PARALLEL TRAINING OF NEURAL NETWORKS FOR SPEECH RECOGNITION

Karel Veselý
Master Degree Programme (2), FIT BUT
E-mail: xvesel39@stud.fit.vutbr.cz

Supervised by: Lukáš Burget,
E-mail: burget@fit.vutbr.cz

ABSTRACT

The feed-forward multi-layer neural networks have significant importance in speech recognition. They can be used for phoneme-state classification, speech parametrisation, language models and for language or speaker recognition. The need for the acceleration of neural network training is caused by huge quantities of training data. A new training tool *TNet* was designed and optimized for multi-processor computers. The acceleration rates are reported on a real speech processing task.

1 INTRODUCTION

The feed-forward multilayer neural networks (NN) have many practical applications. They can be used for classification, pattern recognition, prediction, dimensionality reduction or control. In case of speech recognition, the neural networks are typically used as phoneme-state classifiers [1]. The NNs' input is speech frame transformed to feature vector and the output is a vector of phoneme-state class membership probabilities.

Even if we use previous parallel NN training implementation *SNet* [2], typically the training time exceeds 24 hours (using 163h AMIDA corpus, four-layer network with 1 million parameters and 6 client parallelization). The long training periods are uncomfortable for practical use. This paper describes *TNet* – a new faster implementation of parallel neural network training.

2 PARALLELIZATIONS

Feed-forward neural network is an adaptive multivariate transform function with ability to "memorize" the training examples. It can be seen as a sequence of linear and nonlinear transformations. *Supervised learning* is used, the learning consists of correcting the linear transform weights. Due to Kolmorogovs' theorem [3] we believe that the network is able to express any possible function when having enough layers and neurons per layer. Standard *on-line gradient descent algorithm* with *error backpropagation* is used for the learning, the weight updates are done per bunch (a block of N frames).

The *on-line learning* imposes strong data dependencies which makes the parallelization difficult. Two effective approaches to parallel network training have been reported [4]:

Data parallelization The training data is divided into disjoint sets. Each thread has its own network instance and works on its own data-set. Weight synchronisation occurs periodically when *N* frames are processed. The weight difference matrices are gathered, summed up and a new set of weights is generated and distributed.

Node parallelization In this case, there is only one instance of network. The network layers are divided into disjoint sets of neurons. Each thread has associated its own set. This method imposes higher frequency of synchronisation than *data parallelization* method. The threads are synchronized by a *barrier* before one can proceed to the next layer.

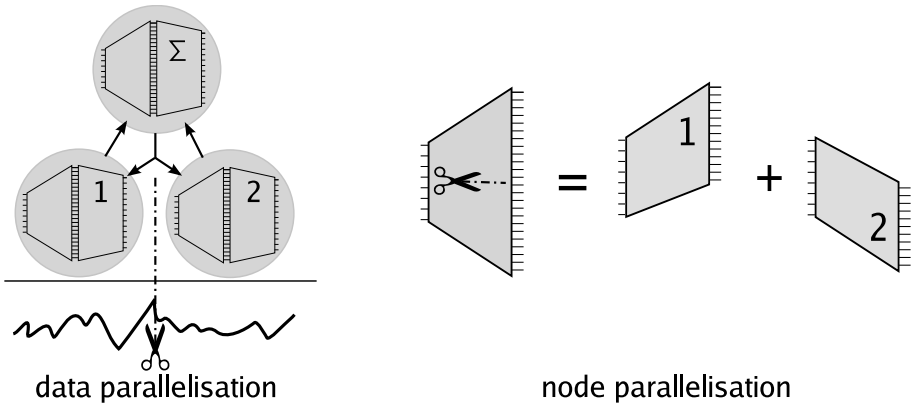

Figure 1: Two types of parallelization

The problem is that with *data parallelization* the overhead of weight synchronisation increases by adding more slave threads while with *node parallelization* poor cache performance will slow-down the training when layer division sets are too small. The promising strategy can be to combine both approaches and find the optimal operation point.

3 SNET VS. TNET

Our baseline is given by previous implementation of parallel NN training *SNet*. The new implementation *TNet* is faster because it is multi-threaded application where all the threads share same address space, while *SNet* is a client server application which uses TCP-IP connection for weight synchronisation.

So far the *TNet* implements *data parallelization*. The design of the tool was chosen with respect to both high performance and simple extensibility. The GotoBLAS[1] library is used to accelerate linear algebra operations. The neuron weights are shared for all the threads which improves the processor cache hit-rate. The network consists of polymorphic classes derived from base class "Component", this ensures the extensibility.

4 EXPERIMENTS

Two-layer network topology was used for all the experiments. The network consists of 598 inputs, 1000 neurons in the hidden layer and 135 neurons in the output layer. The sigmoid

[1] http://www.tacc.utexas.edu/tacc-projects/

nonlinearity is used for the hidden layer and softmax nonlinearity for the output layer.

The network was trained on 10h subset of AMIDA corpus, the cross-validation was performed on 1h from the same corpus. The parameterisation based on critical bands and long temporal contexts (510ms) was used. Same cross-validation accuracies were obtained both by *TNet* and *SNet*. The training speed of *TNet* and *SNet* was measured 10x for each parallelization order. The experiments were performed on the HP ProLiant DL785 G5 server with 8 quad-core AMD Opteron 8356 processors (32 cores). The measured training accelerations are in Fig. 2.

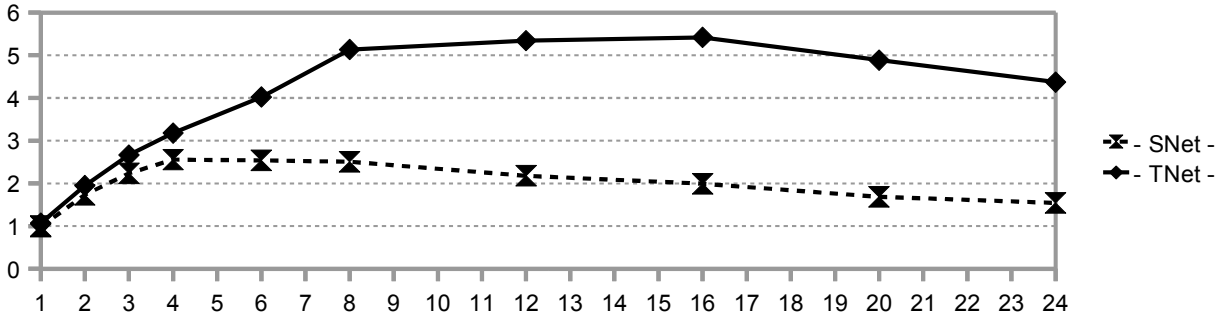

Figure 2: Training acceleration by parallelization

5 CONCLUSION

As we can see in Fig. 2, the *TNet* implementation is 2x faster than *SNet* in case of 8 core parallelization. Further acceleration is still possible, but it will require more complicated and less universal design. Another promising way of acceleration is the use of modern GPUs, which can offer massive parallelization. These experiments will be subject of future work.

ACKNOWLEDGMENTS

This work was partly supported by Grant Agency of Czech Republic project No. 102/08/0707, and by Czech Ministry of Education project No. MSM0021630528, and by Czech Ministry of Interior project No. VD20072010B16, and by the BUT FIT grant FIT-S-10-2, and the research plan MSM0021630528.

REFERENCES

[1] Bourlard, H., Morgan, N.: Connectionist Speech Recognition a Hybrid Approach. Norwell MA USA, Kluwer Academic Publishers 1993, ISBN 0-79-239396-1

[2] Kontár, S.: Paralelní trénování neuronových sítí pro rozpoznávání řeči. [diplomová práce], FIT VUT v Brně

[3] Jan, J.: Číslicová filtrace, analýza a restaurace signálů, Vutium, 2002, ISBN 80-214-1558-4

[4] Pethick, M., Liddle, M., Werstein, P., Huang, Z.: Parallelization of a Backpropagation Neural Network on a Cluster Computer. Dunedin, New Zealand, University of Otago 2003