Improve Performance in Deep Neural Networks: (1) Cost Functions, and (2) Reusable learning

Chetak Kandaswamy

19th February 2014

NNIG Technical Report No. 3/2014

Project "Reusable Deep Neural Networks: Applications to Biomedical Data"
(PDTC/EIA-EIA/119004/2010)

Neural Networks Interest Group

Instituto de Engenharia Biomédica (INEB)
Rua Dr. Roberto Frias, 4200-465, Porto, Portugal

Contents

1 Introduction **3**

2 GPU Parallel Processing **3**

3 Paper overview: Using different cost functions to train stacked auto-encoders **4**

4 Paper overview: Improving Accuracy on Transductive Transfer Learning by Reusing SDA **5**

5 Present Experiment overview: Improving Deep Convolutional Neural Network by Reusing Features Trained with Transductive Transfer Setting **6**

References **6**

1 Introduction

This report presents an overview of three manuscripts. First, an overview of a paper [1] presented at the Mexican International Conference on Artificial Intelligence (MICAI 2013). Second, manuscript [6] that we have recently submitted to International Joint Conference on Neural Networks conference (IJCNN 2014). Finally, ongoing experiments on Stacked autoencoders and Convolutional neural networks.

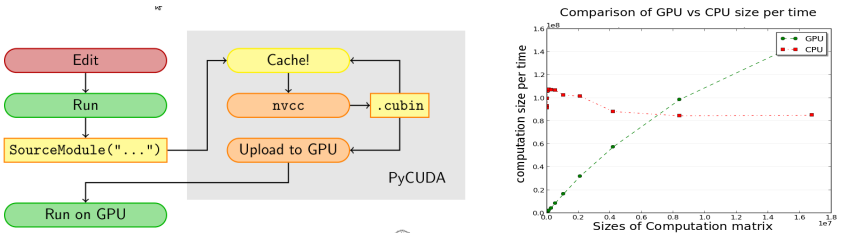
We performed all our experiments on a computer with i7-377 (3.50GHz) 16GB RAM. For faster processing of larger datasets we used Theano Bergstra et al. [4] a GPU compatible machine learning library on a GTX 770 GPU. We used Theano on GPU parallel processing to increase the computation capability than a CPU.

In Section 2 presents an overview GPU Parallel Processing capability in comparison with CPU. In Section 3 presents an overview of paper presented at MICAI 2013. In this paper we describe experiments involving different combinations of pre-training and fine-tuning cost functions. In Section 4 presents an overview of a manuscript submitted to IJCNN 2014, reporting on experiments in which we explored the use of transfer learning to improve the performance of deep neural networks with transductive transfer problems [6].

2 GPU Parallel Processing

The GPU parallel processing allows training both Convolutional Neural Network's and Stacked denoising Autoencoder's with millions of neural connection, for very small learning rate, for large number of epochs, for very large datasets within several days. Each of these experiments are repeated 10 times to increase confidence level of the results. The performance GPU parallel process over CPU is shown in Fig 1b.

The PyCUDA runs like a regular compiler except the ''edit-run-repeat'' working as shown in Fig1a. The compilation and caching operations in the gray box are performed without user involvement.



(a) PyCUDA workflow for GPU [11]

(b) GPU vs CPU

Figure 1: Performance of GPU over CPU with PyCUDA

3 Paper overview: Using different cost functions to train stacked auto-encoders

Deep architectures, such as neural networks with two or more hidden layers of units, are a class of machines that comprise several levels of non-linear operations, each expressed in terms of parameters that can be learned [2]. The organization of the mammal brain, as well as the apparent depth of cognitive processes, are among the main motivations for the use of such architectures. In spite of this, until 2006, attempts to train deep architectures resulted in poorer performance than that achieved by their shallow counterparts. The only exception to this difficulty was the convolutional neural network [13], a specialized architecture for image processing, modeled after the structure of the visual cortex.

A breakthrough took place with the introduction by Hinton *et al.* of the deep belief network [9], a learning approach where the hidden layers of a deep network are initially treated as restricted Boltzmann machines (RBMs) [17] and pre-trained, one at a time, in an unsupervised greedy approach. Given that auto-encoders [5] are easier to train than RBMs, this unsupervised greedy procedure was soon generalized into algorithms that pre-train the hidden levels of a deep network by treating them as a stack of auto-encoders [3, 12].

The auto-encoder (also called auto-associator or Diabolo network) is a type of neural network trained to output a reconstruction of its own input. Thus, in the training of auto-encoders, input vectors can themselves be interpreted as target vectors. This presents an opportunity for the comparison of various training criteria, namely different cost functions capable of reflecting the mismatch between inputs and targets.

In this work, we further investigate the information theoretical concept of minimum error entropy has been recently applied by Marques de Sá *et al.* Marques de Sá et al. [14] to data classification machines, yielding evidence that risk functionals do not perform equally with respect to the attainment of solutions that approximate the minimum probability of error. By compare the performances of squared errors (SSE), cross-entropy (CE), and exponential (EXP) costs when employed both in the unsupervised pre-training and in the supervised fine-tuning of deep networks whose hidden layers are regarded as a stack of auto-encoders. To the best of our knowledge, this type of comparison has not been done before in the context of deep learning. Using a number of artificial and real-world data sets, we first compared pre-training cost functions in terms of their impact on the reconstruction performance of hidden layers. Given that the output layer of our networks was designed for classification learning, we also compared various combinations of pre-training and fine-tuning costs in terms of their impact on classification performance.

In general, the best layer-wise reconstruction performance was achieved by SSE pre-training, though with binary data CE yielded the lowest errors for the first hidden layer. Classification performance was found to vary little with the combination of pre-training and fine-tuning costs. When pre-training with CE, fine-tuning via SSE was found not to be a good choice. In general, the choice of the same pre-training and fine-tuning costs yielded classification errors with

lower variance.

We performed experiments using CPU and GPU to allow the use of more computationally demanding data sets, such as variants of the popular MNIST character recognition set.

4 Paper overview: Improving Accuracy on Transductive Transfer Learning by Reusing SDA

In this work we show using Stacked Denoising Auntoencoders the unsupervised feature transference outperform randomly initialized machine on a new problem. We achieved 7% relative improvement on average error rate and 50% on average computation time with uppercase letters dataset. In the case of supervised feature transference, we achieved 5.7% relative improvement for average error rate by reusing first or second hidden layer to classify the uppercase letters dataset, and 8.5% relative improvement for average error rate by reusing all three hidden layers of a problem that was fine-tuned again with the uppercase letters dataset.

A good machine learning method should be able to self-learn the patterns and extract information from the data, able to reuse the information to solve a similar but different problem, and possible to compete with state-of-the-art technology. Deep learning and Transfer learning are machine learning methods, mimics the multi-layered, deep, and sparsely connected model of the human brain. Deep learning method extracts information same as hierarchical learning method of Human. In Transfer learning, human learns simple concepts first and then builds complicated ideas. In this project, we are interested in reusable deep learning methods for classification of data.

The transfer learning survey [15] also indicates various transfer settings which are commonly classified as *Inductive, Transductive and Unsupervised* transfer learning. In inductive transfer learning the source and target problems have different but related distribution. Research on inductive transfer by Heigold et al. [8] and Huang et al. [10] has shown that by transferring supervised features for classifying cross-lingual speech recognition problems, using deep architectures, improved performance over the one achieved by shallow architectures. The research work of Ciresan et al. [7] has shown transfer learning for latin and chinese characters with deep convolution neural network achieve better transference. The study by Raina et al. [16] shows that a machine is able to learn higher-level features by transferring unsupervised features from source to target problem, for both inductive and transductive transfer settings.

In this paper, we analyze feature transference using Stacked Denoising Autoencoders (SDA) for two different approaches: 1) unsupervised feature transference (USDA); 2) supervised layer based feature transference (SSDA). For that purpose we have carried out experiments to study the transductive transfer learning of *arbitrary distribution* of source and target problems for both USDA and SSDA approaches, for example, by training a machine to classify images of digits

0-to-9 and reusing these trained features to classify images of English characters a-to-z. We also performed experiments by reversing the problem roles: training a machine with images of English characters a-to-z and reusing the features to classify images of digits 0-to-9. Furthermore, we also studied inductive transfer learning of *different but related* problem for USDA approach. Processing large numbers of instances as we did, on millions of neural connections, would take several weeks using traditional CPUs. We used instead a GPU parallel processor for faster processing of these large networks involving several repetitions for both inductive and transductive transfer settings.

We studied the performance of feature transference for both transductive and inductive transfer settings. Both unsupervised (USDA) and supervised (SSDA) feature transference approach significantly reduced the average error and computation time of the baseline for the harder case problems.

5 Present Experiment overview: Improving Deep Convolutional Neural Network by Reusing Features Trained with Transductive Transfer Setting

From the previous paper reusing L1, L2, L3, L1+L2, L2+L3, L1+L3, L1+L2+L3 we observe that reusing L1 and L1+L2 perform better other approaches for both uppercase and lowercase datasets. Thus we perform experiment to test the importance training sample size on the supervised feature transference. The results are in the FCT technical report.

References

[1] T. Amaral, L. M. Silva, L. A. Alexandre, C. Kandaswamy, J. M. Santos, and J. Marques de Sá. Using different cost functions to train stacked auto-encoders. In *Mexican International Conference on Artificial Intelligence (MICAI)*, pages 114–120, 2013. 3

[2] Y. Bengio. Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127, 2009. 4

[3] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Neural Information Processing Systems Conference*, volume 19, pages 153–160, 2007. 4

[4] James Bergstra, Olivier Breuleux, Frederic Bastien, Pascal Lamblin, Razvan Pascanu, Guillaume Desjardins, Joseph Turian, David Warde-Farley, and Yoshua Bengio. Theano: a CPU and GPU math expression compiler. In *Proceedings of the Python for Scientific Computing Conference (SciPy)*, volume 4, 2010. 3

[5] H. Bourlard and Y. Kamp. Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294, 1988. 4

[6] LuÃs M. Silva LuÃs A. Alexandre Jorge M. Santos Chetak Kandaswamy, Joaquim Marques de SÃ¡. Improving accuracy on transductive transfer learning by reusing sda. *(Under Review)*. 3

[7] Dan Claudiu Ciresan, Ueli Meier, and Jürgen Schmidhuber. Transfer learning for latin and chinese characters with deep neural networks. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–6. IEEE, 2012. 5

[8] G Heigold, V Vanhoucke, A Senior, P Nguyen, M Ranzato, M Devin, and J Dean. Multilingual acoustic models using distributed deep neural networks. ICASSP, 2013. 5

[9] G. Hinton, S. Osindero, and Y.-W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006. 4

[10] Jui-Ting Huang, Jinyu Li, Dong Yu, Li Deng, and Yifan Gong. Cross-language knowledge transfer using multilingual deep neural network with shared hidden layers. In *Proc. ICASSP*, 2013. 5

[11] Andreas Klockner, Nicolas Pinto, Yunsup Lee, Bryan Catanzaro, Paul Ivanov, and Ahmed Fasih. PyCUDA and PyOpenCL: a scripting-based approach to GPU run-time code generation. *Parallel Computing*, 38 (3):157– 174, March 2012. ISSN 0167-8191. doi: 10.1016/j.parco.2011.09.001. URL http://www.sciencedirect.com/science/article/pii/S0167819111001281. Cited by 0051. 3

[12] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *International Conference on Machine Learning*, pages 473–480, 2007. 4

[13] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 4

[14] J. Marques de Sá, L. Silva, J. Santos, and L. Alexandre. *Minimum Error Entropy Classification*, volume 420 of *Studies in Computational Intelligence*. Springer, 2013. 4

[15] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *Knowledge and Data Engineering, IEEE Transactions on*, 22(10):1345–1359, 2010. 5

[16] Rajat Raina, Alexis Battle, Honglak Lee, Benjamin Packer, and Andrew Y. Ng. Self-taught learning: transfer learning from unlabeled data. In *Proceedings of the 24th international conference on Machine learning*, pages 759–766, 2007. 5

[17] P. Smolensky. *Parallel distributed processing: explorations in the microstructure of cognition*, volume 1, chapter Information processing in dynamical systems: Foundations of harmony theory, pages 194–281. University of Colorado, 1986. 4