2.2% the optimal speedup is over 20.8 times.

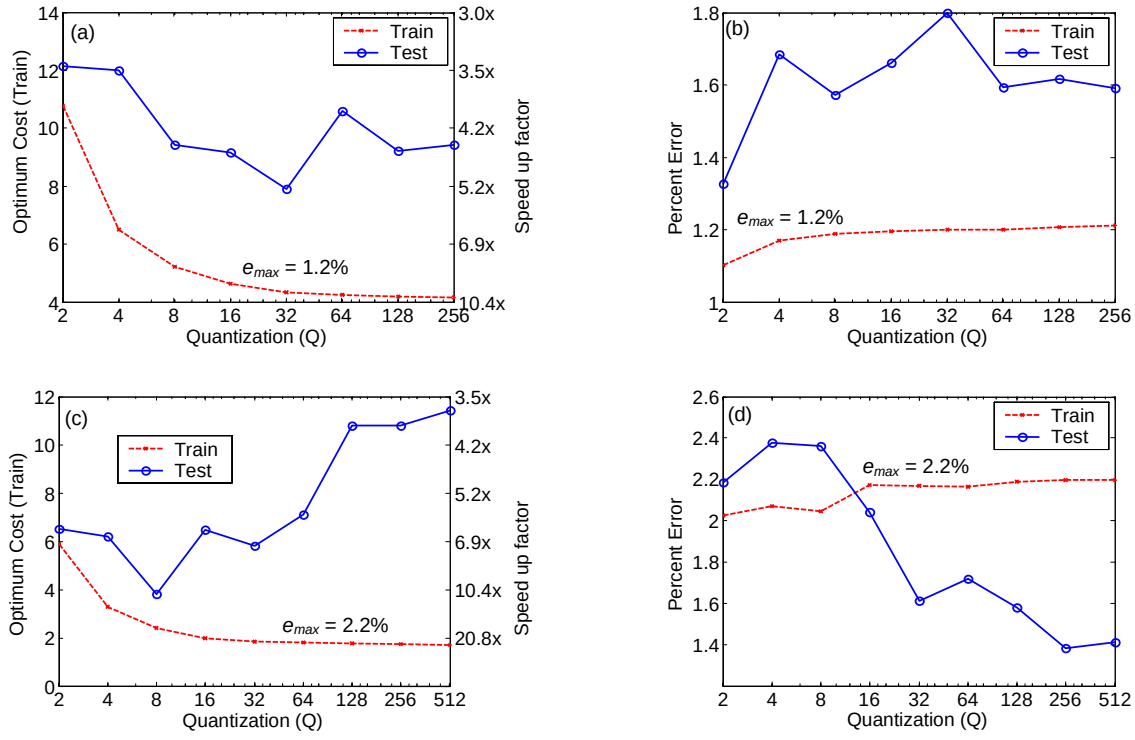

Figure 4. MNIST Error Constraint Results. Quantized DFS* was used find the optimal thresholds for 8 MNIST classifiers (1, 13, 4, 15, 14, 16, 17, and 18) such that cascade error, $e(T) \leq e_{max}$ and the cascade cost, $C(T)$, is minimized. The training and test costs and errors for the optimal solutions found are shown as a function of the number of quantization levels. Figures (a) and (b) present the costs and errors for e_{max} = 1.2% while figures (c) and (d) present the costs and errors for e_{max} = 2.2%. The training errors stay below e_{max}. Though the test errors go above e_{max} their deviation is bounded indicating that the optimal thresholds generalize well to unseen samples.

Generalization was tested using cross validation with test sets consisting of 5000 independent samples. The mean speed ups on the test set (using a 10-fold cross-validation) are 3.5x - 5x for e_{max} = 1.2% and 2.2%. The test speedups are expectedly lower as the optimal solutions found using DFS* are over-trained on the training set (as the cascades are optimal). In spite of the overtraining, the results clearly indicate that the speedups generalize to unseen samples. As seen in Figure 4(c) and 4(d), the larger the deviation between the train and test errors the greater the deviation between the costs and speedups. This behavior is independent of whether the test error was lower or higher than the train error. However, in both cases, there is a strong inverse relationship between the error rate and obtained speedup. By design, the training errors of the optimal solutions stay below e_{max}. Though the test errors go above e_{max} their deviation is bounded indicating that the optimal thresholds generalize to unseen samples.

Figure 5 presents results from cost constraint experiments. Quantized DFS* was used to find the optimal thresholds for 8 MNIST classifiers (1, 13, 4, 15, 14, 16, 17, and 18) such that cascade cost, $C(T) \leq C_{max}$ and the cascade error, $e(T)$, is minimized. The training and test costs and errors for the optimal solutions found are shown as a function of the number of quantization levels. Figures 5(a) and 5(b) present the training and test costs and speedups of the optimal solutions for C_{max} = 2.60, 5.20, 10.40, 20.81, and 41.62, corresponding to a speedup of 16x, 8x, 4x, 2x, and 1x, respectively. By design, the training costs stay below C_{max}. Figures 5(c) and 5(d) present the corresponding optimal (minimum) errors on the training and test sets, respectively. As expected, for each C_{max}, the optimal error (minimum) decreases monotonically as the number of quantized levels increases.

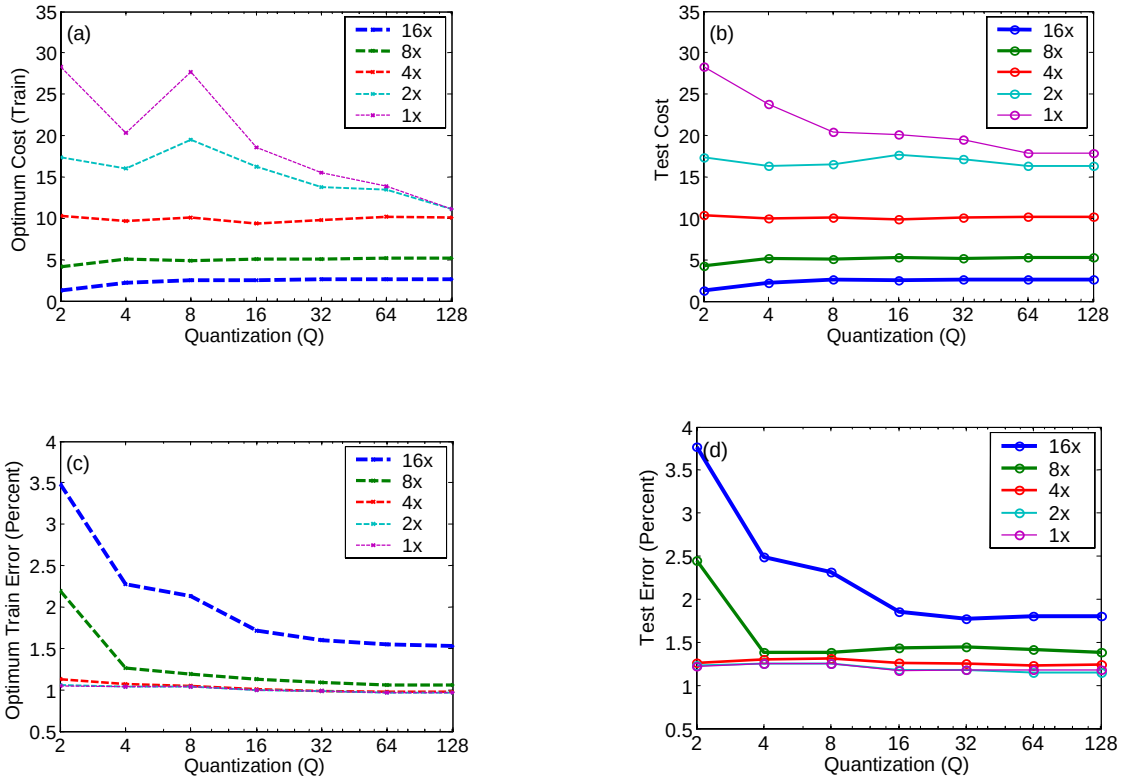

Figure 5. MNIST Cost Constraint Results. Quantized DFS* was used find the optimal thresholds for 8 MNIST classifiers (1, 13, 4, 15, 14, 16, 17, and 18) such that cascade cost, $C(T) \leq C_{max}$ and the cascade error, $e(T)$, is minimized. The training and test costs and errors for the optimal solutions found are shown as a function of the number of quantization levels. Figure (a) presents the cost of the optimal solutions for C_{max} = 2.60, 5.20, 10.40, 20.81, and 41.62, corresponding to a speedup of 16x, 8x, 4x, 2x, and 1x. Figure (b) presents the costs on the corresponding costs on the test set. The training costs stay below C_{max}. Figures (b) and (c) present the corresponding optimal (minimum) errors on the training and test sets, respectively. The test errors are only slightly above the corresponding training errors indicating that the optimal thresholds generalize well to unseen samples.

The test error curves (Figure 5(b) and 5(d)) clearly indicate that a 2x-4x speedup is easily possible with almost no change in test error, while a 4x-8x speedup appears possible with only a slight (10-15%) increase in error. The test errors are only slightly above the corresponding training errors indicating that the optimal thresholds generalize well to unseen samples. Comparing the training and test error curves, we see that the minimum test error gains quickly diminish with increasing number of quantization levels. 32 to 64 quantization levels are sufficient to ensure high quality solutions that generalize well. Comparing generalization properties of the error and cost constraint approaches, it is clear that the cost constraint is superior as the error and speedups generalize better to unseen samples.

5. CONCLUSION

A cascade architecture for combining classifiers was presented along with a branch-and-bound version of depth-first-search (referred to as DFS*) with efficient pruning for finding optimal cascade thresholds. The input to DFS* is a set of rejection curves computed by potential classifiers (of any type) and the output is a set of thresholds (which potentially eliminate unnecessary classifiers). The optimizer is called off-line and its output yields a near-optimal combination classifier, ready for deployment. It produces optimal solutions under arbitrary user specified speed and accuracy constraints. The effectiveness of the approach is demonstrated on handwritten character recognition by finding a) the fastest possible combination given an upper bound on classification error, and also b) the most accurate combination given a lower bound on speed. The speedup results generalize well to unseen data with 32 to 64 quantization levels. Among the error and cost constraint approaches presented, the cost constraint approach produces optimal solutions with error and speedup values that generalize better to unseen test samples. Future work will address scaling this approach to much larger data sets, larger cascade sizes, and re-training constituent classifiers in light of the role played by them in the cascade.

REFERENCES

1. Kuncheva LI, "A Theoretical Study on Six Classifier Fusion Strategies," *IEEE Trans. On Pattern Analysis and Machine Intelligence*, v. 24, No. 2, pp. 281-286, Feb. 2002.
2. RPW. Duin, "The Combining Classifier: To Train or Not to Train?" *Intl. Conf. on Pattern Recognition* (2), pp. 765-770, 2002.
3. J Kittler, M Hatef, RPW Duin, and J Matas, "On Combining Classifiers," *IEEE Trans. On Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, Mar. 1998. Mar. 1998.
4. L Prevost, C Michel-Sendis, A Moises, L Oudot, and M Milgram, "Combining model-based and discriminative classifiers: application to handwritten character recognition," *Intl. Conf. on Document Analysis and Recognition* (ICDAR'03). 2003.
5. H Hao, CL Liu, and H Sako, "Confidence evaluation for combining diverse classifiers," *Intl. Conf. on Document Analysis and Recognition* (ICDAR'03), pp. 760-765, 3-6 Aug. 2003.
6. U. Bhattacharya and B. B. Chaudhuri, "A Majority Voting Scheme for Multiresolution Recognition of Handprinted Numerals," ICDAR'03, pp. 16-20, 3-6 Aug. 2003.
7. PY Simard, D Steinkraus, and J Platt, (2003) "Best Practice for Convolutional Neural Networks Applied to Visual Document Analysis," in ICDAR'03, pp. 958-962.
8. Chellapilla K, Shilman M, and Simard P, "Combining Multiple Classifiers for Fast Optical Character Recognition," submitted to Document Analysis Systems (2006).