Schedae Informaticae Vol. 24 (2015): 41–51
doi: 10.4467/20838476SI.15.004.3026

# Effectiveness of Unsupervised Training in Deep Learning Neural Networks

Andrzej Rusiecki[1], Mirosław Kordos[2]
[1]Wroclaw University of Technology
Faculty of Electronics
ul. Wybrzeże Wyspiańskiego 27, 50-370 Wrocław, Poland
e-mail: *andrzej.rusiecki@pwr.edu.pl*
[2]University of Bielsko-Biała
Department of Computer Science
ul. Willowa 2, 43-309 Bielsko-Biała, Poland
e-mail: *mkordos@ath.bielsko.pl*

**Abstract.** Deep learning is a field of research attracting nowadays much attention, mainly because deep architectures help in obtaining outstanding results on many vision, speech and natural language processing – related tasks. To make deep learning effective, very often an unsupervised pretraining phase is applied. In this article, we present experimental study evaluating usefulness of such approach, testing on several benchmarks and different percentages of labeled data, how Contrastive Divergence (CD), one of the most popular pretraining methods, influences network generalization.

**Keywords:** neural networks, deep learning, restricted Boltzmann Machine, contrastive divergence.

## 1. Introduction

Many complicated machine learning tasks, including computer vision, natural language processing, speech recognition, etc. require representing high-level abstractions [1,2]. The recent researches demonstrated that these representations can be obtained by deep learning methods, designed to establish deep structures. Deep learning algorithms very often match or even outperform the state-of-the-art results on many sophisticated benchmarks [3].

Deep architectures [4], such as artificial neural networks with many hidden layers, usually need to be trained in two stages. The first part of the training process is so-called pretraining, which aims typically at building deep feature hierarchy, and is performed in an unsupervised mode. The latter stage is supervised fine tuning of the network parameters.

Optimizing a training criterion for deep architectures is a very difficult task. As it was experimentally demonstrated [4], for hundreds random initializations, the gradient-based training process may lead to hundreds different local minima. This is why an additional mechanism to regular optimization is required. The network is initialized with the use of unsupervised algorithm, which helps in obtaining better generalization after the network is learned.

There are several hypothesis concerning network pretraining [5]. It is often noticed that unsupervised pretraining helps in extracting important features from the data, as well as in setting initial conditions for the supervised algorithm in the region in the parameter space, where better local optimum may be found. Some hypothesis claim that the pretraining phase is a kind of very particular regularization, which is performed not by changing the optimized criterion or introducing new restriction for the parameters, but by creating a starting point for the optimization process. Regardless of the reason, unsupervised pretraining helps in creating efficient deep architectures.

Hence, the deep learning scheme is usually as follows: greedy, layer-wise unsupervised pretraining precedes classic supervised network training. Each layer is trained separately, based on the output signals of the previous layer. Because only inputs are used and no target values for the layers are required, such unsupervised learning may be performed without labeled data (in this phase of training even unlabeled examples can be used). Depending on the architecture, two main algorithms are applied. The most popular Deep Belief Networks consisting of Restricted Boltzmann Machines (RBM) are trained layer-by-layer with so-called Contrastive Divergence and its variants [6–8], while Stacked Autoencoders and Stacked Denoising Autoencoders [4] are made of feedforward layers previously trained to reconstruct their inputs. Because of limited amount of space in the paper we decided to focus only on Contrastive Divergence pretraining.

It is not clear, however, when applying unsupervised pretraining indeed increases network performance (defined here as its generalization ability). Previous experiments suggest that, in certain conditions, when we consider shallow architecture, or a network with relatively small hidden layers, using unsupervised pretraining may even disturb in classifying new patterns. In this study we intend to examine how the unsupervised pretraining with Contrastive Divergence influences network accuracy for several classification tasks and formulate conclusions concerning its applicability in deep learning. The efficiency of Autoencoder based pretraining was evaluated by Erhan et al. in [3, 9].

Another problem with Deep Neural Network is the vanishing gradient problem. That is when the number of layers increase, the gradient in the lower layers during the backpropagation phase get so small that the network can no longer be efficiently trained. And this is also an important reason why the unsupervised pretraining is used: to lead the network weight to such a point fwim which the gradient-based training will be possible. According to the experiments we conducted, the backpropagation algorithm cannot train the network of more than about six hidden layers. The Levendberg-Marquardt (LM) algorithm and also non-gradient based methods could train in our tests even networks with 10 hidden layers. However, all they turn quite impractical for big networks and big datasets, because their complexity grows approximately quadratically with the number of training vectors and weights in the network.

## 2. Contrastive Divergence and Restriced Boltzmann Machines

Restricted Boltzmann Machines (RBM) belong to stochastic neural network, where there is some randomness incorporated in the neurons responses. An RBM consists of a layer of visible neurons and a layer of hidden neurons. The layers are connected in the same way as in a multilayer perceptron network, however the connections are bidirectional and symmetric and there are no connections between neurons of the same layer (formally RBMs create a bipartite graph). This feature is used in the Contrastive Divergence algorithm, where during the learning the signals get propagated in both directions. Deep belief networks are formed by stacking RBMs. First the RBMs are trained one by one with CD and only the last layer of the network is trained with a supervised algorithm, as Resilient Backpropagation (Rprop) [10]. After the initial unsupervised pretraining, the network is learned with standard learning algorithms used for Multilayer Perceptron (MLP) networks.

Contrastive Divergence (CD) is a method for estimating parameters of Markov Random Fields (MRFs) [11]. The approximation is performed by approximating the gradient of the log-likelihood based on a short Markov chain. In the field of neural networks it was applied and then popularized by Hinton [6] as a method of unsupervised training of Restricted Boltzmann Machines. Although CD performs quite well in empirical studies, it is little known about its theoretical convergence properties.

Contrastive divergence can be implemented in the following way: let $W$ be the matrix of weights of the dimension $I \times J$, where $I$ is the number of visible neurons and $J$ the number of hidden neurons. The state of each hidden neuron $S_j$ is formed by multiplication of each input by weight, summation over all inputs and application of this sum as an argument of nonlinear sigmoidal function:

$$X_j = \sum_i (S_i W_{ij} + R); S_j = (1 + exp(-A * X_j))^{-1} \tag{1}$$

where R is random number from normal distribution with zero mean. The general procedure of Contrastive Divergence is described in Algorithm 1. $CD_p$ and $CD_n$ are matrices. To apply the algorithm one needs to specify $\alpha$. This hyperparameter may potentially influence the convergence of the training procedure. To overcome the problem of choosing proper learning rate, often adapting schemes are used, not only for the CD but also for denoising autoencoders [12]. However, in our study, we decided to use the value of $\alpha$ chosen empirically. In Table 7 one may notice that for exemplary training task, performances obtained for several values of the learning rate were rather similar, so in this case, the method is not very sensitive to the changes of $\alpha$ in a reasonable (known from the previous research [6]) range.

**Algorithm 1** Contrastive Divergence

```
for $eh = 1 \ldots numEpochs$ do
  for $v = 1 \ldots numVectors$ do
    use vector V to set the states of visible neurons
    for $n = 0 \ldots numSteps$ do
      calculate the response of hidden neurons (with eq.1) and if n=0 calculate $S_i S_{j0}$
      use the calculated values to reconstruct the state of visible neurons
      $CD_p = CD_p + S_i S_{j0}/N$
      $CD_n = CD_n - S_i S_{j0}/N$
    end for
    update network weights $W = W + \alpha(CD_p - CD_n) + momentum \cdot W$
    if the signals $S_i$ changed between two epochs less than some threshold then terminate the training
  end for
end for
```

## 3. The Scope of the Study

The purpose of the study was to evaluate how the theory of unsupervised pretraining presented in the introduction applies in practice by evaluating the usefulness of unsupervised pretraining with Contrastive Divergence (CD), depending oh the percentage of the labeled data we have access to, the network architecture and the amount of layers that are pretrained with CD. When all the hidden layers have been pretrained, then we additionally train the output layer with Rprop [10], before training the whole network with Rprop. Finally the whole network is trained (fine-tuned with Rprop). The pretraining sometimes improved the results (mostly in the case of higher number of hidden layers and in the case, where the labels were not available for each training vector) and sometimes it worsen them. For the cases where the results with and without pretraining were similar we propose to pretrain only the first hidden layer. The experimental results confirmed this to be a good solution.

# 4. Experimental Evaluation

## 4.1. Datasets

It is reasonable to use deep architectures if the problem is difficult, so we run the tests on three handwritten digit recognition tasks: the Pen-Based Recognition of Handwritten Digits Data Set (Penbased, 16 attributes, 10 classes), the Optical Recognition of Handwritten Digits Data Set (Optdigits, 1024 attributes, 10 classes), and part of the MNIST Data Set (only the part of the set usually used for testing was used, 666 attributes after reduction, 10 classes) [13]. Two benchmarks come from the UCI Repository [14], and the MNIST is a standard (and historical) benchmark to test deep architectures. The software used in the experiments is created in C#, and the Accord.NET and AForge.NET libraries are used. The datasets and the software can be obtained from [15].

## 4.2. Experiments

In the experiments we evaluated how the unsupervised pretraining improves the classification results depending on several factors. The considered factors were:

1. The properties of the data. We used three different datasets, with 10 classes and number of attributes in the wide range, described in section 4.1. The origin of the benchmarks is digits recognition, so we may assume that the problem is well-defined and the classes mostly do not overlap.
2. The network training method. The tested network were trained with or without unsupervised pretraining. These two learning phases were performed on different amounts of available training data.
3. The architecture. We decided to test deep architectures consisting of up to six hidden layers, as well as shallow nets with one hidden layer. Several sizes of hidden layers were also tested.
4. The amount of available labeled data vs. the amount of unlabeled data. Several combinations of percentages of data taken from labeled and unlabeled training set were examined.

We tested Deep Belief Networks consisting of stacked Restricted Boltzmann Machines trained with Contrastive Divergence, with $\alpha = 0.1$ (although we tried $\alpha$ from 0.05 to 0.4 – see Table 7), $momentum = 0.5$ and 300 epochs. The top layer was trained with Resilient Backpropagation (Rprop) with 150 epochs. Then the whole network was further trained in the supervised mode also with Rprop.

The vectors used for unsupervised and supervised training and for tests were selected from the original datasets in the following way: first the vectors were randomly permutated. Then 25% of vectors were placed in the test set and 75% in the training set. Some of the vectors from the training set were used for unsupervised training (or no unsupervised training was performed). Then the labels were removed from the part of the training data, and the remaining 75%, 25%, 8% or 3% of the vectors were used for supervised training.

In the tables, the number after U represents the percentage of the training set used for unsupervised learning (L is used instead of U in the cases where only the first layer was trained with CD) and the number after S represents the percentage of the training set used for supervised learning. For instance: S25-U75-T25 means that unsupervised learning was performed on a stack of RBMs, trained with Contrastive Divergence on 75% of the whole dataset (100% of the training data), and the top layer was trained with Rprop on 25% of the data and after that the whole network was trained with Rprop on the same 25% of the data. Finally the network was tested on the test data (another 25% of the vectors) and the classification accuracy obtained on the test data was reported in the tables. The first row of each table describes the networks' architectures, i.e. the numbers of hidden neurons (not including input and output layers). Because of limited space in the article we do not present dispersion of the results in the tables. Such dispersions were in the range $1\% - 1.5\%$, and 4 outliers (when a network didn't learn) were removed out of hundreds of simulations.

**Table 1.** Classification results for Penbased dataset with bigger networks.

| Met./Net Size | 100 | 100-100 | 100-100- | 100-100- | 100-100- | 100-100- |
|---|---|---|---|---|---|---|
| | | | -100 | -100-30 | -100-100 | 100-100-30 |
| S75-U75-T25 | 94.76 | 97.76 | 94.40 | 98.51 | 98.62 | 97.78 |
| S75-T25 | 94.76 | 98.73 | 98.44 | 98.36 | 97.85 | 96.04 |
| S25-U75-T25 | 94.25 | 94.93 | 97.60 | 97.49 | 97.42 | 94.03 |
| S25-U25-T25 | 94.47 | 97.96 | 94.14 | 97.34 | 97.27 | 94.29 |
| S25-T25 | 94.10 | 94.69 | 96.54 | 96.07 | 97.78 | 95.78 |
| S08-U75-T25 | 93.52 | 93.20 | 92.83 | 93.34 | 96.12 | 92.03 |
| S08-U25-T25 | 93.20 | 93.89 | 92.83 | 93.78 | 95.16 | 91.85 |
| S08-U08-T25 | 93.45 | 93.27 | 93.60 | 95.45 | 94.98 | 87.81 |
| S08-T25 | 93.27 | 93.56 | 92.72 | 94.98 | 94.07 | 89.48 |
| S03-U75-T25 | 90.76 | 93.81 | 92.14 | 90.98 | 91.16 | 84.53 |
| S03-U25-T25 | 93.23 | 93.34 | 93.01 | 93.05 | 94.94 | 90.03 |
| S03-U08-T25 | 90.94 | 89.70 | 89.52 | 88.39 | 89.85 | 86.11 |
| S03-U03-T25 | 91.05 | 89.37 | 92.47 | 87.66 | 89.99 | 84.25 |
| S03-T25 | 91.08 | 89.77 | 88.25 | 89.56 | 88.76 | 84.53 |

### 4.3. Results

Looking at the Table 2 for Penbased benchmark one may notice that the best performance without pretraining was achieved by 3-layer network. 2-layer and 4-layer networks also perform better than regular shallow network. However, a network having 6 hidden layers presents the lowest classification accuracy. If we consider networks pretrained with the CD, it is clear that this methods only slightly improves the performance for deep structures. Moreover, when the unsupervised phase is performed on smaller training set it may even worsen the results. In the Table 1 we may observe that deep networks with more hidden units achieve similar performance. This time two-layer network is slightly outperformed by nets with 4 hidden layers. However, the overall performance doesn't seem to depend on the last hidden layer size. In this case, the results obtained for 30 (potential bottleneck) and 100 neurons in the last hidden layer are very similar.

**Table 2.** Classification results for Penbased dataset.

| Meth./Net Size | 60 | 60-45 | 60-45-30 | 60-45-30-25 | 60-45-30-25-20-15 |
|---|---|---|---|---|---|
| S75-U75-T25 | 94.51 | 98.33 | 98.07 | 98.29 | 95.71 |
| S75-T25 | 94.61 | 98.33 | 98.65 | 97.27 | 89.30 |
| S75-L75-T25 | 94.65 | 95.05 | 97.85 | 98.47 | 94.72 |
| S25-U75-T25 | 94.07 | 93.41 | 97.56 | 96.11 | 88.90 |
| S25-U25-T25 | 94.18 | 94.10 | 97.56 | 96.03 | 91.81 |
| S25-T25 | 94.14 | 94.25 | 97.49 | 95.85 | 80.24 |
| S25-L75-T25 | 94.18 | 94.00 | 97.71 | 94.65 | 88.76 |
| S25-L25-T25 | 94.14 | 94.21 | 93.56 | 96.91 | 93.41 |
| S08-U75-T25 | 93.09 | 93.41 | 94.10 | 94.00 | 87.01 |
| S08-U25-T25 | 93.12 | 96.32 | 91.59 | 92.65 | 88.61 |
| S08-U08-T25 | 93.12 | 96.98 | 96.14 | 94.58 | 89.12 |
| S08-T25 | 93.01 | 93.12 | 94.65 | 93.67 | 88.46 |
| S08-L75-T25 | 93.30 | 92.65 | 94.51 | 91.19 | 87.70 |
| S08-L25-T25 | 92.83 | 92.79 | 91.56 | 90.14 | 87.19 |
| S08-L08-T25 | 92.87 | 92.58 | 95.38 | 93.49 | 84.68 |
| S03-U75-T25 | 91.08 | 88.54 | 91.30 | 88.32 | 75.87 |
| S03-U25-T25 | 92.83 | 92.61 | 95.41 | 92.54 | 82.28 |
| S03-U08-T25 | 90.14 | 89.59 | 87.74 | 84.75 | 68.34 |
| S03-U03-T25 | 90.65 | 90.25 | 90.07 | 86.24 | 71.11 |
| S03-T25 | 89.88 | 89.05 | 90.10 | 87.23 | 73.44 |
| S03-L75-T25 | 89.59 | 90.32 | 87.45 | 86.64 | 73.69 |
| S03-L25-T25 | 92.69 | 93.52 | 94.65 | 93.78 | 84.43 |

For the Optdigits data, the best performance for the purely supervised algorithm was obtained for the network with 3 hidden layers (Table 3). When the network is trained only on the part of labeled examples, the unsupervised phase increases its performance for deeper network architectures. If we consider bigger networks with more hidden units (Table 4), the performances for networks with or without pretraining are better than in the latter case. The best results are achieved for two and three hidden layers. For such networks, using unsupervised phase increasing the classification rates significantly when only small amount of labeled data is used (S03).

**Table 3.** Classification results for Optdigits dataset.

| Meth./Net Size | 300 | 300-150 | 300-150-50 | 300-150-50-30 | 300-150-50-<br>-30-20-15 |
|---|---|---|---|---|---|
| S75-U75-T25 | 91.3 | 90.91 | 92.56 | 92.36 | 90.02 |
| S75-T25 | 91.32 | 91.74 | 92.91 | 90.91 | 88.15 |
| S25-U75-T25 | 78.72 | 86.78 | 91.77 | 89.10 | 76.45 |
| S25-U25-T25 | 76.24 | 86.16 | 89.19 | 84.27 | 74.75 |
| S25-T25 | 80.17 | 89.46 | 90.09 | 87.10 | 82.34 |
| S08-U75-T25 | 71.90 | 81.20 | 85.86 | 83.43 | 68.87 |
| S08-U25-T25 | 66.74 | 79.93 | 81.26 | 79.28 | 64.86 |
| S08-U08-T25 | 64.67 | 76.03 | 76.03 | 70.01 | 63.92 |
| S08-T25 | 74.17 | 81.82 | 84.15 | 81.22 | 76.60 |
| S03-U75-T25 | 49.17 | 58.06 | 63.02 | 53.93 | 48.07 |
| S03-U25-T25 | 54.96 | 78.95 | 79.99 | 76.62 | 55.01 |
| S03-U08-T25 | 58.72 | 66.54 | 68.11 | 65.55 | 57.93 |
| S03-U03-T25 | 52.69 | 60.95 | 63.22 | 58.47 | 52.56 |
| S03-T25 | 55.58 | 55.17 | 62.19 | 51.86 | 49.04 |

**Table 4.** Classification results for Optdigits dataset with bigger networks.

| Meth./Net Size | 1000 | 1000-700 | 1000-700-100 | 1000-700-<br>-100-30 | 1000-700-500-<br>-250-100-30 |
|---|---|---|---|---|---|
| S75-U75-T25 | 94.21 | 96.07 | 94.83 | 95.45 | 88.84 |
| S75-T25 | 94.42 | 95.66 | 96.07 | 95.45 | 89.05 |
| S25-U75-T25 | 88.43 | 92.56 | 91.94 | 91.12 | 82.64 |
| S25-U25-T25 | 91.32 | 92.15 | 91.94 | 86.98 | 83.26 |
| S25-T25 | 86.16 | 92.98 | 93.60 | 90.50 | 70.87 |
| S08-U75-T25 | 83.88 | 87.81 | 86.98 | 81.61 | 76.86 |
| S08-U25-T25 | 84.09 | 85.74 | 88.22 | 82.23 | 46.49 |
| S08-U08-T25 | 85.12 | 89.46 | 85.33 | 81.20 | 70.04 |
| S08-T25 | 84.09 | 86.98 | 85.54 | 83.26 | 71.07 |
| S03-U75-T25 | 67.77 | 70.87 | 71.28 | 62.60 | 38.64 |
| S03-U25-T25 | 81.82 | 85.54 | 85.54 | 80.99 | 69.83 |
| S03-U08-T25 | 65.50 | 75.00 | 71.07 | 59.50 | 49.86 |
| S03-U03-T25 | 69.63 | 69.83 | 65.91 | 50.62 | 46.12 |
| S03-T25 | 65.70 | 71.90 | 69.21 | 70.04 | 43.39 |

Similar results were obtained for a part of the MNIST dataset (Table 5). For bigger networks, unsupervised pretraining phase helps in obtaining higher accuracy. Better performance is obtained for the nets with more neurons in the hidden layers (Table 6). Highest classification rates are achieved by the architectures with 3 or 4 hidden layers.

**Table 5.** Classification results for MNIST dataset.

| Meth./Net Size | 300 | 300-150 | 300-150-50 | 300-150-50-30 | 300-150-50--30-20-15 |
|---|---|---|---|---|---|
| S75-U75-T25 | 92.78 | 92.91 | 93.16 | 90.34 | 87.47 |
| S75-T25 | 93.52 | 93.98 | 94.25 | 93.08 | 91.33 |
| S25-U75-T25 | 87.48 | 88.22 | 90.02 | 88.95 | 87.76 |
| S25-U25-T25 | 87.45 | 89.50 | 90.02 | 87.51 | 85.12 |
| S25-T25 | 93.56 | 93.95 | 93.80 | 91.68 | 89.16 |
| S08-U75-T25 | 86.95 | 87.07 | 87.12 | 86.03 | 86.72 |
| S08-U25-T25 | 80.88 | 80.84 | 81.25 | 80.54 | 75.92 |
| S08-U08-T25 | 81.86 | 81.34 | 81.89 | 80.25 | 78.12 |
| S08-T25 | 92.80 | 93.07 | 93.87 | 92.27 | 90.00 |
| S03-U75-T25 | 87.02 | 88.15 | 89.05 | 86.41 | 85.50 |
| S03-U25-T25 | 79.49 | 79.86 | 80.85 | 78.66 | 76.80 |
| S03-U08-T25 | 81.32 | 80.88 | 81.16 | 79.67 | 78.14 |
| S03-U03-T25 | 79.16 | 79.60 | 79.48 | 78.72 | 74.89 |
| S03-T25 | 92.60 | 93.11 | 93.80 | 92.16 | 88.55 |

**Table 6.** Classification results for MNIST dataset with bigger networks.

| Meth./Net Size | 1000 | 1000-700 | 1000-700-500 | 1000-700--500-200 | 1000-700-500--400-300-100 |
|---|---|---|---|---|---|
| S75-U75-T25 | 94.14 | 94.99 | 95.22 | 95.01 | 92.22 |
| S75-T25 | 94.16 | 95.05 | 95.08 | 94.73 | 90.92 |
| S25-U75-T25 | 92.99 | 93.35 | 93.87 | 92.48 | 90.05 |
| S25-U25-T25 | 93.44 | 93.97 | 94.01 | 93.23 | 88.98 |
| S25-T25 | 93.87 | 93.70 | 94.90 | 93.22 | 89.22 |
| S08-U75-T25 | 88.16 | 88.86 | 89.82 | 89.10 | 88.14 |
| S08-U25-T25 | 87.03 | 87.45 | 87.55 | 86.94 | 86.70 |
| S08-U08-T25 | 88.11 | 88.12 | 89.13 | 88.06 | 86.22 |
| S08-T25 | 88.53 | 90.12 | 91.27 | 90.15 | 86.49 |
| S03-U75-T25 | 81.95 | 81.88 | 81.55 | 88.55 | 76.16 |
| S03-U25-T25 | 80.51 | 79.88 | 80.88 | 79.36 | 71.15 |
| S03-U08-T25 | 82.16 | 82.33 | 82.17 | 81.18 | 79.15 |
| S03-U03-T25 | 82.45 | 81.60 | 81.48 | 80.44 | 78.12 |
| S03-T25 | 82.09 | 80.64 | 80.92 | 81.83 | 76.90 |

## 5. Conclusions

Based on the experiments we can conclude that the unsupervised pretraining with Contrastive Divergence usually helps in big, deeper networks, with many hidden layers, and does not improve or even worsen the results in shallow architectures. This

**Table 7.** Classification rates for different $\alpha$ (Penbased dataset, hidden layers: 100-100-100-100-100-30).

| Meth./$\alpha$ | 0.05 | 0.1 | 0.2 | 0.4 |
|---|---|---|---|---|
| S75-U75-T25 | 97.45 | 96.91 | 97.05 | 96.72 |
| S25-U75-T25 | 94.80 | 94.54 | 95.34 | 93.20 |
| S25-U25-T25 | 95.20 | 94.83 | 93.70 | 86.90 |
| S08-U75-T25 | 91.30 | 90.14 | 91.52 | 90.17 |
| S08-U25-T25 | 87.99 | 90.05 | 92.25 | 92.72 |
| S08-U08-T25 | 90.25 | 91.74 | 89.88 | 90.32 |
| S03-U75-T25 | 80.71 | 80.97 | 82.06 | 76.82 |
| S03-U25-T25 | 82.17 | 90.65 | 90.90 | 90.21 |
| S03-U08-T25 | 84.10 | 75.25 | 77.80 | 81.73 |
| S03-U03-T25 | 82.79 | 82.39 | 77.26 | 76.38 |

is not because Contrastive Divergence did not work. Just on the contrary: it worked very well and reduced the error an order of magnitude or more. The cause was rather that Rprop worked exceptionally well and in many cases was able to reach the good solution even without any pretraining. When the training set consists of labeled and unlabeled examples, then it is reasonable to apply unsupervised phase to build the representation based on the whole data available. We propose to make the degree of the unsupervised training dependant on the network size. The bigger and deeper the network, the pretraining should be stronger, for moderate depth networks it may comprise only the first or first and second hidden layer. For the shallower networks, with one or two hidden layers, pretraining is not recommended, even if there are missing labels in the part of the data. We are planning to perform much more experiments to formulate more detailed conclusions.

## 6. References

[1] Bengio Y., Lamblin P., Popovici D., Larochelle H., et al., *Greedy layer-wise training of deep networks.* Advances in neural information processing systems, 2007, 19, pp. 153.

[2] Salakhutdinov R., Hinton G., Semantic hashing. In: *Proceedings of the 2007 Workshop on Information Retrieval and applications of Graphical Models (SIGIR 2007)*, 2007.

[3] Erhan D., Bengio Y., Courville A., Manzagol P.A., Vincent P., Bengio S., *Why does unsupervised pre-training help deep learning?* The Journal of Machine Learning Research, 2010, 11, pp. 625–660.

[4] Vincent P., Larochelle H., Lajoie I., Bengio Y., Manzagol P.A., *Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion.* The Journal of Machine Learning Research, 2010, 11, pp. 3371–3408.

[5] Bengio Y., *Learning deep architectures for ai.* Foundations and trends® in Machine Learning, 2009, 2(1), pp. 1–127.

[6] Carreira-Perpinan M.A., Hinton G., On contrastive divergence learning. In: *AISTATS.* vol. 10., Citeseer, 2005, pp. 33–40.

[7] Hinton G.E., Salakhutdinov R.R., A better way to pretrain deep boltzmann machines. In: *Advances in Neural Information Processing Systems*, 2012, pp. 2447–2455.

[8] Tieleman T., Hinton G., Using fast weights to improve persistent contrastive divergence. In: *Proceedings of the 26th Annual International Conference on Machine Learning*, ACM, 2009, pp. 1033–1040.

[9] Erhan D., Manzagol P.A., Bengio Y., Bengio S., Vincent P., The difficulty of training deep architectures and the effect of unsupervised pre-training. In: *International Conference on artificial intelligence and statistics*, 2009, pp. 153–160.

[10] Riedmiller M., Braun H., A direct adaptive method for faster backpropagation learning: The RPROP algorithm. In: *Neural Networks, 1993., IEEE International Conference on*, IEEE, 1993, pp. 586–591.

[11] Sutskever I., Tieleman T., On the convergence properties of contrastive divergence. In: *International Conference on Artificial Intelligence and Statistics*, 2010, pp. 789–795.

[12] Geras K.J., Sutton C., *Scheduled denoising autoencoders.* arXiv preprint arXiv:1406.3269, 2014.

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

[14] Blake C., Merz C.J., {*UCI*} *repository of machine learning databases*, 1998.

[15] *Software and datasets used in the paper.* http://www.kordos.com/datasets, Accessed: 2014-12-30.