INCREMENTAL LEARNING USING THE TIME DELAY NEURAL NETWORK

Minh Tue Vo

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890
U.S.A.
Email: tue@cs.cmu.edu

ABSTRACT

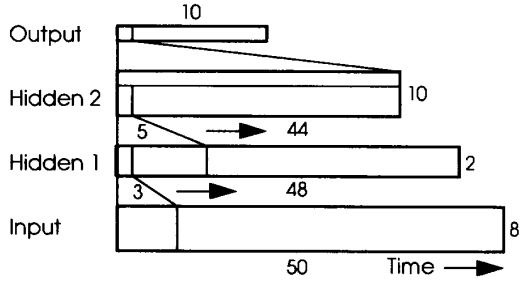
The Time Delay Neural Network (TDNN) is one of the neural network architectures that give excellent performance in tasks involving classification of temporal signals, such as phoneme classification, on-line gesture and handwriting recognition, and many others. One particular problem that occurs in on-line recognition tasks is how to deal with input patterns that are incorrectly recognized because they are totally dissimilar to anything the network has seen during training. In this paper we present an algorithm to add incremental, one-shot learning capability to the TDNN by creating extra hidden units to perform template matching on incorrectly recognized inputs and influence the output units via excitatory or inhibitory connections. In a simple handwritten digit recognition task, the addition of a single extra unit increases recognition rate for a new digit variation from 0% to 99%, while decreasing the performance on the old data by only 0.6%. Thus this Incremental TDNN (ITDNN) can in fact learn a new pattern from one example and perform reasonably well on similar inputs without forgetting what it already knew, thereby enabling it to deal effectively with the on-line misrecognition problem.

1. INTRODUCTION

Neural network techniques have been successfully applied to a wide range of pattern recognition tasks. The Time Delay Neural Network (TDNN) [8] is a network architecture particularly suited to the classification of temporal signals. This capability has been demonstrated for a variety of tasks, including phoneme classification [8], and more recently on-line gesture [7] and handwriting recognition [2][1].

The usefulness of gesture and handwriting recognition depends largely on the ability to adapt to new users because of the great range of variability in the way individuals write or make gestures. No matter how many tokens we put in the training database to cover different gestures that mean "delete text", for example, there will be someone who will use a totally different gesture and break the system. This is particularly troublesome for neural network-based systems because usually the network has to be retrained using all the old training data mixed with a large number of new examples, in order to be able to recognize new patterns without catastrophically forgetting previously learned patterns. Because of the large number of examples needed and the long retraining time, this clearly cannot be done on-line in a way that would enable the user to continue to work productively. A good system should be able to query the user for correction and remember this particular input pattern in order to make intelligent guesses when similar inputs occur and thus offer a reasonable level of performance until the network can be retrained off-line.

FIGURE 1: TDNN for Handwritten Digit Recognition



In this paper we propose to accomplish this for the TDNN by creating template-matching hidden units that influence the output units via excitatory or inhibitory connections. The incremental, one-shot learning capability of this Incremental TDNN, or ITDNN, is tested in a series of experiments involving a simple handwritten digit recognition task. We trained a TDNN to recognize written digits 0-9 and tested the incremental algorithm using a different variation of one of the digits.

The reported recognition rates are not intended to show how well TDNNs can recognize handwritten digits because the task is fairly simple and the data was kept fairly consistent. Rather, the results demonstrate that the ITDNN can immediately improve performance in the presence of completely new inputs and thus could prove very useful for systems requiring this capability, such as gesture/handwriting recognizers for pen-based computers.

2. THE ITDNN ARCHITECTURE

The ITDNN consists of a regular TDNN [8] augmented by special additional hidden units. The TDNN shown in Figure 1 is essentially the same as the one used in [8] to classify phonemes, except for network parameters. The power of the TDNN in classifying temporal data stems from its sliding timer-delay windows which enable the hidden units to discover temporal correlations in the input sequence during training by backpropagation [5][8].

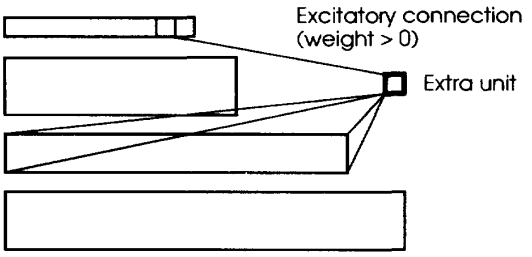
Raw input data goes through preprocessing steps based on those described by Guyon et al. [2] before being fed to the network. Incremental learning is accomplished by adding extra hidden units that influence the output units, similar to those used by Sato et al. [6] for non-TDNN feedforward networks.

2.1. Addition of Extra Units

The activation pattern of an incorrectly recognized input is used as a template to create an extra hidden unit whose weights are proportional to the activations in the first hidden layer, as

0-7803-1775-0/94 $3.00

FIGURE 2: Addition of an Extra Hidden Unit to Excite Output 8



shown in Figure 2. The motivation for this is to maximize the dot product of the weight and activation vectors so that future patterns close to the template in input space will also produce high activation in the extra unit, while other patterns will tend to deactivate it. We try to match activations in the first hidden layer rather than in the input layer because the first hidden layer of the TDNN is essentially a collection of "feature detectors" trained by backpropagation to recognize relevant features in the input.

The template activations are multiplied by a factor of proportionality and copied into the weight matrix of the new extra unit. The activation of this unit is determined by $sigmoid(b + \sum w_i h_i)$ where b is a bias value, w is the weight matrix, and h is the activation matrix of the first hidden layer. The quantity $\sum w_i h_i$ is considered the score for the match between the weights and the activations. The bias value and weight factor are carefully chosen to obtain good selectivity; quantitatively this means these parameters are such that only activation patterns that give a match score better than 80% of the original match score with the template will produce a high activation in the extra unit. Note that reducing the bias increases selectivity by requiring a better match score.

We increase or decrease output activations to correct the classification by connecting such extra units to the outputs using positive (excitatory) or negative (inhibitory) weights of large enough magnitudes to drive the outputs to the desired values. Each extra unit is connected to a single output unit; if more than one output unit needs to be corrected, we create more than one extra unit based on the same template (possibly with different output weights, depending on the desired corrections.)

2.2. Fine-tuning Extra Units

When subsequent input patterns are presented to the network and a recognition error occurs, fine-tuning procedures (modeled after the ones in [6]) are tried before additional extra units are considered. These procedures adjust the bias values and output weights to turn off unwanted extra units and strengthen wanted ones. The procedure is as follows:

1. To increase the activation of an output unit:
   - Find an inhibitory extra unit connected to this output and turned on by the current input pattern; reduce the bias of this extra unit until it is turned off; or
   - Find an excitatory extra unit connected to this output and turned on by the current input pattern; increase its output weight until output activation is high enough.
2. To lower the activation of an output unit:
   - Find an excitatory extra unit connected to this output and turned on by the current input pattern; reduce the bias of this extra unit until it is turned off; or

FIGURE 3: Activations of Subunits Matching Successive Sections in the Activation Matrix

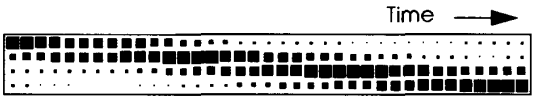
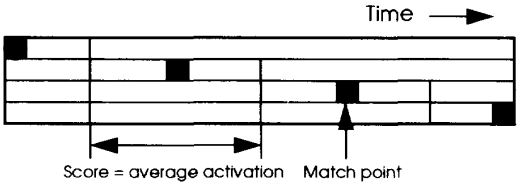


FIGURE 4: Linear Time Warping for Subunits



   - Find an inhibitory extra unit connected to this output and turned on by the current input pattern; increase its output weight until output activation is low enough.

It is not wise to reduce a bias too much because the extra unit could become too selective and reject even good matches. Likewise, making an output weight too large is also disadvantageous because this could let poor matches contribute significantly to influencing the output even when the extra unit's activation is fairly low. For these reasons we impose a minimum bias value and a maximum output weight for all extra units.

2.3. Time-warped Extra Units

In the above implementation, added extra units simply match the whole activation matrix and thus do not take advantage of the time-shift invariant property of the TDNN. One way to remedy this is to divide the activation matrix into sections and decompose each extra unit into subunits, assigning one subunit to match each section of activations (in our experiments we used 4 subunits per extra unit.) Sliding the time windows of these subunits along the time dimension produces activation traces similar to the one shown in Figure 3. Note that each subunit is turned on around the area in which its weights match the activations. These subunits in fact represent successive states in the activation trace, indicating that we need to employ some time-warping technique to evaluate the quality of the match.

We first tried the dynamic time warping (DTW) algorithm [4] to find the optimal state transition path. This turned out not to work very well because activation traces for some input patterns (corresponding to different digits) are very similar but shifted in time, and DTW is too good at warping these paths to match. Duration control did not greatly improve the situation. We believe the poor applicability of DTW in this case is caused by the artificial nature of the state assignments. The subunits seem to represent successive states but actually these states were artificially imposed on the input by dividing the activation matrix into equal-length sections with no regard to the contents.

The above arguments led us to try "linear time warping" as shown in Figure 4. We identify the points at which the weights of the subunits match the template activations exactly, and compute the score for each subunit by averaging its activations around its corresponding match point only. This seemingly simplistic approach turned out to work better than DTW for our ITDNN. Fine-tuning this kind of time-warped extra units is done in the same way as for non-time-warped units (see Section 2.2.)

3. DATA COLLECTION PROCEDURE

We evaluated the ITDNN's incremental learning capability using a simple handwritten digit recognition task. The databases used in network training and testing were collected from a single person writing digits 0-9 on a digitizing tablet with a pressure-sensitive stylus. The raw data consists of sequences of tablet coordinates and pen pressures tracing the handwritten strokes; this data stream is preprocessed as mentioned in the previous section.

The data set for each digit is divided into two or three variations; for example, a "0" can be written in a clockwise or counter-clockwise direction, a "7" can have a bar in the middle ("European 7") or not ("American 7"). Within each variation the data is kept fairly consistent because the purpose of this experiment is to find out if the ITDNN can adapt to a new input variation.

One variation of each digit is selected to form a database of all the digits, henceforth referred to as the ORIGINAL database. The experiments described in the next section also make use of one variation of "6" (written in a clockwise direction rather than counter-clockwise as in ORIGINAL), examples of which are collected into a database called VARIATION. Preliminary experiments identified the combination of variations used here as being relatively difficult to classify due to similarities between different digits, e.g., "0" and "6" both written in the same direction.

ORIGINAL and VARIATION are each further divided into a training set and a test set; examples in the test sets are never seen by the networks during training. The training and test sets for ORIGINAL contain 1000 examples each (100 for each digit), while the data sets for VARIATION have 100 examples each. The two training sets are also combined by randomly interleaving examples from each set; the test sets are combined in the same way. The resulting data sets form a database called COMBINED.

4. EXPERIMENTAL RESULTS

The parameters of the TDNN used here were selected by trial-and-error to find a network large enough to perform well on the ORIGINAL database but small enough to give substantially poorer results when a digit variation is added. We trained two TDNNs with exactly the same parameters on ORIGINAL and COMBINED, respectively called NET-ORG and NET-COM. Three versions of each network were trained starting from 3 different sets of random initial weights. Because of the small network size and the fairly high consistency within each digit variation, convergence was achieved after about 10 epochs in each case. Performance results for the network versions giving the lowest error rates are reported in Table 1. For each pair of performance figures, the upper number is the % recognition rate for the training set, and the lower number is for the test set.

TABLE 1: Performance of Regular TDNNs

Database	NET-ORG	NET-COM
ORIGINAL	100.0	87.9
	99.3	81.3
VARIATION	0.0	98.0
	0.0	100.0
COMBINED	90.9	88.8
	90.3	83.0

We built up ITDNNs (using non-time-warped extra units) from the trained NET-ORG by incremental training using VARIATION. We made many runs with successively higher limits on the maximum allowable number of extra units. The results are reported in the upper half of Table 2. We also fine-tuned each ITDNN by incremental training using COMBINED while disallowing additional extra units in order to force the network to adjust the bias values and output weights only. The results from this fine-tuning procedure are reported in the lower half of Table 2. The shaded entries are the most relevant to the discussion in the next section.

TABLE 2: Performance After Adding Extra Units

	Database	1 extra unit	2 extra units	3 extra units
NO FINE-TUNING	ORIGINAL	99.6	91.6	91.0
		99.3	93.7	93.5
	VARIATION	97.0	100.0	100.0
		80.0	100.0	100.0
	COMBINED	99.4	92.4	91.8
		97.5	94.3	94.1
WITH FINE-TUNING	ORIGINAL	99.9	96.5	96.4
		99.3	98.4	98.4
	VARIATION	75.0	99.0	99.0
		42.0	96.0	97.0
	COMBINED	97.6	96.7	96.6
		94.1	98.2	98.3

We repeated the above procedure but this time we used the time-warped extra units described in Section 2.3. The results are summarized in Table 3.

TABLE 3: Performance for Time-Warped Extra Units

	Database	1 extra unit	2 extra units	3 extra units
NO FINE-TUNING	ORIGINAL	99.4	98.4	86.4
		99.0	98.6	87.8
	VARIATION	99.0	99.0	100.0
		92.0	92.0	100.0
	COMBINED	99.4	98.5	87.6
		98.4	98.0	88.9
WITH FINE-TUNING	ORIGINAL	99.7	99.5	92.7
		99.3	99.3	94.8
	VARIATION	96.0	96.0	100.0
		82.0	83.0	100.0
	COMBINED	99.4	99.2	93.4
		97.7	97.8	95.3

5. DISCUSSION

From the results reported in the previous section, we can make the following observations (note that the figures quoted are for training data unless specified otherwise.)

5.1. Non-time-warped Extra Units

- One extra unit is enough to cover almost all (97%) the examples of the new variation while reducing the perfor-

mance on the original data only marginally if at all (from 100% down to 99.6%.)

- Additional extra units cover the remaining few examples of the new variation but can cause performance on the original data to drop significantly (from 99.6% down to 91.6% when the 2nd extra unit is added.) An analysis of the activations in the network reveals that the new "6" is easily confused with "0" and "5" in ORIGINAL; 1 extra unit is not enough to affect performance appreciably, but 2 extra units matching similar patterns cause some 0s and 5s to be incorrectly classified as 6s.
- Fine-tuning extra units improves the performance for the original data but dramatically decreases the performance on the new variation (from 97% down to 75%) because of the increased selectivity of the extra units, thus hurting the overall performance.
- The overall recognition rates of NET-ORG augmented by extra units exceed those achievable by the non-incremental TDNN of the same size (NET-COM) trained on the original data combined with the new variation.

5.2. Time-warped Extra Units

The above observations also apply to the ITDNN constructed using time-warped extra units. However, adding a 2nd extra unit has less effect on the recognition rate for the original data (from 99.4% down to 98.4% instead of 91.6% as for non-time-warped units) and fine-tuning does not hurt performance on the new variation as much (from 99% down to 96% instead of 75%.) This means the time-warped units achieve a better match function.

6. CONCLUSIONS AND FUTURE DIRECTIONS

In this paper we have described a technique to augment a TDNN with extra hidden units in order to correct recognition errors using template matching. The experimental data shows that the ITDNN is capable of quickly adding coverage for a new input variation without forgetting previously learned information and thus is a good candidate for systems requiring on-line, immediate recognition improvement during use, such as gesture and handwriting recognizers for pen-based computers. Such systems capable of incremental learning will in fact be able to adapt quickly to a new user at a reasonable level of performance to let the user do productive work; during the subsequent work sessions new data can be quietly collected for off-line training of a regular network that will do a better job later on.

An alternative to off-line retraining is to develop a "network compacting" algorithm that will allow the extra units to be integrated into the regular TDNN in some fashion. Some fine-tuning (by making only a few passes through the original training data combined with the newly collected data) may still be needed to bring the performance to a level comparable to complete retraining, but there will still be a net saving in training costs. It is possible that the time-warped extra units will facilitate the integration process because their structure is closer to the original TDNN. If we could find a way to remove the artificial nature of the state assignment in these units and employ real DTW successfully, we could replace the TDNN with the Multi-State TDNN (MS-TDNN) [3] which also uses DTW to exploit the state succession mechanism; this should facilitate network compaction even further. Developing and refining these techniques would give us a novel neural network structure capable of not only learning incrementally but also achieving better performance than the regular TDNN and MS-TDNN.

Our experiments show that time-warped extra units (using "linear warping" instead of DTW) do better at template matching than non-time-warped units. It seems that at most one extra unit should be used for each new input variation; otherwise the performance on the original data may suffer. This leads to the question of how to evaluate the worth of adding an extra unit quickly and effectively when the target system is in use. The most feasible approach we can see at this time is to try the network augmented by the new unit on a test set of the original data to see if the addition of the new extra unit is beneficial.

The experiments did not demonstrate any clear advantage of fine-tuning extra units on the original and new data combined. This may be due to the fact that our data is so consistent that the network already achieves a very low error rate even without fine-tuning, and thus fine-tuning cannot significantly improve performance on the original data while performance on the new variation can only go down. We may need to reevaluate the worth of fine-tuning using a more realistic training corpus.

Another research direction worth investigating is how to generalize the "templates" (which will cease to be true templates) using subsequent examples to improve the matching. This will help avoid the problem of hurting performance on new data after fine-tuning on the old data, caused by our simple scheme of adjusting bias values and output weights.

7. ACKNOWLEDGEMENTS

This research was sponsored by the Department of the Navy, Office of Naval Research under Grant number N00014-93-1-0806.

The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Navy or the U.S. Government.

The author would like to thank Dr. Alex Waibel and Herman Hild for their helpful suggestions.

8. REFERENCES

[1] U. Bodenhausen and S. Manke, "Connectionist Architectural Learning for High Performance Character and Speech Recognition," *ICASSP-93*.

[2] I. Guyon, P. Albrecht, Y. LeCun, W. Denker, and W. Hubbard, "Design of a Neural Network Character Recognizer for a Touch Terminal," *Pattern Recognition*, 1991.

[3] P. Haffner, M. Franzini, and A. Waibel, "Integrating Time Alignment and Neural Networks for High Performance Continuous Speech Recognition," *ICASSP-91*.

[4] H. Ney, "The Use of a One-Stage Dynamic Programming Algorithm for Connected Word Recognition," *ICASSP-84*.

[5] D. Rumelhart and J. McClelland, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, MIT Press, 1986.

[6] A. Sato, K. Yamada, J. Tsukumo, and T. Temma, "Neural Network Models for Incremental Learning," *ICANN-91*.

[7] M.T. Vo and A. Waibel, "A Multimodal Human Computer Interface: Combination of Gesture and Speech Recognition," *Adjunct Proc. InterCHI-93*, Amsterdam, April 1993.

[8] A. Waibel, T. Hanazawa, G. Hinton, K. Shiano, and K. Lang, "Phoneme Recognition using Time- Delay Neural Networks," *IEEE Trans. ASSP*, 1989.