# Deep Neural Networks

eingereichte
HAUPTSEMINARARBEIT
von

Florian Winter

geb. am 17.02.1987
wohnhaft in:
Apianstr. 7
85774 Unterföhring

Lehrstuhl für
STEUERUNGS- und REGELUNGSTECHNIK
Technische Universität München

Univ.-Prof. Dr.-Ing./Univ. Tokio Martin Buss
Univ.-Prof. Dr.-Ing. Sandra Hirche

| | |
|---|---|
| Betreuer: | M.Sc. Indar Sugiarto |
| Beginn: | 23.04.2012 |
| Abgabe: | 29.06.2012 |

## Abstract

Deep Neural Networks extract, like the human brain, multiple levels of representation from the input data and hence these networks with deep architecture are ideally suited for tasks in the field of Artificial Intelligence such as object and speech recognition. Thus it is a major aim to develop a learning algorithm which enables the training of deep networks effectively, whereby research results have already found promising approaches. This paper introduces in the state of the art in Deep Neural Networks and discusses in which practical applications they are beneficial in comparison to shallow networks.

## Zusammenfassung

Deep Neural Networks extrahieren, ähnlich dem menschliche Gehirn, mehrere Darstellungsebenen der Eingangsdaten und damit sind diese Netzwerke mit tiefer Struktur hervorragend geeignet für Aufgaben im Bereich der Künstlichen Intelligenz, wie beispielsweise der Objekt- und Spracherkennung. Deshalb ist es eine große Bestrebung einen Lernalgorithmus zu entwickeln, der das effektive Trainieren von tiefen Netzwerken erlaubt, wobei Forschungsergebnisse bereits vielversprechende Ansätze entdeckt haben. Diese Arbeit stellt den Stand der Technik von Deep Neural Networks vor und diskutiert in welchen praktischen Anwendungen sie vorteilhaft gegenüber flachen Netzwerken sind.

# Contents

# Chapter 1

# Introduction

The Multilayer Perceptron with the standard learning algorithm Backpropagation was the first computationally efficient model of an artificial neural network inspired by the human brain and is able to approximate a function to a certain level of precision. This type of neural network works quite well, but only with a small number of layers, i.e. with a shallow architecture.
Research results suggest to extend these shallow networks by adding more layers and hence to create a network with deep architecture, that might be able to perform more complicated tasks like our brain it does. The idea behind this is that each layer learns a different representation of the input and hence discovers features in the training data. However training Deep Neural Networks is known to be difficult and Backpropagation leads to poor solutions, wherefore novel strategies are necessary.
This paper gives a brief introduction in the principles of Multilayer Perceptrons and illustrates why learning more layers works not well with Backpropagation.
Since training deep networks is a major challenge, the benefits of a higher number of layers are summarized. There are already some promising approaches in deep learning like the Stacked Restricted Boltzmann Machines and the Stacked Autoassociators, which are the current state of the art in Deep Neural Networks and presented in this paper. These networks are trained with an efficient unsupervised greedy layerwise pre-learning algorithm, followed by a supervised fine-tuning phase and thus only a small number of training data need to be labeled. Furthermore this paper gives an overview of practical applications for Deep Neural Networks.
Finally the benefits of deep networks in comparison to networks with shallow architecture are discussed and open questions and possible directions for future research are outlined.

# Chapter 2

# Multilayer Perceptron

The Multilayer Perceptron (MLP) is an artificial neural network based on the biological microstructure of the human brain and has the aim to model the input data onto an appropriate output. This chapter describes briefly the structure of an MLP with its utilized learning algorithm called Backpropagation and summarizes the drawbacks of this type of network.

## 2.1 Structure of a Multilayer Perceptron

Perceptrons are the elementary computational units in an MLP and seek to emulate the action of a biological neuron. Since an MLP is a feedforward network, it can be described as a directed graph consisting of multiple layers of nodes [Jan04]. As illustrated in figure 2.1 the visible layer which represents the input layer and the output layer are separated by the $l$ hidden layers (in figure 2.1 one has $l = 2$). Each node of the graph in the output layer and the hidden layers is a Perceptron with a nonlinear activation function and in the visible layer the nodes are represented by a data vector $\boldsymbol{v}$. Given an input $\boldsymbol{v}$, the units of the hidden layers (hidden units) compute the hidden vectors $\boldsymbol{h}^m$ with $m = 1,2,\ldots,l$ and out of that the output vector $\boldsymbol{o}$ can be derived. In order to obtain a desired output the weights of the connections between the Perceptrons have to be adjusted accordingly, that can be achieved by the iterative learning algorithm Backpropagation.

## 2.2 Learning with Backpropagation

As mentioned in the previous section, changing the weights by using Backpropagation [RHW86] is the learning procedure in MLPs. After presenting the data to the network, the output will be calculated by propagating the input across the hidden units to the output layer. By comparing this result with the desired target an error function is determined and in order to adapt the randomly initialized weights after minimizing the error plane by gradient descent, it can be back-propagated how the

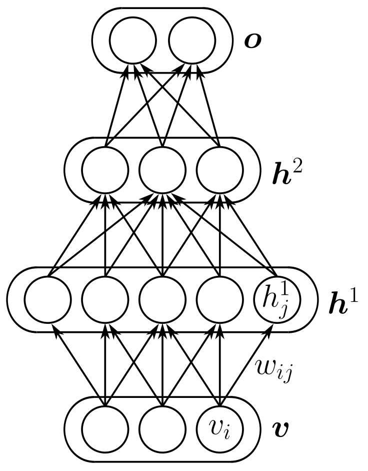

Figure 2.1: Structure of an MLP with a visible ($\boldsymbol{v}$), an output ($\boldsymbol{o}$) and two hidden layers ($\boldsymbol{h}^1$ and $\boldsymbol{h}^2$).

weights should be changed. Improvements relating to the generalization of the network can be reached by the use of various input vectors which are called training data and since Backpropagation is a supervised training algorithm, these have to be labeled[1].

## 2.3 Drawbacks in Training Multilayer Perceptrons

The adaption of the randomly initialized weights in MLPs is realized by a gradient-based minimization of a nonlinear error function and one of the issues in training MLPs is that gradient descent can get stuck in a poor local minima. Despite this problem, an MLP with one or two hidden layers works quite well if the number of layers and the number of Perceptrons in each layer is selected according to the task of the network. But it is known that an MLP with 3 or more hidden layers trained with Backpropagation leads to poor solutions [LBLL09]. The reasons for that are not known yet, but [LBLL09] suggests that the number or the width of unfavorable basins of attraction, which lead to suboptimal results when minimizing the error plane via gradient descent, could increase with the number of layers and furthermore there might exist many basins that can bring a low training error, but lead to very different generalization errors.

Another disadvantage in training MLPs is posed by the relatively slow learning, because Backpropagation becomes extremely computationally expensive in networks with more hidden layers and randomly initialized weights.

Furthermore, as described in the previous section, Backpropagation requires labeled training data and thus there is a problem if labeled data are scarce or nonexistent.

[1]In order to calculate the error signal the desired output for each training vector needs to be available and thats why these type of training data are called labeled.

By adding more hidden layers the number of labeled training samples should be increased to reach a good generalization of the network and thus additional layers entail a larger effort for the teacher.
Artificial neural networks with one or two hidden layers are referred to as shallow and by increasing number of layers it arises a Deep Neural Network (DNN). There are further drawbacks in learning with Backpropagation, but in this context just these ones are mentioned which lead to challenges in training DNNs.

# Chapter 3

# State of the Art in Deep Neural Networks

This chapter introduces in the current state of the art in DNNs and since the learning procedure Backpropagation is inefficient in training deep networks, it discusses why it make sense to research the depth, i.e. why it is worthwhile to develop learning algorithms which are able to train networks with deep architecture effectively. In this respect two different approaches are present to train these deep networks by using Restricted Boltzmann Machines (RBM) or Autoassociators (AA), respectively. At the end of this chapter an overview of the practical applications for DNNs is given.

## 3.1 Why Go Deep?

An artificial neural network with shallow architecture is able to achieve good results in a wide range of applications and training networks with more layers is known to be hard. It raises the question of why it could be useful to add more layers, i.e. why we should dispute with DNNs. Research findings suggest that deep architectures can be much more efficient than shallow architectures, e.g. some highly nonlinear functions can be represented more compactly in respect of parameters and computational units (neurons) [BL07], [Ben07].
In the development of artificial neural networks the ambitions are to emulate the human brain in order to create Artificial Intelligence (AI) and hence to learn complex behaviors like visual and semantic concepts. Our brain naturally envisions multiple layers of abstraction when we describe an aspect of our world and so we decompose a problem into sub-problems and multiple levels of representation [Ben07]. This fact leads to the idea that in AI tasks it could be very meaningful to create a network with a high number of layers to capture abstract representations of the input. Furthermore it can be assumed that learning highly nonlinear functions is essential to achieve an intelligent behavior, what presents difficulties in shallow networks [Ben07].

## 3.2 Restricted Boltzmann Machine

One promising approach to overcome the challenges in deep learning is presented by the use of RBMs [AHS85], [Smo86] which consist of stochastic binary neurons. The idea behind this concept is to adjust the weights “to maximize the probability that the network would generate the training data” [Hin07], thus this kind of network represents a generative model which is able to generate samples similar to those in the training set. In order to train this model, an unsupervised[1] algorithm has been found and after the learning procedure it has also the ability to classify.

### 3.2.1 Architecture of a Restricted Boltzmann Machine

An RBM (see figure 3.1) is composed of a visible and only one hidden layer[2] with undirected symmetrical connections between the stochastic binary neurons in those both layers and no connections between the units in one layer. The states of the

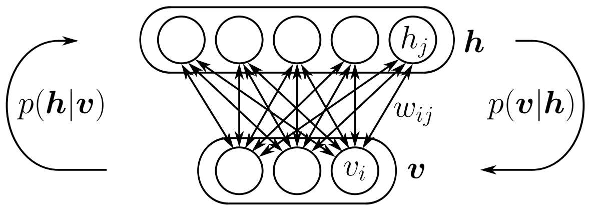
$h_j$
$\boldsymbol{h}$
$p(\boldsymbol{h}|\boldsymbol{v})$
$w_{ij}$
$p(\boldsymbol{v}|\boldsymbol{h})$
$v_i$
$\boldsymbol{v}$

Figure 3.1: Architecture of an RBM with a visible ($\boldsymbol{v}$) and a hidden layer ($\boldsymbol{h}$).

visible vector $\boldsymbol{v}$ are associated to the input data and the hidden vector $\boldsymbol{h}$ presents the states of the hidden neurons, i.e. the hidden features. Given a data vector $\boldsymbol{v}$, the conditional probability that the units of the hidden layer will be activated, i.e. $h_j = 1$, can be determined as

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_j p(h_j|\boldsymbol{v}) \quad \text{with} \quad p(h_j = 1|\boldsymbol{v}) = \operatorname{sigm}\left(b_j + \sum_i v_i w_{ij}\right) \tag{3.1}$$

where $\operatorname{sigm}(x) = 1/(1 + \exp(-x))$ is the logistic activation function, $w_{ij}$ the weight of the connection between the neurons $i$ and $j$ and $b_j$ the bias associated to the hidden variable $h_j$ [LBLL09]. After the states of the hidden units have been set, it is possible to produce a reconstruction of the input data by choosing the activation of the visible neurons with the conditional probability

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_i p(v_i|\boldsymbol{h}) \quad \text{with} \quad p(v_i = 1|\boldsymbol{h}) = \operatorname{sigm}\left(c_i + \sum_j h_j w_{ij}\right) \tag{3.2}$$

[1]An unsupervised learning algorithm does not require labeled training data and thus a teacher is unnecessary.

[2]Since one single RBM consists of only two layers it is not a type of DNN, but after stacking RBMs in subsection 3.2.3 they jointly form an deep network.

where $c_i$ is the bias associated to the visible variable $v_i$ [LBLL09]. Since an RBM is an energy-based model, an energy function over couples $(\boldsymbol{v},\boldsymbol{h})$ of visible and hidden vectors is defined by

$$E(\boldsymbol{v},\boldsymbol{h}) = -\sum_i c_i v_i - \sum_j b_j h_j - \sum_{i,j} v_i h_j w_{ij} \tag{3.3}$$

and hence the probability distribution over the visible vector $\boldsymbol{v}$ can be determined as

$$p(\boldsymbol{v}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) = \sum_{\boldsymbol{h}} p(\boldsymbol{v}|\boldsymbol{h})\, p(\boldsymbol{h}) = \sum_{\boldsymbol{h}} \frac{e^{-E(\boldsymbol{v},\boldsymbol{h})}}{\sum_{\boldsymbol{u},\boldsymbol{g}} e^{-E(\boldsymbol{u},\boldsymbol{g})}} \tag{3.4}$$

[LBLL09]. With these equations (3.3) and (3.4) each configuration $(\boldsymbol{v},\boldsymbol{h})$ corresponds to a specific energy whereas a high energy implies a low probability of occurrence and a low energy a high probability.
It should be noted that the RBM presented here is just suitable for binary data, but it is also possible to define the energy and probabilities for continuous valued inputs [BLLP06], [HN10], [Kri10].

### 3.2.2 Learning with Contrastive Divergence

Training an RBM implies to improve the reconstruction ability and thus to obtain a better generative model by maximizing the log-likelihood $\log p\left(\boldsymbol{v}^{(0)}\right)$ over the weights and biases for a given training example $\boldsymbol{v}^{(0)}$ [LBLL09]. This optimization problem can be solved by using alternating Gibbs sampling, where the pairwise correlation $< v_i^{(0)} h_j^{(0)} >$ can be calculated after a training vector $\boldsymbol{v}^{(0)}$ is clamped on the visible neurons and then running a Markov chain by alternating between updating all the hidden features $\boldsymbol{h}^{(k)}$ in parallel using equation 3.1 and the visible units $\boldsymbol{v}^{(k)}$ with $k = 0,1,2,\ldots$ in parallel using equation 3.2 [HOT06]. As soon as the Markov chain reaches its stationary distribution ($k \to \infty$) the correlation $< v_i^{(\infty)} h_j^{(\infty)} >$ can be determined. Hence the gradient over the weights of the log-probability of the training sample is given by

$$\frac{\partial \log p\left(\boldsymbol{v}^{(0)}\right)}{\partial w_{ij}} =< v_i^{(0)} h_j^{(0)} > - < v_i^{(\infty)} h_j^{(\infty)} > \tag{3.5}$$

and the gradients over the biases can be determined analogously [HOT06].
Since the convergence obtained from this learning rule typically is very slow [CPH05], the Contrastive Divergence (CD-$n$) algorithm provides to run alternating Gibbs sampling just for $n$ full steps[3] [Hin02]. It has been even found that running Gibbs sampling for only one full step (CD-1 for $n = 1$), before updating the hidden neurons

[3]Each full step is composed of updating $\boldsymbol{h}^{(k)}$ in parallel given $\boldsymbol{v}^{(k)}$ and then updating $\boldsymbol{v}^{(k+1)}$ in parallel given $\boldsymbol{h}^{(k)}$ to get a reconstruction of the training data.

$\boldsymbol{h}^{(1)}$ again and measuring the correlation $< v_i^{(1)} h_j^{(1)} >$, the training still works well [CPH05] and hence there is given an effective learning rule

$$\Delta w_{ij} = \epsilon \left( < v_i^{(0)} h_j^{(0)} > - < v_i^{(1)} h_j^{(1)} > \right) \tag{3.6}$$

for adjusting the weights[4] of an RBM with the learning rate $\epsilon$ [Hin07]. Thus CD-1 corresponds to a Markov chain with the sampling procedure

$$\boldsymbol{v}^{(0)} \xrightarrow{p\left(\boldsymbol{h}^{(0)}|\boldsymbol{v}^{(0)}\right)} \boldsymbol{h}^{(0)} \xrightarrow{p\left(\boldsymbol{v}^{(1)}|\boldsymbol{h}^{(0)}\right)} \boldsymbol{v}^{(1)} \xrightarrow{p\left(\boldsymbol{h}^{(1)}|\boldsymbol{v}^{(1)}\right)} \boldsymbol{h}^{(1)} \tag{3.7}$$

with the conditional probabilities $p\left(\boldsymbol{h}^{(k)}|\boldsymbol{v}^{(k)}\right)$ and $p\left(\boldsymbol{v}^{(k+1)}|\boldsymbol{h}^{(k)}\right)$ from equations (3.1) and (3.1) at the sampling step $k$ [LBLL09].

### 3.2.3 Stacked Restricted Boltzmann Machines

A better model of the training data can be obtained by stacking RBMs (see figure 3.2), i.e. treating the hidden features of one RBM as the input for a next RBM [Hin07]. In this way the Stacked RBMs (SRBM) can be trained in a greedy layer-

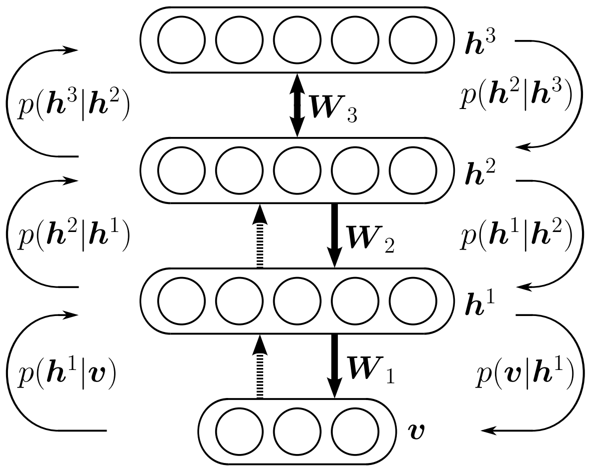
$\boldsymbol{h}^3$
$p(\boldsymbol{h}^3|\boldsymbol{h}^2)$
$\boldsymbol{W}_3$
$p(\boldsymbol{h}^2|\boldsymbol{h}^3)$
$\boldsymbol{h}^2$
$p(\boldsymbol{h}^2|\boldsymbol{h}^1)$
$\boldsymbol{W}_2$
$p(\boldsymbol{h}^1|\boldsymbol{h}^2)$
$\boldsymbol{h}^1$
$p(\boldsymbol{h}^1|\boldsymbol{v})$
$\boldsymbol{W}_1$
$p(\boldsymbol{v}|\boldsymbol{h}^1)$
$\boldsymbol{v}$

Figure 3.2: Architecture of Stacked RBMs with a generative (bold arcs) and a recognition (dashed arcs) path where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and $\boldsymbol{W}_3$ are the weight matrices.

wise manner (illustrated in figure 3.3) by learning a different representation of the data in a higher RBM after training the lower RBM and hold its weights and biases. It has been proved that stacking an additional RBM always increases a variational lower bound on the log-likelihood of the training data [Hin06] and hence the generative model will be improved by adding layers.
The SRBM represents a Deep Belief Network (DBN) and thus a type of DNN, whose

[4]The learning rules for adjusting the biases can be determined analogously.

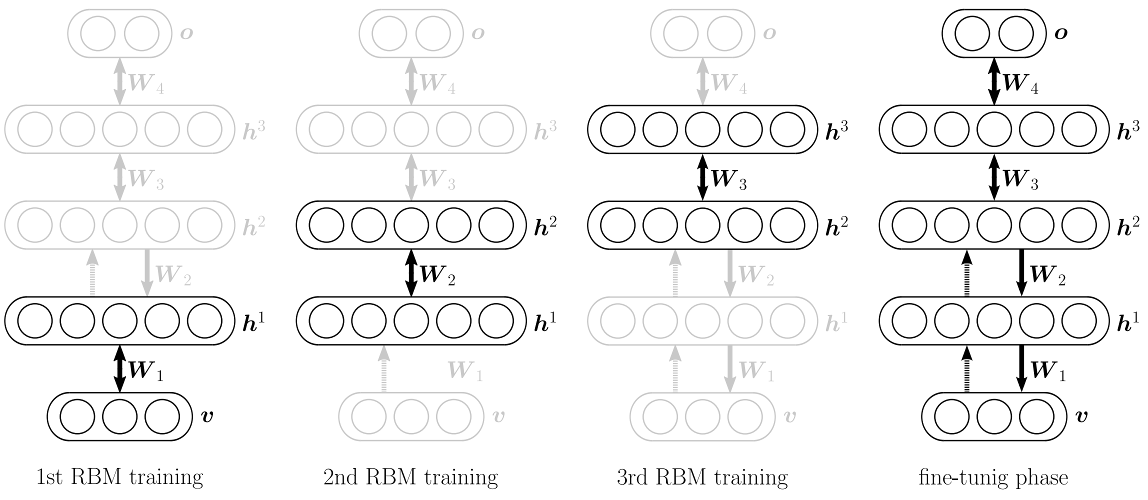

Figure 3.3: Illustration of the unsupervised greedy layerwise pre-training in an SRBM followed by a supervised fine-tuning phase.

top two hidden layers (in figure 3.3: $\boldsymbol{h}^2$ and $\boldsymbol{h}^3$) form an undirected associative memory and whose lower layers have directed either generative (top-down) or recognition (bottom-up) connections [HOT06]. The bottom-up connections can be used to infer a factorial representation from the states in the layer below and during the greedy layerwise initial learning these connections are tied to the top-down connections [HOT06].
After the unsupervised greedy layerwise pre-training phase, the weights and biases can be fine-tuned to improve either the generative or the discriminative performance of the SRBM by using a variant of the so-called wake-sleep algorithm [DFHN95] or Backpropagation, respectively. In order to improve the discriminative model, the unsupervised greedy layerwise learning phase finds features in the training data and then the supervised algorithm Backpropagation fine-tunes the recognition ability by slightly modifying "these features to adjust the boundaries between classes" [Hin07]. The top two hidden layer associative memory includes an energy landscape (cf. equation (3.3)) and the output layer $\boldsymbol{o}$ (see figure 3.3), which consists of the so-called label neurons, has the task to assign valleys in this landscape to an appropriate output signal, in order to display the result of the recognition or to calculate the error signal for the Backpropagation procedure, respectively [HOT06].

## 3.3 Stacked Autoassociators

AAs or Autoencoders (AE) are artificial neural networks with the aim to determine a representation of the input data from which the data can be reconstructed with high accuracy [Hin89]. An AA (see figure 3.4 where $l = 1$) consists of an encoding and a decoding layer with the same number of neurons in both layers, separated from $l$ hidden layers. In figure 3.4 the hidden representation of the input data $\boldsymbol{x}$

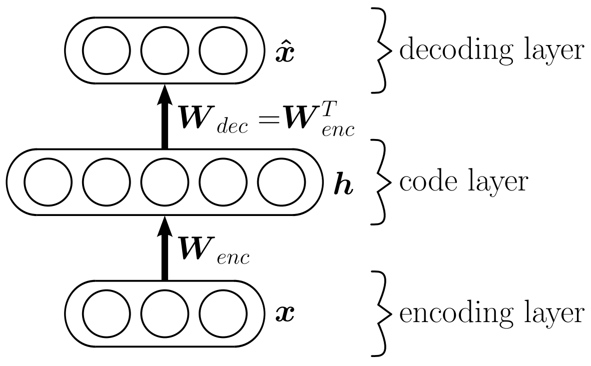

Figure 3.4: Structure of an AA consisting of an encoding ($\boldsymbol{x}$) and a decoding layer ($\hat{\boldsymbol{x}}$), separated from a hidden layer ($\boldsymbol{h}$).

represents the code $\boldsymbol{h}$ and after encoding, the reconstruction $\hat{\boldsymbol{x}}$ is obtained [LBLL09]. Furthermore the decoder's weight matrix $\boldsymbol{W}_{dec}$ can be set to the transpose of the encoder's weight matrix $\boldsymbol{W}_{enc}^T$ [LBLL09].
Stacked AAs (SAA) form an DBN and thus a type of DNN, which can be trained in an unsupervised greedy layerwise manner [BLLP06], [BLMV08]. This happens by using an SRBM as encoder part of the SAA and train it in the same way like in section 3.2 and then treating the same SRBM with transposed weight matrices as decoder part (see figure 3.5). After that pre-taining phase the network can be fine-tuned by using the supervised procedure Backpropagation to improve the accuracy of the reconstruction [HS06]. An SAA offers the possibility of nonlinear dimensionality reduction, if the number of neurons in the encoder part is decreasing in the higher layers [HS06].

## 3.4 Practical Applications of Deep Neural Networks

Practical applications of DNNs are primarily conceivable if it is necessary to deal with highly nonlinear functions, especially if labeled data are scarce or nonexistent. Hence the deep architecture enables the usage in AI tasks, e.g. to learn visual and semantic concepts.
DNNs have already been applied to generating and recognizing of images [HMRS11], [Hin06], acoustic signals [DHM12], [DHM$^+$11], videos [RT10] and motion-capture data [HT09]. That can be implemented to realize tasks like handwritten character recognition, audio classification, acoustic modeling, phone recognition, modeling motion style or recognition of facial expressions from videos. Deep architecture is also well suitable for natural language call-routing [HS11], regression with Gaussian Processes [HS08] or Spam filtering [TL07]. A further approach in natural language processing shows how an DNN is able to perform multi-task learning [CW08].

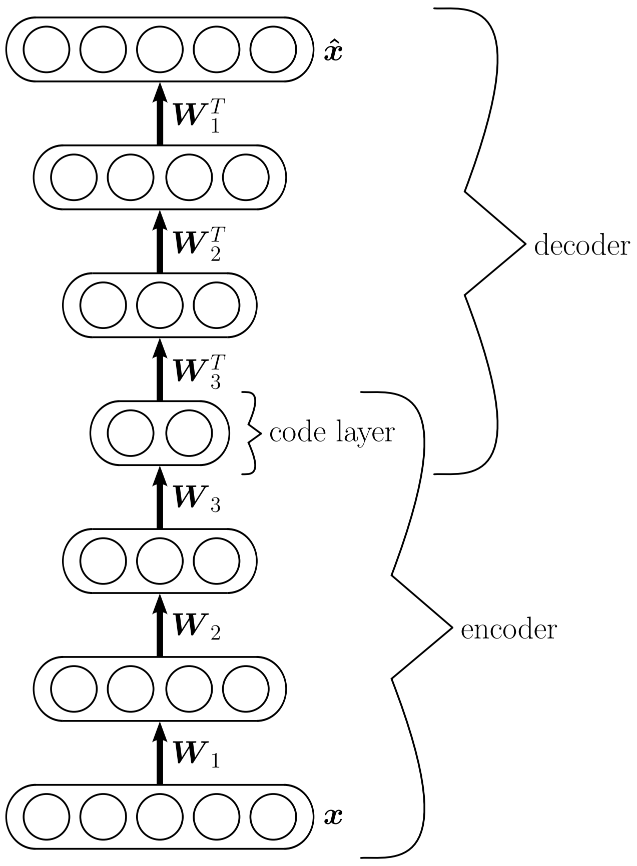

Figure 3.5: Architecture of an SAA consisting of an encoder and a decoder part with the code layer in the middle of the network.

SAAs enable a nonlinear dimensionality reduction which can be used for nonlinear data compression. Hence this type of deep network applies to semantic hashing and can be implemented for fast retrieval of documents [LMR10], [HS07] or images [HK11].

# Chapter 4

# Discussion

After introducing in the state of the art in DNNs, this chapter discusses about the utility of deep architecture and compares it with shallow networks. Furthermore open questions and the future direction of potential research efforts are summarized, after which this chapter and hence this paper closes with a conclusion.

## 4.1 Deep Architecture vs Shallow Architecture

The performance between shallow networks and different types of networks with deep architecture can be compared by a commonly used benchmark for handwritten digit recognition based on the MNIST data set[1]. Table 4.1 contains the classification errors achieved in [LBLL09] and shows how the deep networks which use unsupervised greedy layerwise pre-training (SRBM and SAA) outperform the other models such as the shallow network. Furthermore this experiment illustrates the importance of the unsupervised pre-training phase in DNNs. Despite these results it can not be assumed that networks with more layers are always better for each task, instead [Ben07] suggests that each function (i.e. each task) might require a specific minimum number of layers.
A great disadvantage in training shallow networks such as MLPs with the supervised learning algorithm Backpropagation is the necessity of labeled data. DBNs such as SRBMs or SAAs overcome this issue by containing not only a recognition model of the data, but also a generative one and hence an unsupervised training is feasible. Adding layers to a network might enable to learn representations that are high-level abstractions of the input which will be automatically discovered in DNNs [LBLL09]. Thus networks with deep architecture decompose a problem into sub-problems and multiple levels of representation, and for that reason they enable to perform multi-task learning.
Some families of functions which can be compactly represented by a network with $l$ layers might require an exponential number of computational units to be rep-

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

| Models | Training | Validation | Test |
|---|---|---|---|
| SRBM | 0% | 1.20% | 1.20% |
| SAA | 0% | 1.31% | 1.41% |
| DNN with supervised pre-training | 0% | 1.74% | 2.04% |
| DNN without pre-training | 0.004% | 2.07% | 2.40% |
| Shallow network without pre-training | 0% | 1.91% | 1.93% |

Table 4.1: Classification error in handwritten digit recognition based on MNIST training achieved in [LBLL09].

resented by a depth $l-1$ architecture [LBLL09]. Hence DNNs express their full potential when dealing with highly nonlinear functions, because some of them can be represented much more compactly in respect to the number of parameters and computational units than in shallow networks, in which a high number of labeled training data is required to capture such functions [LBLL09].

## 4.2 Open Questions and Future Research Directions

As mentioned in the previous section each task might require a particular minimum depth and thus choosing the dimensions of a network is not trivial. Since the architecture has a major influence on the performance of the network it is an aim to answer the question of how to determine the number of layers and computational units in each layer.
Besides SRBMs and SAAs there are other approaches to realize networks with deep architecture successfully. The first DNN which has effectively achieved a well generalization on visual data is the Convolutional Network [BDH$^+$89]. It is interesting that this network, typically consisting of 5, 6 or 7 layers, achieves the best results in handwritten character recognition, although it is trained by Backpropagation without any pre-training. This can be explained by the use of a topographic structure in each layer, i.e. each neuron has a fixed two-dimensional position that is associated with a location in the input image, along with a receptive field[2] [Ben07]. Thus the weights of the Convolutional Network are not initialized randomly, but manually and that entails a high effort in building such networks. The unsupervised greedy layerwise pre-training phase of the SRBM and the SAA seek to find a structure in the data automatically and thats a great benefit in contrast with Convolutional Networks. An interesting question at this is whether it is possible to understand the representations that are learned in the deeper layers of an DBN and whether are they similar to them in the human vision system. Although it is easy to generate samples form a generative model, but it is very difficult to discover the meaning of

[2]A receptive field is the range of the input image in which the neuron alter the firing.

the state of an individual neuron.
Instead of applying RBMs to create an DNN, the usage of Conditional RBMs (CRBM) has given promising results [HLM11]. A further development in the field of deep networks is presented by Multilayer Kernel Machines [CS09].

## 4.3 Conclusion

The approach of DNNs holds great promise for AI tasks like human-level object recognition and speech perception, in which networks with shallow architecture lead to poor results, because they are not able to learn multiple layers of representation. A significantly benefit of deep networks in comparison to shallow networks is that they enable to capture highly nonlinear functions with a compact set of parameters, especially if labeled training data are scarce or nonexistent.
Well working deep network approachs are DBNs, such as the SRBM and the SAA and can be trained with the unsupervised greedy layerwise pre-training algorithm CD which allows setting the parameter space region in a manner, that supervised fine-tuning avoids local minima. The idea behind this is to learn a model that is able to generate data, what decreases the number of required labeled training samples. By learning layer-by-layer a different representation of the input, the network discovers features in the data automatically that enable good generation. After this pre-training phase the supervised algorithm Backpropagation fine-tunes the weights in order to improve the recognition ability. This training procedure achieves a great accuracy in many complex applications, even if there is just a small number of labeled training data available.

# List of Figures

# Bibliography

[AHS85] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski. A Learning Algorithm for Boltzmann Machines. *Cognitive Science*, Volume 9:147–169, 1985.

[BDH+89] B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, L. D. Jackel, and Y. LeCun. Backpropagation Applied to Handwritten Zip Code Recognition. *Neural Computation*, Volume 1(4):541–551, 1989.

[Ben07] Y. Bengio. Learning Deep Architectures for AI. Technical Report 1312, Dept. IRO, Universite de Montreal, 2007.

[BL07] Y. Bengio and Y. LeCun. Scaling Learning Algorithms towards AI. *Large-Scale Kernel Machines*, Volume 34, 2007.

[BLLP06] Y. Bengio, P. Lamblin, H. Larochelle, and V. Popovici. Greedy Layer-Wise Training of Deep Networks. Technical Report 1282, Dept. IRO, Universite de Montreal, 2006.

[BLMV08] Y. Bengio, H. Larochelle, P. A. Manzagol, and P. Vincent. Extracting and Composing Robust Features with Denoising Autoencoders. Technical Report 1316, 2008.

[CPH05] M. A. Carreira-Perpignan and G. E. Hinton. On Contrastive Divergence Learning. *Artificial Intelligence and Statistics*, 2005.

[CS09] Y. Cho and L. K. Saul. Kernel Methods for Deep Learning. *NIPS*, 2009.

[CW08] R. Collobert and J. Weston. A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning. *International Conference on Machine Learning*, 2008.

[CT02] J. Conradt, G. Tevatia, S. Vijayakumar, & S. Schaal. On-line Learning for Humanoid Robot Systems, International Conference on Machine Learning (ICML2000), Stanford, USA, 2000.

[DFHN95] P. Dayan, B. J. Frey, G. E. Hinton, and R. Neal. The wake-sleep algorithm for unsupervised neural networks. *Science*, Volume 268(5214):1158–1161, 1995.

[DHM+11] G. E. Dahl, G. E. Hinton, A. Mohamed, M. Picheny, B. Ramabhadran, and T. Sainath. Deep Belief Networks using Discriminative Features for Phone Recognition. *ICASSP*, 2011.

[DHM12] G. E. Dahl, G. E. Hinton, and A. Mohamed. Acoustic Modeling using Deep Belief Networks. *IEEE Trans. on Audio, Speech, and Language Processing*, 2012.

[Hin89] G. E. Hinton. Connectionist Learning Procedures. *Artificial Intelligence*, Volume 40:185–234, 1989.

[Hin02] G. E. Hinton. Training Products of Experts by Minimizing Contrastive Divergence. *Neural Computation*, Volume 14:1771–1800, 2002.

[Hin06] G. E. Hinton. To Recognize Shapes, First Learn to Generate Images. Technical Report UTML TR 2006-004, University of Toronto, 2006.

[Hin07] G. E. Hinton. Learning multiple layers of representation. *Trends in Cognitive Sciences*, Volume 11:428–434, 2007.

[HK11] G. E. Hinton and A. Krizhevsky. Using Very Deep Autoencoders for Content-Based Image Retrieval. *European Symposium on Artificial Neural Networks ESANN*, 2011.

[HLM11] G. E. Hinton, H. Larochelle, and V. Mnih. Conditional Restricted Boltzmann Machines for Structured Output Prediction. *Uncertainty in Artificial Intelligence*, 2011.

[HMRS11] G. E. Hinton, V. Mnih, M. Ranzato, and J. Susskind. On deep generative models with applications to recognition. *IEEE Conference on Computer Vision and Pattern Recognition*, 2011.

[HN10] G. E. Hinton and V. Nair. Rectified Linear Units Improve Restricted Boltzmann Machines. *International Conference on Machine Learning*, (27th), 2010.

[HOT06] G. E. Hinton, S. Osindero, and Y. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, Volume 18(7):1527–1554, 2006.

[HS06] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, Volume 313(5786):504–507, 2006.

[HS07] G. E. Hinton and R. R. Salakhutdinov. Semantic Hashing. *Proceedings of the SIGIR Workshop on Information Retrieval and Applications of Graphical Models*, 2007.

[HS08] G. E. Hinton and R. Salakhutdinov. Using Deep Belief Nets to Learn Covariance Kernels for Gaussian Processes. *Advances in Neural Information Processing Systems*, (20), 2008.

[HS11] G. E. Hinton and R. Sarikaya. Deep Belief Nets for Natural Language Call-Routing. *ICASSP*, 2011.

[HT09] G. E. Hinton and G. W. Taylor. Factored Conditional Restricted Boltzmann Machines for Modeling Motion Style. *International Conference on Machine Learning*, (26th):1025–1032, 2009.

[Jan04] A. Janczak. *Identification Of Nonlinear Systems Using Neural Networks And Polynomial Models: A Block-Oriented Approach*. Springer, 1st edition, 2004.

[Kri10] A. Krizhevsky. Convolutional Deep Belief Networks on CIFAR-10. Technical report, University of Toronto, 2010.

[LBLL09] H. Larochelle, Y. Bengio, J. Louradour, and P. Lamblin. Exploring Strategies for Training Deep Neural Networks. *Journal of Machine Learning Research*, (10):1–40, Jan 2009.

[LMR10] Y. LeCun, P. Mirowski, and M. A. Ranzato. Dynamic Auto-Encoders for Semantic Indexing. *Proceedings of the NIPS 2010 Workshop on Deep Learning*, 2010.

[RHW86] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning Internal Representations by Error Propagation. *Parallel distributed processing: explorations in the microstructure of cognition*, Volume 1:318–362, 1986.

[RT10] A. Rao and N. Thiagarajan. Recognizing facial expressions from videos using Deep Belief Networks. Project, Stanford University, 2010.

[Smo86] P. Smolensky. Information Processing in Dynamical Systems - Foundations of Harmony Theory. *Parallel distributed processing: explorations in the microstructure of cognition*, Volume 1:194–281, 1986.

[TL07] G. Tzortzis and A. Likas. Deep Belief Networks for Spam Filtering. *Tools with Artificial Intelligence*, Volume 2:306–309, 2007.