A Comparative Evaluation of Deep Belief Nets in Semi-supervised Learning

Huixuan Tang
Department of Computer Science
University of Toronto
40 St. George Street, M5S 2E4, Toronto, Canada.
hxtang@dgp.toronto.edu

Abstract

In this report I studied the performance of deep belief nets (DBNs) on semi-supervised learning problems, in which only a small proportion of data are labeled. First the performance between DBNs and support vector machines (SVMs) are compared to investigate the advantage of deep models over shallow ones. I also explored the use of DBNs as pre-training for SVMs and feed-forward nets (FFNs). The experimental results show that DBN is able to yield state-of-art modeling power in semi-supervised learning.

1 Introduction

Semi-supervised learning is an important paradigm of machine learning as in reality labeled data are usually expensive to acquire. The inexpensive unlabeled data are believed to help model the marginal distribution of data $\Pr(x)$. Many algorithms are now motivated to exploit a large unlabeled training set to facilitate supervised learning.

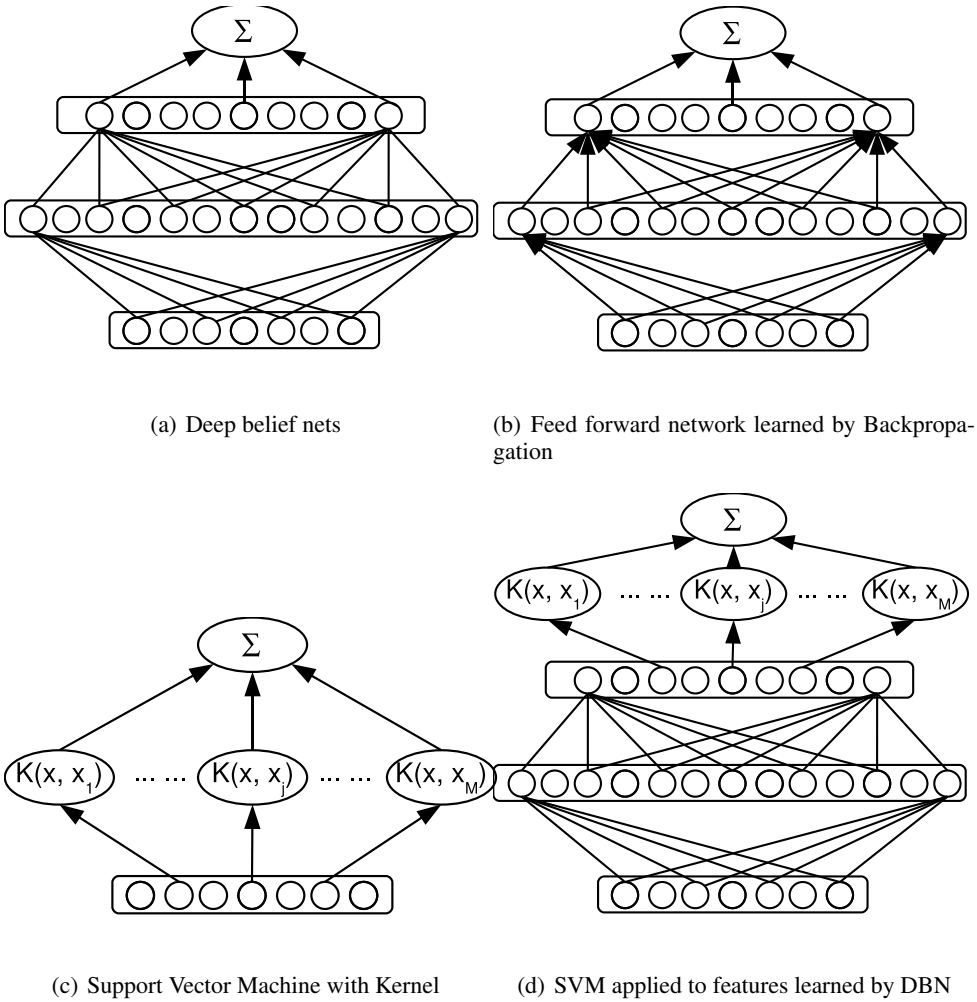
Deep belief nets [1] is promising regarding this task since it exhibits an impressive power to learn the encoding of a complex distribution in an unsupervised manner. The greedy pretraining of DBN results in a deep structure of Markov random field (MRF), which could be considered a Helmholtz machine. Its recognition weights could be fine-tuned with a soft-max layer added to the top for recognition tasks. Since the unsupervised learning has already provided a reasonable initial weight for the network, a small amount of labeled data would be sufficient for fine-tuning the network. Its generative weights could be employed to generate samples and fine-tuned with reconstruction error. In this report, only the first part is explored.

Although deep belief nets seems ideal for semi-supervised learning, it is unknown how well it works in comparison with other state-of-art models. In this work I investigate the performance of DBNs, SVMs [2], SVMs working on the features learned by DBNs, and FFNs learned with backpropagation (BP) algorithm on semi-supervised tasks. All these methods are evaluated on the MNIST[1] data set, each of which is trained with a varying proportion of labeled data.

It is most interesting to compare DBN with the SVM because the latter is well known for its capability to cope with sparse data of high dimensions. SVM, as a form of a perceptron, is of special interest because it may be the most competitive shallow model *status quo* for classification tasks. Also, because SVM is so good, it is interesting to see how much DBN can improve its classification performance on a small proportion of labels.

Another interesting view of DBNs is to see it as a recurrent network trained in a greedy layer-wise manner. Therefore, a comparison with an FFN learned with BP reveals the advantage of greedy

[1] http://yann.lecun.com/exdb/mnist



(a) Deep belief nets

(b) Feed forward network learned by Backpropagation

(c) Support Vector Machine with Kernel

(d) SVM applied to features learned by DBN

Figure 1: Architectures of Evaluated Models

pre-training strategy. This also shows the usefulness of a good initial weights learned by DBN in training a FFN with BP for classification tasks, as BP, though capable of training deep structures, will easily be trapped in local minima without a reasonable initial value.

2 Related Works

2.1 Learning Algorithms

Fig. 1 shows four architectures evaluated in this work. Kernelized SVM is a radial basis function (RBF) network that has one layer of weights to be determined, while DBN and FFN can have multiple hidden layers. Therefore, SVM is referred to as a "shallow" model, while DBN and FFN as those of deep architectures. A deep structure is believed to be able to capture more regularity of data by extracting "low-level" features with the bottom hidden layers, as well as "abstract" features with the top hidden layers. If DBN actually extracts the desired features, the performace of SVM in (d) might still be improved w.r.t. the one in (c).

The four models not only differ in architecture, but also in the way they are learned. A DBN could be viewed as a stack of restricted Boltzman machines (RBM) [3], which are motivated from the idea

of equilibrium from the statistical physics literature:

$$E(v,h) = -\sum_{i,j} w_{i,j} v_i h_j - \sum_i v_i b_i^v - \sum_j h_j b_j^h. \tag{1}$$

where v are the visible units (input) and h are the hidden units. Eq. (1) could be optimized in a tricky way by contrastive divergence [3].

A DBN could then be trained in a greedy layer-wise manner as in [1]:

- Train an RBM from input space to the first hidden layer with all other weight matrices tied.
- Freeze W_0 in the first RBM and generate the factorial posterior distributions of the hidden layer.
- Use the posterior distribution as the input distribution of a second RBM. Train the RBM likewise recursively.

This generative model could be extended for classification tasks by adding a soft-max layer on top of the architecture, and then fine-tuned with BP, w.r.t. cross entropy loss:

- Freeze all layers in the DBN and learn the weights for the top layer. (This is because the added layer is not as mature as other layers)
- Fine-tune the weights for all layers.

The objective function of fine-tuning is to minimize the negative likelihood of labels:

$$L(W) = -\sum_i \sum_j t_{i,j} \log P(t_{i,j} = 1 \mid W, x_i), \quad t_{i,j} = 1 \text{ when } x_i \text{ is from class } j, \tag{2}$$

which could be learned by BP. In the current implementation, this is done by a number of conjugate gradient (CG) searches.

An FFN is also learned by optimizing Eq. (2), but it starts from a random configuration and in each iteration all weights are simultaneously updated.

In comparison to Eq. (2) and Eq. (1), the objective function of support vector machines smartly avoids overfitting by maximizing the margins:

$$L(w) = \min \frac{1}{2}\|w\|^2 + C\sum_i \xi_i, \tag{3}$$

subject to

$$t_i(w^\top x_i + b) \geq 1 - \xi_i, \qquad i = 1, \ldots, N, \tag{4}$$

where $t_i = \pm 1$ for binary classification.

As the weight vector could be expressed as the linear combination of support vectors at the optimal solution, SVMs could perform non-linear classification by replacing every inner product with non-linear kernel functions (kernel tricks). This indicates that SVMs might have the potential to compete with models of deep architecture when given appropriate kernels and kernel parameters.

2.2 Previous Empirical Evaluation of Deep Belief Nets

DBN has been tested in comparison to shallow models including RBM, SVM, and nearest neighbors as soon as it was proposed ([4, 1]). Further, Bengio performed experiments on a various versions of DBNs in [5] to support the view that the unsupervised layer-wise learning is essential in optimizing DBNs, and some supervised learning can help improve the model. In [6], Larochelle and his wo-workers compared DBN with shallow networks, including SVM, RBM and shallow FFN on a collection of datasets manipulated from the MNIST dataset.

My work is closely related with the above works as it concerns how much DBN facilitates other models. However, it has the unique contribution of concerning the change of performance of the various models as a function of portion of labeled data, i.e. in a semi-supervised settings.

3 Experiments

3.1 Benchmark

The same data set as in [1], the MNIST data set is employed in our experiments to evaluate the four architectures. The original data set consists of 70,000 handwritten digits, 10,000 for testing and 60,000 for training and validation.

The 60,000 samples are divided into two parts: 50,000 samples are used for unsupervised training and 10,000 samples for supervised training and validation, in order to avoid the overlap between the training dataset and the validation dataset. In the evaluation process I tried to use 100, 200, 500, 1000, 2000, 5000, and 10000 labeled data to train the supervised part of each model.

I intentionally keep the proportion of labeled data much smaller than the unlabeled part, since the concern of semi-supervised learning is to cope with the situation when label is expensive. Also, a very small labeled trainset will fail supervised learning models and thus make comparison easier.

3.2 Implementation Details

I used the implementation[2]provided by the authors of [1] (modified a little so as to allow learning arbitrary layers of DBNs), which means mean-field approximation are used to approximate sampled hidden states, and conjugate gradients search is used to fine-tune the weights. The same implementation was used to learn the weights of an FFN, though no pre-training was performed to initialize the weights. The learning rate, momentum, weight cost and initial weight distributions are just the same as the original code.

The implementation of SVM is SVMStruct[3] (in particular, SVM-multiclass), and all SVMs are trained with only a proportion of the 10,000 labeled from the training set. It might be more appropriate to evaluate transductive SVM on the dataset as a comparison, but my work does not cover this part because of time limitations and the inefficiency of transductive SVM on large datasets.

For SVM applied to DBN features, all labeled data are transformed into the feature space learned by DBN and then are used to train SVM. It is interesting to see the performance of SVM combined with DBN with fine-tuning, but it is not covered either since fine-tuning with SVM is non-trivial.

3.3 Model Selection

For DBNs and FFNs, the number of hidden layers and number of hidden units for each layer have to determined. Also, the maximal number of epochs either to train RBMs or of BP should also be determined. For SVM the type of kernel, its parameters and the regularization parameter C in (3) are also required to be determined.

A 10-fold cross validation is employed on a training set of 1,000 labeled data for model selection. It is observed that 1,000 samples are enough to provide a relatively reliable estimation of testing error and are not too many to become fully supervised. I find both error rate or cross entropy loss (average negative log likelihood of correct labels) on the validation set are sensitive criterion to use. (Note that reconstruction error does not apply here as it tells nothing about how discriminative the representation is.) To calculate this criterion, a soft-max layer with 10 output units is first added to the top of each unsupervisedly trained DBN and later learned with BP. The output of each unit is just the probability of assigning the corresponding label. Since we aim at selecting the model for unsupervised learning, no fine-tuning of the rest layers is performed.

With the number of hidden units fixed, the optimal number of epochs can be selected by taking the epoch with minimal cross entropy loss. I believe this optimal epoch is independent of the structure of network (say, number of layers and number of hidden units), and therefore select this option by looking at the best epoch of a fixed structure (one hidden layer of 500 hidden variables). The resulting curve of both criterions are shown in Fig. 2, and the two curves appear to provide consistent evaluation on the number of epochs. Surprisingly, no overfitting is observed with respect to number of epochs. This is because the model is trained on a very large training set, and the model does

[2] `http://www.cs.toronto.edu/~hinton/MatlabForSciencePaper.html`

[3] `http://svmlight.joachims.org/`

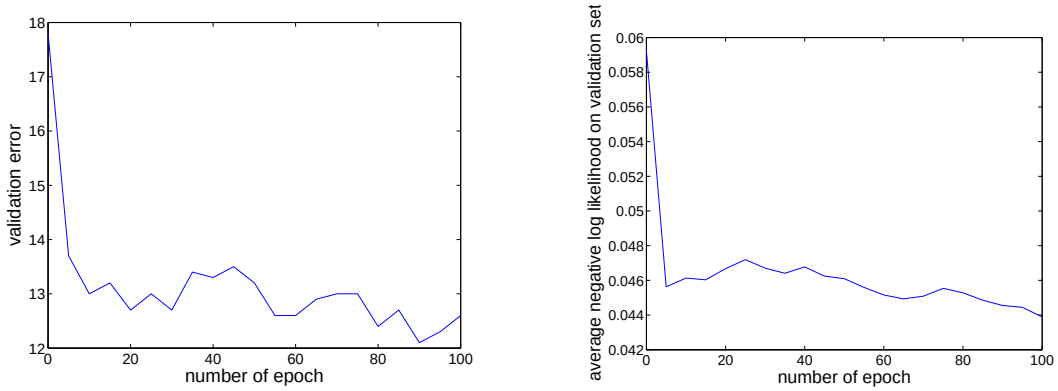


Figure 2: Validation error (left) and cross entropy loss (right) on validation set as a function of epochs. The model is trained on a DBN with 1 layer of 500 units by 10-fold cross validation.

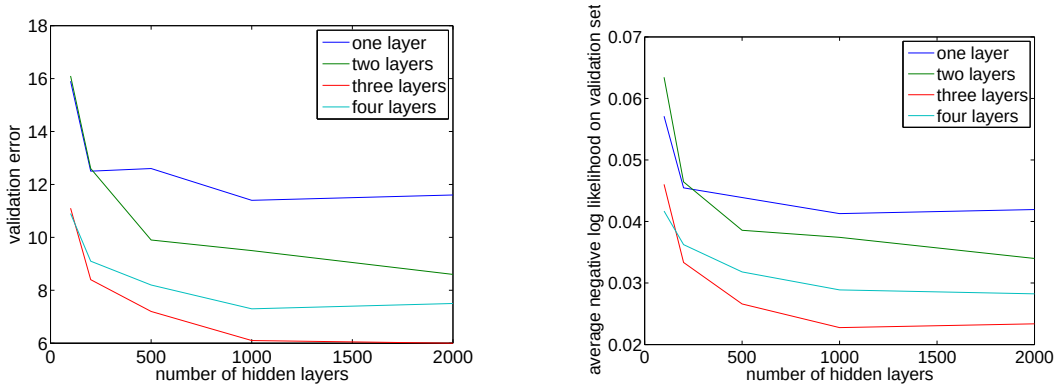


Figure 3: Validation error (left) and cross entropy loss (right) on validation set as a function of number of the top hidden layer. The model is trained on a DBN with 1 layer of 500 units by 10-fold cross validation. The other hidden layers are fixed to the the optimal ones determined with less number of layers.

not overfit easily. However, both criterion stop improving as the number of epochs becomes large, therefore any epoch in the approximately constant area (from 10 to 100) is appropriate. The maximal epoch is selected to be 30.

With fixed number of hidden layers, the ideal way of deciding the number of hidden units for each layer is grid-search. However, grid-search is too time-consuming to afford in this project and therefore a greedy approach to search for the best hidden layers is adopted instead: freeze the optimal number of hidden layers for the bottom layers and only search on the last layer. The resulting validation errors is shown in Fig. 3. There is no obvious overfitting as the number of hidden neurons increases, and I choose the number with which the validation error stops decreasing significantly. Fig. 3 also contrasts the validation performances w.r.t. number of layers, and it is shown that adding the fourth layer would not further improve the performance. Therefore, the structure of the DBN is finalized to be a 784-1000-500-1000 network.

My results is different from the optimal structure in [1], possibly due to two reasons: firstly, grid search is performed in those works which means the model architecture is more carefully selected; secondly, the training set I use is significantly different with the ones used in [1], namely with different numbers of labeled training cases. I have also tried the 500-500-2000 architecture which was reported to be optimal in [1]: its validation error indeed is about 1% lower than the 1000-500-1000 architecture.

The same architecture is chosen to generate features to for SVMs. In my opinion this is reasonable since the goodness of DBN is independent of the classifier used.

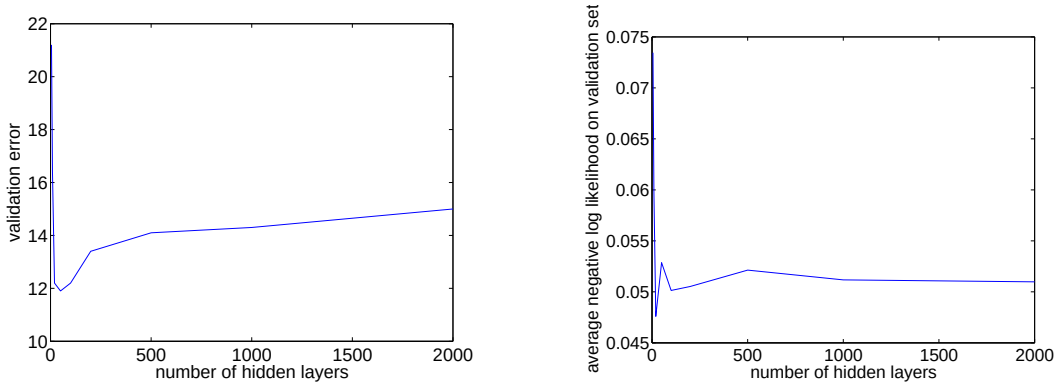


Figure 4: Validation error (left) and average negative Log likelihood (right) on validation set as a function of number of the hidden units.

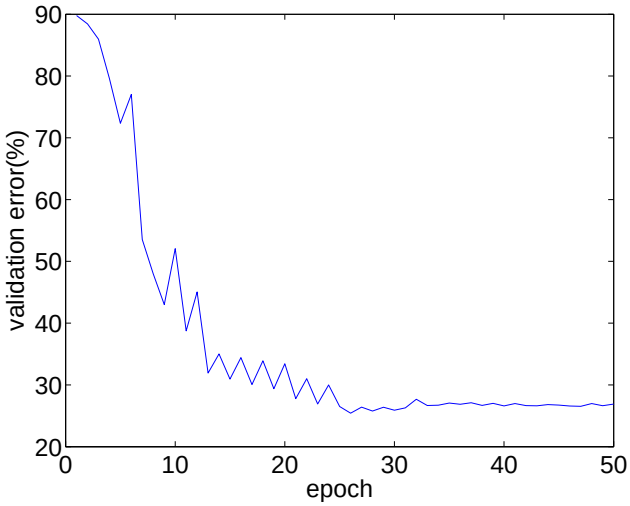


Figure 5: Validation error (left) as function of time in training FFNs with BP. The FFN has the same architecture as the optimal DBN in this project.

Since it is slow to train kernelized SVM on large data sets, no fine tuning of parameters for SVM is done. Instead, a number of parameter settings for four types of kernels (linear, rbf, polynomial, and sigmoid) are tried and the best of them (the lowest test error) is chosen. In both cases, linear SVM and SVM with RBF kernels work better. For SVM alone, I observed that RBF kernel with $\gamma = 1$ works best and with features provided by the proposed DBN, linear kernel works best. SVM is not very sensitive to the weight C in (3) and I set $C = 1000$ for all the machines.

Fig. 4 shows the validation error of an FFN with a single hidden layer as the number of hidden units varies. Unlike DBN, FFNs quickly overfit as the number of hidden units increases, and layer-wise search would not work as is expected to find optimal structure for FFNs. Therefore a 784-1000-500-100-10 FFN is trained to compare with the fine-tuned DBN of the same architecture. This decision is not only due to the fact that grid search is computationally heavy, but also because this would be a fair comparison to see the contribution of pretraining. To avoid overfitting, the maximal epoch is set to the time when the validation error is smallest, as shown in Fig. 5.

3.4 Results

The best performance of the above methods on the test set is summarized in Fig. 6. The figure show that fine-tuned DBN outperforms all other methods significantly for all training set sizes. The

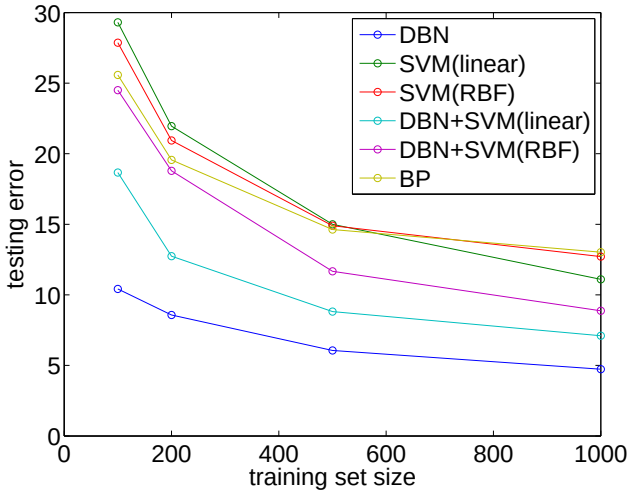


Figure 6: Testing error of the four models as a function of training set size. (Color image, please see electronic version of the report.)

second best method is linear SVM working on features learned by DBN. This is an evidence that the large set of unlabeled data provides much information of the distribution of data. SVM alone does not work well in the comparison, partly because of limited training data, partly because of a non-specific kernel. The significant improvement made by working on features provided by DBN shows that DBN is able to extract "good" features in a general way, which requires no expertise in the application. Therefore, SVM actually leaves the tough part to kernel designers while DBN provides a general way to find nice feature spaces.

It is interesting to see that DBN fine-tuned by BP outperforms SVM applied to DBN features, while FFN performs almost as well as SVM. This indicates fine-tuning may play a significant role even in supervised learning.

Another interesting issue is that linear SVM performs best in comparison to other kernelized SVM models when features provided by DBN are fed as input. This indicates that after mapping into the feature space, the classes becomes so well separated that linear classifier is sufficient.

4 Conclusion

The above results answer the questions mentioned at the begining of this report, in the context of semi-supervised learning, which are summarized in the following:

- A model of deep architecture (e.g. a 3-layer DBN) works much better than a shallow model (e.g. a single SVM or RBM) in generating a distributed encoding of dataset for further classification.
- Semi-supervised learning of such deep architecture makes the model much more superior than all supervised shallow models (e.g. SVM).
- Although SVM has a smart scheme for learning small sets of high dimensional data, features learned by DBN would still significantly improve the classification results.
- The nice initial weights obtained by unsupervised learning of DBN would obviously facilitate BP on deep models.

Although I believe this work has provided a meaningful insight in the use of DBN dealing with semi-supervised learning, it is limited in several ways due to time constraints and lack of computational resources, and should be improved later. First of all, model selection for both the deep belief nets and the feed forward nets can be conducted by grid search, and a more carefully selected model is expected to provide a more accurate evaluation of the methods. Secondly, the current comparison

is not sufficient to compare deep and shallow semi-supervised learning models since SVM learning is purely supervised. Further experiments on transductive SVM and other S^3VM algorithms is necessary for such comparisons. Third, the DBN+SVM model does not include fine tuning and makes comparison a bit unfair. It would be interesting to investigate a way to adapt the weights in DBN with regard to the SVM learning rule. Finally, since MNIST is a comparatively easy data set, this work does not provide much evidence on how well DBN would perform on harder data sets. It would be interesting to evaluate all the methods on such "difficult data," e.g. the modified MNIST dataset as in [6], and examine how they perform compared to each other.

Acknowledgements

This work is a project of the course CSC2515 at University of Toronto, and is supervised by Professor G. E. Hinton and R. Salakhutdinov. The author is also grateful to her significant other, Li He, for detailed discussion on DBN and designing experiments, proofreading the report, sharing our independent work with each other, and his spiritual support.

References

[1] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimentionality of data with neural network. *Science*, 313(5786):504–507, 28 July 2006.

[2] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory*, pages 144–152. ACM Press, 1992.

[3] Geoffrey E. Hinton. Training products of experts by minimizing contrastive divergence. *Neural Computation*, 14(8):1771–1800, 2002.

[4] Geoffrey E. Hinton, Simon Osindero, and Yee Whye Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[5] Nicolas Le Roux and Yoshua Bengio. Representational power of restricted boltzmann machines and deep belief networks. *Neural Computation*, 20(6):1631–1649, 2008.

[6] Hugo Larochelle, Dumitru Erhan, Aaron Courville, James Bergstra, and Yoshua Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. *ICML2007*, pages 473–480, 2007.