Feature Extraction with Deep Neural Networks by a Generalized Discriminant Analysis

André Stuhlsatz, Jens Lippel, and Thomas Zielke

***Abstract*— We present an approach to feature extraction that is a generalization of the classical linear discriminant analysis (LDA) on the basis of deep neural networks (DNNs). As for LDA, discriminative features generated from independent Gaussian class conditionals are assumed. This modeling has the advantages that the intrinsic dimensionality of the feature space is bounded by the number of classes and that the optimal discriminant function is linear. Unfortunately, linear transformations are insufficient to extract optimal discriminative features from arbitrarily distributed raw measurements. The g*eneralized* d*iscriminant* a*nalysis* (GerDA) proposed in this paper uses nonlinear transformations that are learnt by DNNs in a semisupervised fashion. We show that the feature extraction based on our approach displays excellent performance on real-world recognition and detection tasks, such as handwritten digit recognition and face detection. In a series of experiments, we evaluate GerDA features with respect to dimensionality reduction, visualization, classification, and detection. Moreover, we show that GerDA DNNs can preprocess truly high-dimensional input data to low-dimensional representations that facilitate accurate predictions even if simple linear predictors or measures of similarity are used.**

***Index Terms*— Deep neural networks, dimensionality reduction, discriminant analysis, face detection, feature extraction, restricted Boltzmann machines, sensor fusion.**

I. Introduction

THERE are two competing families of approaches for the extraction of optimal discriminative features from arbitrarily distributed data. One uses local information instead of global statistics, as in [1] and [2]. The other family uses nonlinear transformations [3], [4], with kernelization being the most important method [5]–[7]. The approach taken for genralized discriminant analysis (GerDA) belongs to both of these families. So does the nonlinear neighborhood components analysis (NNCA) proposed by Salakhutdinov and Hinton [8]. GerDA learns nonlinear transformations from arbitrary data distributions to Gaussian distributions, while a weighting scheme forces localized neighborhoods.

Linear discriminant analysis (LDA) [9] seeks an invertible linear transform $\mathbf{A} : \mathbb{R}^d \rightarrow \mathbb{R}^r$, improving discrimination between features $\boldsymbol{h}_n = \mathbf{A}\boldsymbol{x}_n$ lying in a subspace $\mathbb{R}^r \subseteq \mathbb{R}^d$ of transformed measurements $\boldsymbol{x}_n \in \mathbb{R}^d$, $1 \leq n \leq N$, of different classes ω_i, $1 \leq i \leq C$. Assuming discriminative features $\boldsymbol{h} = \mathbf{A}\boldsymbol{x}$, which are identically and independently drawn from Gaussian class-conditional distributions $p(\boldsymbol{h}|\omega_i) = \mathcal{N}(\boldsymbol{\mu}_i, \Sigma)$ with class-common diagonal covariance matrix Σ and class-means $\boldsymbol{\mu}_i$, in which the last $d - r$, $r \leq d$, components are tied for all classes, a maximum likelihood estimation of $\mathbf{A}$ can equivalently be computed by a maximization of a discriminant criterion $Q_h : \mathcal{F} \rightarrow \mathbb{R}$, $\mathcal{F} = \{\mathbf{A} : \mathbf{A} \in \mathbb{R}^{r \times d}\}$ with

$$Q_h(\mathbf{A}) := \text{trace}\left\{\mathbf{S}_T^{-1}\mathbf{S}_B\right\}. \tag{1}$$

The matrix $\mathbf{S}_T := (1/N)\sum_{n=1}^{N}(\boldsymbol{h}_n - \boldsymbol{m})(\boldsymbol{h}_n - \boldsymbol{m})^T$ is the common total scatter matrix with total mean $\boldsymbol{m} := (1/N)\sum_{n=1}^{N}\boldsymbol{h}_n$, and $\mathbf{S}_B := (1/N)\sum_{i=1}^{C} N_i(\boldsymbol{m}_i - \boldsymbol{m})(\boldsymbol{m}_i - \boldsymbol{m})^T$ is the common between-class scatter matrix with class means $\boldsymbol{m}_i := (1/N_i)\sum_{n:\omega_i=\omega(\boldsymbol{x}_n)}\boldsymbol{h}_n$.

With these assumptions, in a Bayesian sense, the optimal discriminant function is linear in the LDA feature space. Moreover, the discriminant information is not affected by reducing the feature space to $r \geq C-1$ dimensions. Therefore, LDA is a popular dimension reduction and visualization tool used in many pattern recognition applications. Unfortunately, LDA and a subsequent linear classification often fail to deliver satisfactory accuracy in real-world applications, because in general a linear mapping $\mathbf{A}$ cannot transform arbitrarily distributed random variables into independently Gaussian distributed ones. A natural generalization is to define $\mathcal{F}$ to be a space of nonlinear transformations $f : \mathbb{R}^d \rightarrow \mathbb{R}^r$, still assuming discriminative features $\boldsymbol{h} = f(\boldsymbol{x})$ with statistical properties as of LDA features. The idea is that a sufficiently large space $\mathcal{F}$ potentially contains a nonlinear feature extractor $f^* \in \mathcal{F}$ increasing the discriminant criterion Q_h compared to an insufficient linear extractor $\mathbf{A}$

$$Q_h(\mathbf{A}) \leq Q_h(f^*) = \max\left(\text{trace}\left\{\mathbf{S}_T^{-1}\mathbf{S}_B\right\} : f \in \mathcal{F}\right) \tag{2}$$

which in turn implies $p(f^*(\boldsymbol{x})|\omega_i)$ approximates the assumed model better than $p(\mathbf{A}\boldsymbol{x}|\omega_i)$. In the following, we choose $\mathcal{F}$ to be defined by a deep neural network (DNN), which is a multilayer artificial neural network with more than one hidden layer and often millions of free parameters. A space $\mathcal{F}$ defined by a DNN may contain nearly any continuous function. Therefore, the chance to obtain $f^* \in \mathcal{F}$ depends only on how the learning is performed. Unfortunately, optimizing a DNN with standard methods, such as backpropagation, is known to be challenging due to many local optima in

Manuscript received September 1, 2009; revised December 24, 2011; accepted December 29, 2011. Date of publication February 8, 2012; date of current version March 6, 2012.

A. Stuhlsatz is with the Research and Development Division, SMS Siemag AG, Düsseldorf 40237, Germany (e-mail: andre.stuhlsatz@sms-siemag.com).

J. Lippel and T. Zielke are with the Department of Mechanical and Process Engineering, Düsseldorf University of Applied Sciences, Düsseldorf 40474, Germany (e-mail: jens.lippel@fh-duesseldorf.de; thomas.zielke@fh-duesseldorf.de).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNNLS.2012.2183645

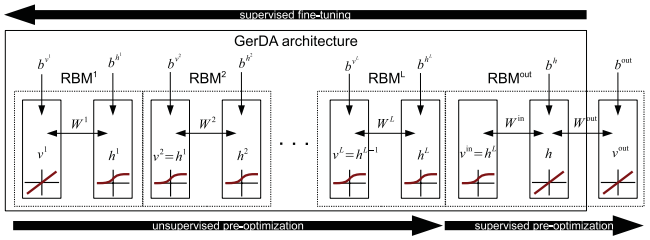

Fig. 1. GerDA architecture. Multiple layers pre-optimized partly unsupervised and supervised, followed by a supervised fine-tuning.

the considered objective function. In [10], a very effective stochastic pre-optimization of DNNs has been proposed for an auto-encoder task to circumvent the drawback of optimizing DNNs. Recently, [11] and [12] explored different DNN architectures and optimization strategies, all based on the idea of unsupervised pre-optimization and subsequent supervised fine-tuning.

In [13], unsupervised pre-optimization is uncovered as a new form of regularization that improves generalization besides an optimization effect supporting the fine-tuning phase. In [14] and [15], we introduced a novel DNN learning scheme combining unsupervised and supervised pre-optimization. Here we present our learning scheme and supporting experiments together with an improvement of the proposed fine-tuning method.

Following this introduction, Section II-A describes the method chosen for the unsupervised pre-optimization of a stack of restricted Boltzmann machines (RBMs). This includes the way how real-valued inputs to the network are handled and our extension of an output RBM designed for supervised training. Section II-B presents our modification of the contrastive divergence algorithm (CD_n^{IO}) in order to realize a supervised RBM training. Section II-C presents our fine-tuning method featuring a global weighting scheme. Our experiments described in Section III have been devised to answer the question whether CD_1^{IO} really improves GerDA fine-tuning (III-A) and to test GerDA's performance for several different applications, namely, dimensionality reduction and data visualization (III-B), classification of truly high-dimensional data (III-C), and sensor fusion (III-D). Within these sections on different applications, the results obtained with GerDA are compared with results from other methods. Finally, the last section summarizes the essential ideas of our work and explains why using DNNs has an advantage over kernel methods in many practical applications.

II. Semisupervised GerDA

A. Semisupervised Pre-Optimization

The unsupervised pre-optimization of a GerDA DNN is performed by first training a stack of RBMs (Fig. 1), and then using the RBM's stochastically learned parameters as initial configuration for the full network [16]. The initial configuration is hopefully near a good solution w.r.t. the objective to be optimized. After the first RBM of a stack is trained, the expectations of the output states $\boldsymbol{h}^i$ are used as input states $\boldsymbol{v}^{i+1}$ for training the next RBM in a stack, which in turn provides the input of its successor, and so forth, until the topmost RBM is trained. This greedy layerwise training has been proposed in [17]. It has been shown that layerwise training improves a variational bound of the logarithmic prior of the input data $\log p(\boldsymbol{v}^1)$ when layers with increasing number of states are added and the higher level weights are appropriately initialized.

Unsupervised training of a single binary RBM of the ith layer ($2 \leq i \leq L$), shown in Fig. 1, is performed via stochastic gradient descent in the Kullback–Leibler divergence

$$d(P^0||P^\infty;\Theta^i) := \sum_{\boldsymbol{v}^i} P^0(\boldsymbol{v}^i)\log\left(\frac{P^0(\boldsymbol{v}^i)}{P^\infty(\boldsymbol{v}^i;\Theta^i)}\right) \quad (3)$$

assuming system states $\boldsymbol{s}^i := ((\boldsymbol{v}^i)^T, (\boldsymbol{h}^i)^T)^T$, $\boldsymbol{v}^i \in \{0,1\}^{N_{v^i}}$, $\boldsymbol{h}^i \in \{0,1\}^{N_{h^i}}$ with distribution

$$P^\infty(\boldsymbol{v}^i;\Theta^i) = \frac{1}{Z(\Theta^i)}\sum_{\boldsymbol{h}^i}\exp\left(-H(\boldsymbol{s}^i;\Theta^i)\right) \quad (4)$$

$$Z(\Theta^i) := \sum_{\boldsymbol{s}^i}\exp\left(-H(\boldsymbol{s}^i;\Theta^i)\right) \quad (5)$$

given the network parameters $\Theta^i := (\mathbf{W}^i, \boldsymbol{b}^i)$, i.e., the weights $\mathbf{W}^i \in \mathbb{R}^{N_{v^i}\times N_{h^i}}$ and biases $\boldsymbol{b}^i := ((\boldsymbol{b}^{v^i})^T, (\boldsymbol{b}^{h^i})^T)^T$, $\boldsymbol{b}^{v^i} \in \mathbb{R}^{N_{v^i}}$, $\boldsymbol{b}^{h^i} \in \mathbb{R}^{N_{h^i}}$. For binary states, the function H reads as

$$H(\boldsymbol{s}^i;\Theta^i) := -(\boldsymbol{v}^i)^T\mathbf{W}^i\boldsymbol{h}^i - (\boldsymbol{b}^i)^T\boldsymbol{s}^i. \quad (6)$$

Given a finite input sample $(\boldsymbol{x}_n)_{n=1}^N$, the distribution P^0 is assumed to be the empirical distribution $P^0(\boldsymbol{v}^i) = (1/N)\sum_{n=1}^N \delta(\boldsymbol{v}^i - \boldsymbol{v}^i(\boldsymbol{x}_n))$, with $\boldsymbol{v}^i(\boldsymbol{x}_n)$ denoting the visual state of the RBM in the ith layer dependent on the input sample $\boldsymbol{x}_n$, e.g. $\boldsymbol{v}^1(\boldsymbol{x}_n) = \boldsymbol{x}_n$.

For effectively pre-optimizing GerDA DNNs, we propose an extended semisupervised architecture: First, the visual states $\boldsymbol{v}^1$ of an input layer RBM (Fig. 1) are modeled continuously and Gaussian-distributed in order to facilitate real-valued inputs to the network, which is an important requirement for many real-world applications. Gaussian-distributed visual states for the input RBM, i.e., $\boldsymbol{s}^1 = (\boldsymbol{v}^1, \boldsymbol{h}^1) \in \mathbb{R}^{N_{v^1}} \times \{0,1\}^{N_{h^1}}$, can be introduced through a quadratic energy function [18]

$$H(\boldsymbol{s}^1;\Theta^1) := \frac{1}{2}(\boldsymbol{v}^1 - \boldsymbol{b}^{v^1})^T(\Sigma^1)^{-1}(\boldsymbol{v}^1 - \boldsymbol{b}^{v^1}) - (\boldsymbol{v}^1)^T(\Sigma^1)^{-\frac{1}{2}}\mathbf{W}^1\boldsymbol{h}^1 - (\boldsymbol{b}^{h^1})^T\boldsymbol{h}^1 \quad (7)$$

with diagonal covariance matrix

$$\Sigma^1 := diag\left((\sigma_1^1)^2, \ldots, (\sigma_{N_{v^1}}^1)^2\right). \quad (8)$$

Since supervised optimization of Q_h via RBMs is not possible directly, our second extension is an output RBM with extra visual output units. Added on top of the stack (Fig. 1), it learns input–output associations leading to an indirectly maximized Q_h: [19] shows that multilayer artificial neural networks with linear output layer

$$\boldsymbol{v}^{\text{out}}(\boldsymbol{x}) = \boldsymbol{W}^{\text{out}}\boldsymbol{h}(\boldsymbol{x}) + \boldsymbol{b}^{\text{out}} \quad (9)$$

maximize asymptotically a specific discriminant criterion evaluated in the r-dimensional space spanned by the last hidden

layer outputs $\boldsymbol{h}(\boldsymbol{x}) \in \mathbb{R}^r$ if the mean squared error (MSE) is minimized between outputs $(\boldsymbol{v}^{\text{out}}(\boldsymbol{x}_n))_{n=1}^{\infty}$ and associated targets $(\boldsymbol{t}_n)_{n=1}^{\infty}$. In particular, for a finite sample $(\boldsymbol{x}_n)_{n=1}^{N}$, an MSE minimization regarding the target coding

$$t_n^i := \begin{cases} \sqrt{N/N_i} & \text{if } \omega(\boldsymbol{x}_n) = \omega_i \\ 0 & \text{otherwise} \end{cases} \tag{10}$$

where N_i the number of examples of class ω_i, and t_n^i is the ith component of a target vector $\boldsymbol{t}_n \in \mathbb{R}^C$, approximates the maximum of the discriminant criterion Q_h. Since the considered target codes (10) are real-valued and we want to extract continuous features as well, the output RBM's visual output states and hidden states are modeled Gaussian-distributed by introducing an extended energy function

$$\begin{aligned} H(\boldsymbol{s};\Theta) &:= \frac{1}{2}\left(\boldsymbol{v}^{\text{out}} - \boldsymbol{b}^{\text{out}}\right)^T (\Sigma^{\text{out}})^{-1}\left(\boldsymbol{v}^{\text{out}} - \boldsymbol{b}^{\text{out}}\right) \\ &-(\boldsymbol{v}^{\text{out}})^T(\Sigma^{\text{out}})^{-\frac{1}{2}}\mathbf{W}^{\text{out}}\boldsymbol{h} - (\boldsymbol{v}^{in})^T\mathbf{W}^{in}(\Sigma^h)^{-\frac{1}{2}}\boldsymbol{h} \\ &+\frac{1}{2}\left(\boldsymbol{h} - \boldsymbol{b}^h\right)^T(\Sigma^h)^{-1}\left(\boldsymbol{h} - \boldsymbol{b}^h\right) \end{aligned} \tag{11}$$

with $\boldsymbol{s} := (\boldsymbol{v}^{in}, \boldsymbol{v}^{\text{out}}, \boldsymbol{h}) \in \{0,1\}^{N_{\boldsymbol{v}^{in}}} \times \mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}} \times \mathbb{R}^r$, $\Theta := (\mathbf{W}^{in}, \mathbf{W}^{\text{out}}, \boldsymbol{b}^h, \boldsymbol{b}^{\text{out}})$, $\mathbf{W}^{in} \in \mathbb{R}^{N_{v^{in}} \times r}$, $\mathbf{W}^{\text{out}} \in \mathbb{R}^{N_{v^{\text{out}}} \times r}$, $\boldsymbol{b}^h \in \mathbb{R}^r$, $\boldsymbol{b}^{\text{out}} \in \mathbb{R}^{N_{v^{\text{out}}}}$ and diagonal covariance matrices

$$\Sigma^{\text{out}} := \text{diag}\left((\sigma_1^{\text{out}})^2, \ldots, (\sigma_{N_{v^{\text{out}}}}^{\text{out}})^2\right) \tag{12}$$

$$\Sigma^h := \text{diag}\left((\sigma_1^h)^2, \ldots, (\sigma_r^h)^2\right). \tag{13}$$

All σ_n^1 (input layer) and σ_n^{out} (output layer) are chosen to be the mean variance of v^1 and v^{out}, respectively. The σ's of the output RBM's hidden layer are set to 1.0 (c.f. [8]). To pre-optimize the last hidden layer w.r.t. a Fisher criterion following [19], we use a stochastic search for minimizing the MSE at the output layer instead of a deterministic optimization.

B. Efficient Training of Single RBMs Via Contrastive Divergence

RBM training (Section II-A) is known to be prohibitively slow, because a gradient descent

$$\Theta_{kl}^i \longleftarrow \Theta_{kl}^i - \eta \cdot \frac{\partial d(P^0||P^{\infty};\Theta^i)}{\partial \Theta_{kl}^i} \eta \in (0,1] \tag{14}$$

in the Kullback–Leibler divergence (3) involves computation of stochastic gradients

$$\begin{aligned} \frac{\partial d(P^0||P^{\infty};\Theta^i)}{\partial \Theta_{kl}^i} &= \left\langle \left\langle \frac{\partial H(\boldsymbol{s}^i;\Theta^i)}{\partial \Theta_{kl}^i} \right\rangle_{P^{\infty}(\boldsymbol{h}^i|\boldsymbol{v}^i;\Theta^i)} \right\rangle_{P^0(\boldsymbol{v}^i)} \\ &- \left\langle \frac{\partial H(\boldsymbol{s}^i;\Theta^i)}{\partial \Theta_{kl}^i} \right\rangle_{P^{\infty}(\boldsymbol{s}^i;\Theta^i)} \end{aligned} \tag{15}$$

where $\langle\cdot\rangle_{P(\cdot)}$ denotes expectations w.r.t. a distribution P. Practically, these expectations are estimated by sampling from the distributions P^0 and P^{∞} running a Markov chain to equilibrium via Gibbs sampling [20]. While the first expectation in (15) is easy to compute, an efficient calculation of the second expectation remains a problem. Making RBM training practical, unlike minimizing (3), in [21] the difference of two Kullback–Leibler distances

$$CD_n(\Theta^i) := d(P^0||P^{\infty};\Theta^i) - d(P^n||P^{\infty};\Theta^i) \tag{16}$$

is minimized instead. Here, P^n denotes the distribution of visual states $\boldsymbol{v}^i$ in case the Markov chain for actually sampling from P^{∞} is already stopped after a small number $n > 0$ of steps. The idea behind the CD_n-heuristics is that the distribution P^n is closer to the equilibrium distribution than P^0. Thus, $d(P^0||P^{\infty};\Theta^i)$ is greater than $d(P^n||P^{\infty};\Theta^i)$ unless P^n equals P^0. If all transitions of the Markov chain have nonzero probability, then $P^n = P^0$ yields $P^{\infty} = P^0$. Therefore, it holds $CD_n(\Theta^i) = 0$ if the RBM is perfectly trained.

In analogy with (15), the approximated[1] partial derivatives of CD_n read as

$$\begin{aligned} \frac{\partial CD_n(\Theta^i)}{\partial \Theta_{kl}^i} &\approx \left\langle \left\langle \frac{\partial H(\boldsymbol{s}^i;\Theta^i)}{\partial \Theta_{kl}^i} \right\rangle_{P^{\infty}(\boldsymbol{h}^i|\boldsymbol{v}^i;\Theta^i)} \right\rangle_{P^0(\boldsymbol{v}^i)} \\ &- \left\langle \frac{\partial H(\boldsymbol{s}^i;\Theta^i)}{\partial \Theta_{kl}^i} \right\rangle_{P^n(\boldsymbol{s}^i;\Theta^i)}. \end{aligned} \tag{17}$$

Moreover, it is actually possible to compute empirical expectations by simulation from P^1 using a 1-step Gibbs sampler. Let be $\mathbf{W}_{k(\cdot)}$ the kth row of a matrix $\mathbf{W}$ and $\mathbf{W}_{(\cdot)l}$ the lth column, respectively. For layers $2 \le i \le L$, hidden states $\hat{\boldsymbol{h}}^i$ are first sampled with clamped $\boldsymbol{v}^i = \boldsymbol{v}^i(\boldsymbol{x}_n)$ according to

$$P^{\infty}(h_l^i = 1|\boldsymbol{v}^i;\Theta^i) = \text{sigm}\left((\boldsymbol{v}^i)^T\mathbf{W}_{(\cdot)l}^i + b_l^{h^i}\right) \tag{18}$$

with sigmoid function $\text{sigm}(x) := 1/(1+\exp(-x))$. Then, visual states $\hat{\boldsymbol{v}}^i$ are sampled with fixed hidden states $\boldsymbol{h}^i = \hat{\boldsymbol{h}}^i$ according to

$$P^{\infty}(v_k^i = 1|\boldsymbol{h}^i;\Theta^i) = \text{sigm}\left(\mathbf{W}_{k(\cdot)}^i\boldsymbol{h}^i + b_k^{v^i}\right) \tag{19}$$

and, vice versa, the hidden states are updated by simulation from (18) with $\boldsymbol{v}^i = \hat{\boldsymbol{v}}^i$ clamped at the visual units. This procedure is repeated until the full sample $(\boldsymbol{x}_n)_{n=1}^N$ is processed.

Similarly, for layer $i = 1$, the sampling procedure is the same, but the simulation of states is according to

$$P^{\infty}(h_l^1 = 1|\boldsymbol{v}^1;\Theta^1) = \text{sigm}\left((\boldsymbol{v}^1)^T(\Sigma^1)^{-\frac{1}{2}}\mathbf{W}_{(\cdot)l}^1 + b_l^{h^1}\right) \tag{20}$$

and

$$p^{\infty}(v_k^1|\boldsymbol{h}^1;\Theta^1) = \mathcal{N}\left(\mu_k^1(\boldsymbol{h}^1), (\sigma_k^1)^2\right) \tag{21}$$

which is a normal distribution with mean $\mu_k^1(\boldsymbol{h}^1) := b_k^{v^1} + \sigma_k^1\mathbf{W}_{k(\cdot)}^1\boldsymbol{h}^1$ and variance $(\sigma_k^1)^2$.

For pre-optimization of the output RBM with real-valued target codes (10), we modified the CD-heuristic (16)

$$CD_n^{IO}(\Theta) := d(p^0||p^{\infty};\Theta) - d(p^n||p^{\infty};\Theta) \tag{22}$$

[1]The term $((\partial P^n)/(\partial \Theta_{kl}^i))((\partial d(P^n||P^{\infty};\Theta^i))/(\partial P^n))$ is neglected [21].

with

$$d(p^n||p^\infty;\Theta) := \sum_{\boldsymbol{v}^{in}} \int_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} P^0(\boldsymbol{v}^{in}) p^n(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta) \cdot \log\left(\frac{p^n(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)}{p^\infty(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)}\right) d\boldsymbol{v}^{\text{out}} \quad (23)$$

and $p^0(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta) := p^0(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in})$. In case of a finite input sample $(\boldsymbol{x}_n)_{n=1}^N$, empirical distributions $P^0(\boldsymbol{v}^{in}) = (1/N)\sum_{n=1}^N \delta(\boldsymbol{v}^{in}-\boldsymbol{v}^{in}(\boldsymbol{x}_n))$ and $p^0(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in}(\boldsymbol{x}_n)) = \delta(\boldsymbol{v}^{\text{out}}-\boldsymbol{t}_n)$ are assumed. This modification yields the following approximate partial derivatives with respect to the parameters Θ (for details, see Appendix):

$$\frac{\partial CD_n^{IO}(\Theta)}{\partial\Theta_{kl}} \approx \left\langle\left\langle \frac{\partial H(\boldsymbol{s};\Theta)}{\partial\Theta_{kl}}\right\rangle_{p^\infty(\boldsymbol{h}|\boldsymbol{v}^{in},\boldsymbol{v}^{\text{out}};\Theta)}\right\rangle_{p^0(\boldsymbol{v}^{in},\boldsymbol{v}^{\text{out}})} - \left\langle\left\langle \frac{\partial H(\boldsymbol{s};\Theta)}{\partial\Theta_{kl}}\right\rangle_{p^\infty(\boldsymbol{h}|\boldsymbol{v}^{in},\boldsymbol{v}^{\text{out}};\Theta)}\right\rangle_{p^n(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)P^0(\boldsymbol{v}^{in})}. \quad (24)$$

Again, the first expectation in (24) is readily available. Fortunately, the estimation of the second expectation with free varying output states and input states clamped throughout is also substantially reduced using CD_1^{IO}. While the input and output units are clamped, i.e., $\boldsymbol{v}^{in} = \boldsymbol{v}^{in}(\boldsymbol{x}_n)$ and $\boldsymbol{v}^{\text{out}} = \boldsymbol{t}_n$, hidden states $\hat{\boldsymbol{h}}$ are sampled according to a normal distribution

$$p^\infty(h_l|\boldsymbol{v}^{in},\boldsymbol{v}^{\text{out}};\Theta) = \mathcal{N}\left(\mu_l^h,(\sigma_l^h)^2\right) \quad (25)$$

with variance $(\sigma_l^h)^2$ and mean

$$\mu_l^h = b_l^h + \sigma_l^h(\boldsymbol{v}^{in})^T\mathbf{W}_{(\cdot)l}^{in} + (\sigma_l^h)^2(\boldsymbol{v}^{\text{out}})^T(\Sigma^{\text{out}})^{-\frac{1}{2}}\mathbf{W}_{(\cdot)l}^{\text{out}}.$$

Then, a reconstruction $\boldsymbol{v}^{rec}$ is sampled with fixed hidden states $\boldsymbol{h} = \hat{\boldsymbol{h}}$ according to the normal distribution

$$p^\infty(v_k^{\text{out}}|\boldsymbol{h};\Theta) = \mathcal{N}\left(\mu_k^{\text{out}},(\sigma_k^{\text{out}})^2\right) \quad (26)$$

with variance $(\sigma_k^{\text{out}})^2$ and mean $\mu_k^{\text{out}} := b_k^{\text{out}} + \sigma_k^{\text{out}}\mathbf{W}_{k(\cdot)}^{\text{out}}\boldsymbol{h}$, vice versa, the hidden states are updated by simulation from (25) with $\boldsymbol{v}^{\text{out}} = \boldsymbol{v}^{rec}$ clamped at the output units. This procedure is repeated until the full sample $(\boldsymbol{x}_n)_{n=1}^N$ is processed. Consequently, if the density $p^1(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in}(\boldsymbol{x}_n);\Theta^*)$ reflects sufficiently well the empirical density $p^0(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in}(\boldsymbol{x}_n))$ for all $1 \le n \le N$ at a minimizer Θ^* of (22), then this implies also a minimum MSE between the desired target codes $\boldsymbol{t}_n$ and a linear output model $\boldsymbol{\mu}^{\text{out}} = \boldsymbol{b}^{\text{out}} + (\Sigma^{\text{out}})^{(1/2)}\mathbf{W}^{\text{out}}\boldsymbol{h}$. Similarly, Θ^* maximizes Q_h in the space spanned by the hidden layer of the output RBM.

C. Supervised Fine-Tuning with Global Weighting

Semisupervised pre-optimization alone is not optimal for maximizing Q_h. Subsequent supervised fine-tuning is necessary for learning an optimal feature extractor $f^* \in \mathcal{F}$.

After pre-optimization, the topmost RBM's output units are discarded to obtain a GerDA DNN up to the last hidden layer (Fig. 1): that is, the feature extraction layer. The remaining parameters $(\boldsymbol{b}^{h^1},\ldots,\boldsymbol{b}^{h^L},\boldsymbol{b}^h,\mathbf{W}^1,\ldots,\mathbf{W}^L,\mathbf{W}^{in})$ serve as the starting point for the fine-tuning using a conjugate gradient ascent algorithm with gradients of Q_h computed via a modified backpropagation (for details, see Appendix).

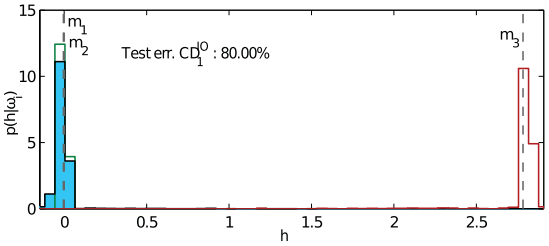

Fig. 2. Class-conditional distributions $p(h|\omega_i)$ of 1-D GerDA feature values of the three-class Swiss-roll data (Fig. 3) after fine-tuning without using the global weighting heuristics. Pre-optimization was performed using CD_1^{IO}. Vertical lines mark the class means m_i, and the resulting test error also is given.

To further improve discrimination between different classes, we developed an iterative weighting. It is well known that the discriminant function Q_h overemphasizes large distances to the disadvantage of neighboring classes causing a large overlap and therefore inferior discrimination. A simple but effective way to resolve this drawback of Q_h is to reduce the influence of classes on the between-class scatter matrix $\mathbf{S}_B$ in proportion to the distance of their class means. Therefore, a GerDA DNN is fine-tuned with a globally weighted discriminant criterion

$$Q_h^\delta(f) := \text{trace}\left\{(\mathbf{S}_T^\delta)^{-1}\mathbf{S}_B^\delta\right\} \quad (27)$$

with weighted total scatter matrix $\mathbf{S}_T^\delta := \mathbf{S}_W + \mathbf{S}_B^\delta$. The matrix $\mathbf{S}_W$ is the common within-class scatter matrix $S_W := (1/N)\sum_{i=1}^C N_i\Sigma_i$ of the class covariances $\Sigma_i := (1/N_i)\sum_{n:\omega_i=\omega(\boldsymbol{x}_n)}(\boldsymbol{h}_n-\boldsymbol{m}_i)(\boldsymbol{h}_n-\boldsymbol{m}_i)^T$. The matrix $\mathbf{S}_B^\delta$ is the weighted between-class scatter matrix

$$\mathbf{S}_B^\delta := \frac{1}{2N^2}\sum_{i,j=1}^C N_iN_j\cdot\delta_{i,j}\cdot(\boldsymbol{m}_i-\boldsymbol{m}_j)(\boldsymbol{m}_i-\boldsymbol{m}_j)^T \quad (28)$$

with an appropriate symmetric weighting $\delta_{i,j}$. In particular, for $\delta_{i,j} = 1\forall 1 \le i,j \le C$, one easily verifies that $\mathbf{S}_B^\delta = \mathbf{S}_B$.

We used as weighting

$$\delta_{i,j} := \begin{cases} 1/\|\boldsymbol{m}_i-\boldsymbol{m}_j\|^2 & \text{if } i \ne j \\ 0 & \text{if } i = j. \end{cases}$$

After each update step of the gradient ascent

$$\Theta_{kl}^i \longleftarrow \Theta_{kl}^i + \eta\cdot\frac{Q_h^\delta(f)}{\partial\Theta_{kl}^i},\ \eta\in(0,1] \quad (29)$$

the weights $\delta_{i,j}$ are recomputed and the scatter matrices $\mathbf{S}_B^\delta$ and $\mathbf{S}_T^\delta$ are updated accordingly.[2] Our experiments show that the proposed global weighting improves the quality of the feature extraction substantially, in particular for feature spaces of a very low dimension $r \ll (C-1)$. Fig. 2 shows the 1-D GerDA feature values of the Swiss-roll data (Fig. 3) and the associated test error when global weighting was not used, in contrast to Fig. 5 (top) where global weighting was used.

[2]Note that the weighted discriminant criterion is not scale invariant. But $\delta_{i,j}$ can be multiplied with $\min_{k,l;k\ne l}\{\|m_k-m_l\|^2\}$ for scale invariance.

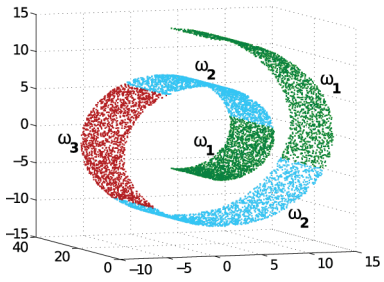

Fig. 3. Three classes ω_1, ω_2, and ω_3, lying on a Swiss-roll manifold. Note that ω_1 and ω_2 are each split into two discontiguous regions.

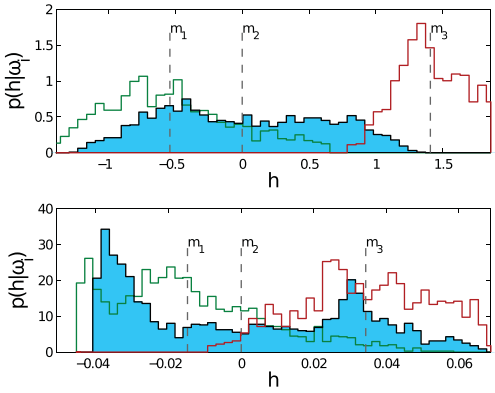

Fig. 4. Class-conditional distributions $p(h|\omega_i)$ of 1-D GerDA feature values of the three-class Swiss-roll test data (Fig. 3) after pre-optimization with CD_1^{IO} (top) and CD_1 (bottom). Vertical lines mark the class means m_i.

III. Experiments

A. Does CD_1^{IO} Improve GerDA Fine-Tuning?

The partly unsupervised and supervised pre-optimization improves fine-tuning of Q_h^δ in two ways: the unsupervisedly trained part of the RBM stack acts like a regularizer restricting the optimization to appropriate regions in the space $\mathcal{F}$ of feature extractors, the supervisedly trained top-most RBM restricts $\mathcal{F}$ further and improves the fine-tuning, because it provides starting points near good local maxima of Q_h^δ.

While the regularization effect has been elaborately discussed in [13], we investigate the effect of supervised pre-optimization of the topmost RBM using CD_1^{IO} compared to an unsupervised CD_1 pre-optimization. Recall that, using CD_1 means no output units are required in the topmost RBM because training is performed unsupervisedly. For this purpose, each RBM of a GerDA DNN up to the output RBM were trained using the unsupervised CD_1 heuristics. Then, we examined the effect on the subsequent fine-tuning through pre-optimizing the output RBM with the supervised CD_1^{IO} compared to the unsupervised CD_1 method. For a first experiment, we used an artificial three-class dataset lying on a Swiss-roll manifold (Fig. 3).

For mapping the 3-D Swiss-roll data to a 1-D feature space, a GerDA DNN with topology 3-**40-20-10**-1[3] was used. Fig. 4 shows the class-conditional distributions of the extracted GerDA features after pre-optimizing the output RBM with CD_1^{IO} and CD_1, respectively. After pre-optimization, the classes tend to drift away from each other in both cases. However, noting the distances between the class means m_i, this repulsion effect is stronger in the case of CD_1^{IO}. Moreover, the data distributions better concentrate around their means, and the shape is closer to unimodality. Thus, we conclude that CD_1^{IO} provides better initial conditions for a subsequent fine-tuning. Although fine-tuning was performed identically for the CD_1^{IO} and CD_1 pre-optimized GerDA-DNNs, CD_1^{IO} results in a lower test error after fine-tuning,[4] namely, CD_1^{IO} : 0.34% and CD_1 : 0.72%. The class-conditional distributions of the GerDA features after fine-tuning are depicted in Fig. 5. Again, CD_1^{IO} pre-optimization is superior to CD_1 with respect to

[3]Sizes of the DNN layers: *bottom layer–hidden layer 1–· · ·–top layer.*

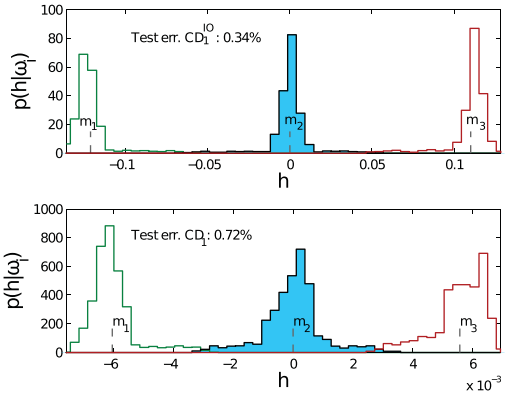

Fig. 5. Class-conditional distributions $p(h|\omega_i)$ of 1-D GerDA feature values of the three-class Swiss-roll test data (Fig. 3) after fine-tuning using CD_1^{IO} (top) and CD_1 (bottom) pre-optimization. Vertical lines mark the class means m_i, and resulting test errors are also given.

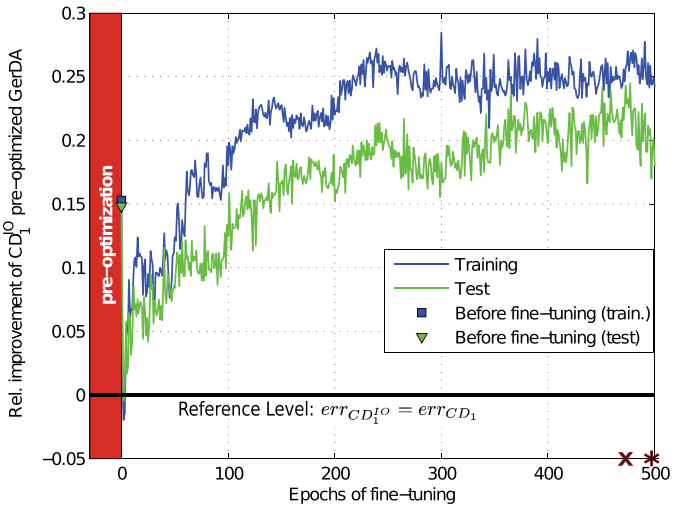

Fig. 6. Relative improvement of organic acid prediction [22] by CD_1^{IO}. The reference level corresponds to equal error rates (EERs) from CD_1^{IO} and CD_1. The **relative improvement**[5] by CD_1^{IO} is shown for 500 fine-tuning epochs of a GerDA DNN pre-optimized with CD_1^{IO} and CD_1, respectively. The test errors before fine-tuning (epoch 0) are 44.35% and 52.34%, respectively. The lowest test error is 19.39% for CD_1^{IO} (epoch marked with $\times$) and 23.90% for CD_1 (epoch marked with $*$).

[4]Classification error of the DNN-transformed test samples (features), computed by a linear minimum (Euclidean) distance (to means) classifier (see [23] pp. 306–307) trained with the DNN-transformed training samples.

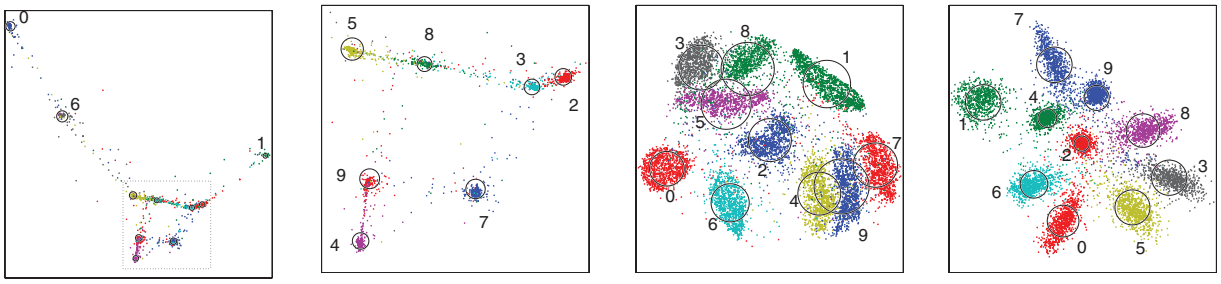

Fig. 7. Comparison of 2-D mappings obtained using GerDA, t-SNE, and NNCA on the MNIST test images. The class centers are marked with circles and the associated digits. The radii of the circles correspond to the mean distance of the 2-D data points to their corresponding class centroid. Classification results for the 2-D features are given for t-SNE and GerDA only, because [8] does not state the classification error.

feature concentration around their class-means m_i and the unimodality of the distribution's shape.

Further justification for CD_1^{IO} gives an experiment on a dataset of UV/vis spectroscopic online measurements for the prediction of organic acid concentrations in biogas plants. As reported in [22], GerDA features and linear prediction of the concentrations outperform nonlinear support vector machines (SVMs) on this task. The GerDA features from this experiment cannot be visualized. We show the relative improvement[5] by CD_1^{IO} pre-optimization recorded over 500 epochs of fine-tuning, with the CD_1 pre-optimized GerDA DNN as reference level (Fig. 5). The CD_1^{IO} pre-optimization improves the fine-tuning compared to its CD_1 counterpart significantly in terms of both the training and test error. CD_1^{IO} also achieves the lower test error (19.39% versus 23.90% with CD_1), and the increasing trend of the training curve in Fig. 5 implies a faster convergence rate of the fine-tuning after CD_1^{IO} pre-optimization.

B. Dimensionality Reduction and Data Visualization

Dimensionality reduction is a primary objective of most methods for discriminant analysis [6]. For visualization, the target space has 1–3 dimensions. Mapping a high-dimensional data space onto a 2-D scatterplot is a particularly interesting way of analyzing the structure of this data space and/or the properties of the dimensionality reduction algorithm. We analyze GerDA's abilities of dimensionality reduction on multiclass data with the MNIST database of handwritten digits,[6] which is a standard benchmark.

Unsupervised pre-optimizing of the GerDA network was carried out using all 60k training samples. That is, we first trained all RBM layers, except for the last one, with all available training samples, but not using the class labels. Then the last RBM layer was trained with the supervised CD^{IO} method, not using any of the samples from the validation dataset. With a few trials on different network topologies, the topology 784-**1500-375-750**-2 was selected as giving the lowest linear classification error on the 2-D features of the validation set. The best MNIST 28 × 28 to 2-D mapping, found with GerDA on the basis of the validation error, is shown in Fig. 7(a) for the 10k test images. The globally weighted discriminant criterion (see Section II-C) causes the easily separable classes to be projected quite far away from the group of harder to separate classes. Therefore we show the latter group of classes again as enlarged detail in Fig. 7(b). The class centroids are marked with circles whose radii correspond to the mean distance of the samples to their corresponding class centroid in two dimensions. The *accuracy* of the mapping (ACC), measured as 100% *minus linear test error*, is 96.8% for the mapping found with GerDA. We compare our result with two other advanced dimensionality reduction methods, for which we could also obtain a 2-D scatterplot of the MNIST test data.

Parametric t-SNE is a new unsupervised dimensionality reduction technique recently introduced by van der Maaten [25]. Fig. 7(c) shows the 2-D scatterplot of the MNIST test data obtained with t-SNE trained on the MNIST training data. Like GerDA, t-SNE is based on DNNs with RBM pretraining and fine-tuning using backpropagation, but the class labels are not used during training. The digit classes in Fig. 7(c) are surprisingly well separated, considering the fact that the mapping was learned with an unsupervised method. We show the t-SNE result as a baseline and also in order to test the so-called distance consistency measure (DSC) for quantifying class separation in a scatterplot. In the context of visualization, "class consistency" and its measure DSC have been proposed for calculating the quality of mappings from high-dimensional data spaces to lower dimensional views [26]. We use DSC alongside the error or accuracy of the classification on the 2-D features. DSC = 100% means that all data points have a smaller Euclidean distance to their corresponding class centroid than to any of the other class centroids. When DSC is calculated on the projected data points only, it is a good measure of (visual) class separation in the absence of any knowledge on the original data space. For the scatterplots in Fig. 7, the DSC of the results from GerDA and t-SNE is 96.83% and 88.99%, respectively.

Among the various methods for dimensionality reduction that GerDA has to compete with, NNCA [8] seems to give particularly good results. According to [8], NNCA achieves 1.0% error rate on the MNIST test data with 3-NN classifi-

[5]Relative improvement is measured as $1-err_{CD_1^{IO}}/err_{CD_1}$, i.e., it is zero if both error rates are the equal and positive for a higher error of CD_1.

[6]The MNIST database [24] contains grayscale images of handwritten digits with 28 × 28 pixels resolution, i.e., 784 real-valued features. The whole dataset is divided into 60k training and 10k test images. For validation, we randomly split the training set into a 55k training set and a 5k validation set.

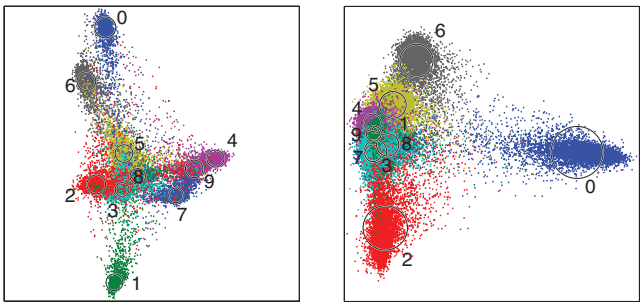

Fig. 8. Comparison of 2-D mappings obtained using GerDA and MKFD on the **MNIST10k** test images.

cation in a 30-D feature space. This is to date the best error rate reported for the MNIST recognition task when absolutely no domain-specific knowledge is used [24]. Since we could not get access to an implementation of NNCA for doing our own experiments, we reproduced and analyzed the MNIST 2-D scatterplot from the embedded postscript in [8, Fig. 4].[7] In Fig. 7(d), the NNCA scatterplot is shown with the overlaid circles for the class centroids. Although the classes seem to be very well separated, it is clear that they are represented less compactly than in the scatterplot produced by GerDA. The DSC value of 95.03% calculated from Fig. 7(d) implies that GerDA (DSC = 96.83%) yields a significantly lower classification error rate than NNCA for the MNIST 28×28 to 2-D mapping.

Kernel Fisher discriminant (KFD) analysis is a well-established method for finding a nonlinear dimension-reducing transformation. Invented for two-class discrimination by Mika [3], it can easily be generalized for multiclass dimensionality reduction. The multiclass KFD (MKFD) is an obvious choice for experimental comparisons with GerDA. We used the MATLAB KFD by Franc *et al.* (http://cmp.felk.cvut.cz/cmp/software/stprtool/) and extended the code for multiple classes. Unfortunately, the main memory needed by the KFD scales quadratically with the number of training samples. It turns out that the MNIST experiment cannot be completed in a practical time frame unless a minimum of 80 GB of main memory is available. For this reason, we present a couple of experiments that compare GerDA with MKFD on the "MNIST10k" dataset advocated by Li [27]. MNIST10k is just the MNIST dataset but the 10k test images are used for training and, vice versa, the 60k training images are treated as test data. The error rates achieved by Li [27] with five different algorithms lie between 23.25% and 3.50%, indicating that MNIST10k is a relatively hard learning task.

The 10k training images of MNIST10k were randomly split into a training and a validation set with about the same relative proportions chosen for MNIST. With the validation set, we optimized the MKFD parameters for the RBF kernel and the regularization. The 28×28 to 2-D mapping obtained with MKFD, for the parameter values at the lowest validation error, is shown in Fig. 8(b) on the 60k test images. The corresponding ACC is 76.62% and DSC is 76.35%. For an analogous experiment with GerDA, we first deployed a semiautomatic method for finding the best topology.[8] The best mapping obtained with the topology 784-**787-280-1201**-2 is shown in Fig. 8(a) (ACC = 92.10%, DSC = 93.89%).

For the mapping of MNIST10k onto a 2-D scatterplot, we had expected a better result from the MKFD. Therefore we also compared GerDA and MKFD based on mappings to nine ($= C - 1$, c.f. Section I) dimensions. The 9-D feature space learned by the MKFD turned out to be excellent for (linear) classification, yielding a test error rate of 3.34%, compared to 3.97% and 3.50% reported for robust logitboost and abc-boost, respectively, in [27]. We did the same experiment also with a recently introduced kernel method, i.e., kernel spectral regression (KSR)[9] [5]. After an extensive parameter scan, KSR achieved an error rate of 3.14% (RBF kernel, $\alpha = 0.32$, $t = 5$). In comparison, with GerDA we achieved 3.61%.

C. Classification of Truly High-Dimensional Data

Hinton and Salakhutdinov [10] characterize the 784-D MNIST images as "high-dimensional data." However, for many practical applications the data dimensions are much higher [29]. We wanted to study how GerDA copes with input dimensions that are at least 10 times higher than the data dimensions mostly used in publications on DNNs.

When dealing with image data, such as the MNIST digits or face images, it is well known that preprocessing with a multi-scale filter bank can greatly improve the performance of recognition and detection schemes [30], [31]. Image preprocessing with a comprehensive filter bank also enormously increases the number of "raw measurements." For these reasons, we again chose the MNIST image data for an experiment, but this time with a log-Gabor preprocessing stage for which a good MATLAB implementation is available from Peter Kovesi [32]. As a modification of the original implementation, we used subsampling on the log-Gabor representation on all but the level of highest spatial resolution, similar to [30, p. 532].

The original MNIST images are of size 28×28 pixels, thus three filter scales are sufficient for a comprehensive representation. Without any attempt to optimize the parameters of the log-Gabor representation for this data, we chose the number of filter orientations to be 8. The resulting input vectors for the GerDA network had a dimension of 8232 ($28^2 \times 8 + 14^2 \times 8 + 7^2 \times 8$). The GerDA network topology used in this experiment was 8232-**280-1201**-9. This network has only two hidden layers because we expect the Gabor preprocessing stage to effectively function as a convolutional input layer of the network (c.f. [33]). Hence, one obvious candidate topology for this experiment was the topology used in the MNIST10k experiments (784-**787-280-1201**-2) with the first hidden layer removed. Since we defined our goal to be a

[7]This data consists of exactly 8000 (10 times 800) 2-D feature points. However, the text associated with this figure in [8] suggests that all 10 000 MNIST test images were used for generating this scatterplot. The authors of [8] could not be reached for a comment on this inconsistency.

[8]Related to the work by Islam *et al.* [28], subject of a future publication.

[9]The MATLAB KSR is available from the authors of [5] under www.zjucadcg.cn/dengcai/SR/. Unfortunately, a mapping to two dimensions is not possible with this implementation.

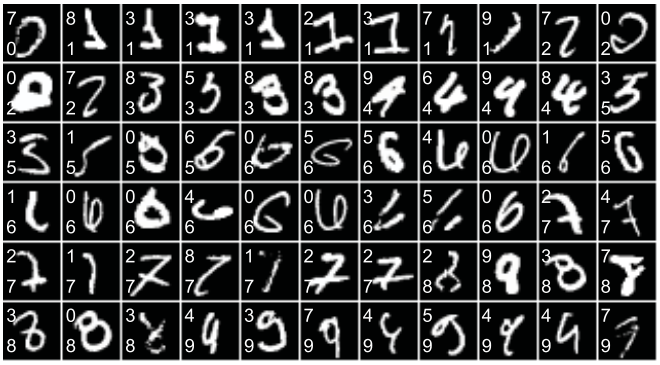

Fig. 9. Misclassified MNIST test images. The correct class labels are shown in the lower left corners, and the predicted class labels in the upper left corners.

small classification error for the 10 digits while keeping the feature dimensions as low as possible, the target dimensions of the network was set to 9.

For the unsupervised part of the DNN pre-optimization, all 60k training samples were used. Then, supervised pre-optimization and network fine-tuning was carried out, not using the validation samples. The network status was saved at the point of minimal validation error. At this point, the fine-tuning was continued with all 60k training samples until the value of the training error again reached the previously measured value at minimal validation error. With the final GerDA DNN, all test images were transformed to 9-D features vectors. Linear classification (see footnote 4) of the 9-D test vectors misclassified a total of 66 images of the MNIST test set (0.66%), as shown in Fig. 9.

The result obtained with GerDA has to be discussed in the context of the publications by Ranzato *et al*. [34], Labusch *et al*. [35], and Jarrett *et al*. [33]. Their best results on MNIST are 0.60%, 0.59%, and 0.53%, respectively.[10] Even better results have been achieved by using an expanded training set, i.e., additional training images that are artificially distorted or translated versions (jittering) of the original training images [24]. Artificial training images have not been used for our experiment and the three other results mentioned above.

Except for the fact that the input data are images, our convolutional preprocessing stage does not use any domain-specific knowledge. A multiscale Gabor representation is a generic image preprocessing step. Using it for images like the MNIST digits may even be regarded as rather unconventional since Gabor filters are known to be particularly effective for natural images. In contrast, [33]–[35], deploy one or more layers of convolution and pooling operators that are adapted to the specific pattern domain by design.

Another fundamental difference between [33]–[35], and our work is the goal of the training process. We wanted to obtain an effective feature extractor, in this case for 9-D features. Subsequent operations on these features are not confined to classification. In contrast, the classification results of [34], [35], and [33] are obtained by networks that are trained/designed specifically for classification. In case of [35], this classifier is even composed of 45 SVMs that jointly work on a 160-D feature vector.

[10]On MNIST, differences of more than 0.1% are statistically significant [11].

D. Sensor Fusion: TOF Camera Data

This section is on an experiment with face detection on intensity and range image data. We have become interested in the so-called time-of-flight (TOF) cameras because of the new chances this technology opens up for 3-D scene analysis and other fields of research [36]. A TOF camera produces an intensity and a range image simultaneously. With TOF cameras, 3-D objects can be detected and recognized not only by the reflectance properties of a visible surface but also by a 2.5-D representation[11] of that surface. Fusing the intensity and the range data for the improvement of object recognition is a challenging task. Specifically for TOF face detection, pioneering work has been done by Böhme *et al*. [37]. In general, a detection problem can be treated either as a binary classification problem or as dimensionality reduction to 1-D feature values with subsequent thresholding. The latter has the advantage that it should naturally produce a quantitative membership measure for the pattern class to be detected.

A dataset for TOF face detection is available from the authors of [37] and the ARTTS project (www.arrts.eu).[12] The approach to face detection taken in [37] is based on the well-known Viola–Jones algorithm and related methods [38]. It consists of a cascade of boosted classifiers using simple spatial image features for face detection. Fusion of intensity and range information is achieved by combining image features from both channels. We used log-Gabor wavelets as image features, similar to what we did in the experiment described in Section III-C. As a modification to the original log-Gabor implementation from [32], we added the option of locally normalizing the filter outputs about all filter directions. This is a proven means to reduce the influence of the lighting conditions for real-world camera images [31]. Let $G_{f,\theta}(x, y) \neq 0$ be the magnitude of the complex filter response at an image position (x, y) for given spatial frequency f and orientation θ, Θ denoting the set of all computed filter orientations, then the normalized filter response used in our implementation is

$$Q_{f,\theta}(x, y) = \frac{G_{f,\theta}(x, y)}{\sum_{\alpha \in \Theta} G_{f,\alpha}(x, y)}. \tag{30}$$

Deploying the state-of-the-art data preprocessing in a way one would (should) do for a real application, the log-Gabor representation was used for both the intensity and the range image, for the former with the normalization (30). In the range image there is no lighting problem. We also performed re-cropping and scaling on the TOF face images. The ARTTS face data come as $2 \times 24 \times 24$ pixel values in the range

[11]A simplified parametrization (x, y, z) of a surface using depth z.

[12]The training set contains 5412 faces and 3486 "noface" images, the validation set 1752 faces and 1145 noface images, and the test set 534 faces and 349 noface images. The face images are cropped and resized to 24×24 pixels. The full-frame noface images (176×144 pixels) are guaranteed not to contain any face patterns. They are provided for the purpose of generating "noface" samples (negative face samples). Details can be found in [37].

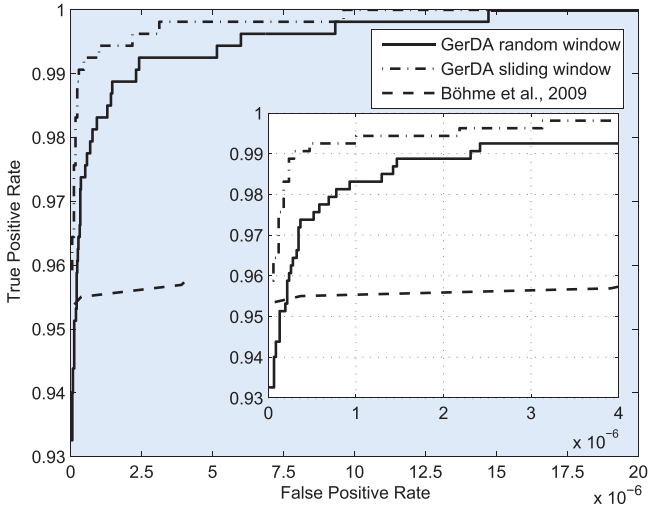

Fig. 10. Receiver operating characteristics (ROC) of a GerDA-based TOF face detector in comparison to the ROC of the detector proposed in [37]. The "sliding window" and the "random window" test data contain ca. 17 millions and ca. 46 millions "noface" images, respectively.

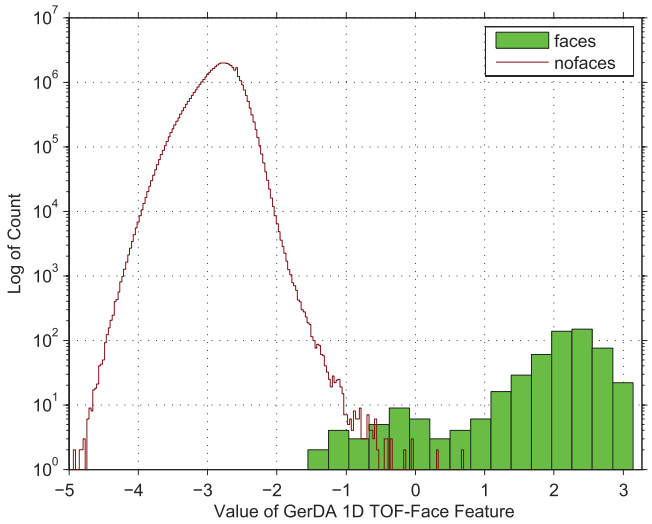

Fig. 11. Distribution of the values of the 1-D features produced by the GerDA TOF face detector from the "random window" test data. The (binned) features of the 534 faces are marked by bars. The ca. 46 million nofaces are marked by points forming a relatively smooth curve. The y-axis has logarithmic scale.

$0, \ldots, 2^{16}$–1. We re-cropped all face images to 20 × 20 pixels. This subregion was vertically centered. In the horizontal direction, the center was chosen at the minimum of the horizontal projection of the range image (see [23, pp. 256–257]).[13] By this way, we avoided cutting off parts of the face in the subregion. On face images, this effectively is a horizontal adjustment to the approximate position of the nose. Of course, the same operation was performed on all 24 × 24 noface image patches likewise. Finally, we scaled the face (noface) images to 16 × 16 pixels, a size often used for real-time face detectors. For this image size, only two scales are needed in the log-Gabor representation. With the standard eight-filter orientations, the resulting input vectors for the GerDA network had 5120 $[2 \times (16^2 \times 8 + 8^2 \times 8)]$ dimensions.

The real hard challenge in building face detectors is a false-positive rate (FPR) close to zero. For this reason, in [37] the accuracy of the face detector was measured by an ROC curve that shows the FPR only up to a value of 4×10^{-6}. We also concentrated our experiment on achieving the lowest possible FPR on the validation and test data. However, in GerDA the EER is implicitly minimized. Therefore, we cannot expect optimal results if the FPR is actually the more important minimization goal. We solved this problem by a kind of "bootstrap" training strategy [39], better described as an iterative aggregation of the training set.[14] We trained a few different network topologies and selected the 5120-**1000**-**500**-1 topology by reason of its performance on the validation data.

Measuring a value of FPR in an order of magnitude of 10^{-6} requires millions of negative validation samples (test samples). This data had to be generated from the 1145 (349) noface images of size 176 × 144 pixels. For this purpose, we implemented a *generator* program that produces 132 912 different 24 × 24 patches from a single full-frame noface image by scaling, rotation, mirroring, and cropping. For the aggregation of the training set, in every iteration we randomly fed patterns from the generator to the previously trained network until a preselected number of difficult samples was collected.

Fig. 9 shows the final results of our aggregative training process with GerDA, alongside the result published in [37]. One ROC curve was computed with the GerDA network using 171 055 37 noface samples generated from the 349 noface test images by the common multiscale sliding window technique [39], [40], presumably similar to how this was done in [37]. At an FPR = 0 our TPR is 95.88% versus 95.3% reported in [37]. While these two results are basically the same, the GerDA ROC clearly outperforms the ROC from [37] for FPR > 0. Another ROC curve was computed using noface test samples randomly generated from the noface test images. These samples included mirrored and rotated patches and their total number was 463 862 88. For this harder test set, our TPR at FPR = 0 is 93.2% and the overall run of the ROC is slightly worse than the ROC based on the sliding window technique. Fig. 11 shows the distributions of the 1-D feature values for the noface (random window) and the face test samples. The counts are displayed on a logarithmic axis because of the huge difference between the number of noface samples and face samples. The x-axis shows the raw feature values obtained from computing the trained DNN on the 5120-D log-Gabor representation.

IV. Discussion and Conclusion

We have presented an approach to feature extraction with deep neural networks. At its core is a generalized

[13] If the minimum of the horizontal projection was more than two pixels off the image center, no adjustment of the horizontal center position was done.

[14] For the ARTTS dataset, our *aggregation* technique worked as follows. The initial training set consisted of all positive training samples (faces) and about the same number of negative samples (nofaces) generated from the full-frame noface training images. The validation set was constructed similarly. In each training iteration, the previously trained network was used to add a few replicates of "difficult" positive samples and significantly more new and also difficult negative samples to the final training set being aggregated.

discriminant analysis (GerDA). Our work was motivated by the idea that discriminative features having similar properties to features sought by classical LDA could be obtained from arbitrarily distributed raw measurements by some complex nonlinear transformation. We proposed a novel procedure of training DNNs so as to easily arrive at a good solution for the nonlinear transformation that provides features having all the desirable properties of LDA features from a linearly separable dataset. This transformation is not limited to dimension reduction [14].

For a similar goal, kernel methods for generalizing the classical discriminant analysis have been proposed [3], [6]. Our experiments with the MKFD and the KSR (Section III-B) tell us that the classification performance of these kernel methods is excellent on small multiclass datasets. As the size of the training set increases, kernel methods become intractable in practice, although methods have been proposed to lessen the problem [41], [42]. With our computers, we cannot train the standard MNIST classification benchmark because the memory requirements of kernel methods, e.g., MKFD and KSR, scale quadratically with the number of training samples (typically, the computational complexity scales cubically). Even in the application phase, the computational complexity and the memory requirements depend on the number of training samples. With neural networks, this only depends on their topology. As an example, the DNN used in Section III-C (8232-**280-1201**-9) for an experiment with 60 000 training samples needs ca. 2.6 million multiply–add operations and 1481 computations of a sigmoid function. With modern processors, this poses no problem for real-time applications. The memory space required for this network is ca. 11 MB.

In our experiments with GerDA, we demonstrated the success of our approach for the task of dimensionality reduction and data visualization on the popular MNIST database of handwritten digits. The 2-D feature mapping of the MNIST digit data found by GerDA is the best reported in the literature, in terms of linear classification error. By conducting an experiment on face detection from simultaneously acquired intensity and range images, we tested GerDA with a challenging sensor fusion task. The face detector generated with GerDA was found to outperform a recently published approach based on the popular Viola–Jones algorithm and the AdaBoost algorithm [37]. One particular virtue of using GerDA for detection tasks is the genuine continuity of the membership measure for the object/event class. This is a requirement in biometric applications, for example, where the tradeoff between false rejection rate and false acceptance rate must be a configuration parameter of the system.

In another experiment, we demonstrated that GerDA can readily be applied to very high dimensional input data. With GerDA we obtained a transformation that maps a 8232-D log-Gabor representation of a MNIST digit image to a 9-D feature space in which the test digits can be linearly classified with an accuracy of 99.34%.

GerDA belongs to a class of methods that integrate unsupervised and supervised learning (c.f. [8], [43], [4]). For the experiments described in Section III, this was exploited by using all available training data for the pre-optimization of the DNN, whereas the supervised part of the training always required a part of the training data for validation. However, for a thorough assessment of the potential of semisupervised training with GerDA, systematic experiments for this purpose need to be done, e.g., as in [44].

It is striking that both learning a 2-D feature mapping and learning a pattern detector (1-D feature mapping) seem to give better results with GerDA than with competing methods. We attribute this to the global weighting scheme introduced in Section II-C. However, this has to be analyzed in depth by future work. Moreover, other variations of the objective function could be investigated. GerDA may be viewed as a DNN framework for finding nonlinear data mappings that optimize a certain objective function in feature space. As such, it can readily be used to build a nonlinear realization for other objective functions that have proven to be useful in linear realizations, e.g., the joint maximization of marginal sum of negentropy and Fisher discriminant score, recently proposed by Dhir and Lee [45].

Other promising work with GerDA for the future include search applications in large databases where a simple linear classifier operating on low-dimensional discriminative features often is the only feasible solution.

APPENDIX

A. Derivation of CD_n^{IO}

For the sake of brevity, as long as there is no confusion about the arguments that are affected by the involved operations (summation, integration, differentiation), we omit the arguments in what follows:

$$\begin{aligned}
&\frac{\partial CD_n^{IO}(\Theta)}{\partial \Theta_{kl}} \\
&= \frac{\partial}{\partial \Theta_{kl}} \sum_{\boldsymbol{v}^{in}} P^0 \int\limits_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} \left(p^0 \cdot \log\left(\frac{p^0}{p^\infty}\right) \right. \\
&\quad \left. - p^n \cdot \log\left(\frac{p^n}{p^\infty}\right) \right) d\boldsymbol{v}^{\text{out}} \\
&= \sum_{\boldsymbol{v}^{in}} P^0 \int\limits_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} \left(p^0 \left(-\frac{\partial \log p^\infty}{\partial \Theta_{kl}} \right) \right. \\
&\quad \left. - \left(\frac{\partial p^n}{\partial \Theta_{kl}} \log\left(\frac{p^n}{p^\infty}\right) + p^n \frac{\partial \log\left(\frac{p^n}{p^\infty}\right)}{\partial \Theta_{kl}} \right) \right) d\boldsymbol{v}^{\text{out}} \\
&= \sum_{\boldsymbol{v}^{in}} P^0 \int\limits_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} \left(-\frac{p^0}{p^\infty}\frac{\partial p^\infty}{\partial \Theta_{kl}} - \frac{\partial p^n}{\partial \Theta_{kl}} \log\left(\frac{p^n}{p^\infty}\right) \right. \\
&\quad \left. - \left(\frac{p^n}{p^n}\frac{\partial p^n}{\partial \Theta_{kl}} - \frac{p^n}{p^\infty}\frac{\partial p^\infty}{\partial \Theta_{kl}} \right) \right) d\boldsymbol{v}^{\text{out}} \\
&= \sum_{\boldsymbol{v}^{in}} P^0 \int\limits_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} \frac{p^n}{p^\infty}\frac{\partial p^\infty}{\partial \Theta_{kl}} - \frac{p^0}{p^\infty}\frac{\partial p^\infty}{\partial \Theta_{kl}} \, d\boldsymbol{v}^{\text{out}} \\
&\quad - \sum_{\boldsymbol{v}^{in}} P^0 \int\limits_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} \frac{\partial p^n}{\partial \Theta_{kl}} \left(1 + \log\left(\frac{p^n}{p^\infty}\right) \right) d\boldsymbol{v}^{\text{out}}. \quad (31)
\end{aligned}$$

Neglecting the last addend (which is usually very small, see the CD_n-heuristic [21]), the approximate derivative reads as

$$\frac{\partial CD_n^{IO}(\Theta)}{\partial \Theta_{kl}} \approx \sum_{\boldsymbol{v}^{in}} P^0(\boldsymbol{v}^{in}) \int_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} \left(\frac{p^n(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)}{p^\infty(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)} - \frac{p^0(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)}{p^\infty(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)} \right) \frac{\partial p^\infty(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)}{\partial \Theta_{kl}} \, d\boldsymbol{v}^{\text{out}}. \tag{32}$$

With

$$p^\infty(\boldsymbol{v}^{\text{out}}, \boldsymbol{v}^{in};\Theta) = \int_{\mathbb{R}^r} p^\infty(\boldsymbol{s};\Theta) \, d\boldsymbol{h} \tag{33}$$

$$p^\infty(\boldsymbol{v}^{in};\Theta) := \int_{\mathbb{R}^r} \int_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} p^\infty(\boldsymbol{s};\Theta) \, d\boldsymbol{v}^{\text{out}} \, d\boldsymbol{h} \tag{34}$$

$$p^\infty(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta) = \frac{p^\infty(\boldsymbol{v}^{\text{out}}, \boldsymbol{v}^{in};\Theta)}{p^\infty(\boldsymbol{v}^{in};\Theta)} \tag{35}$$

we have

$$\frac{\partial p^\infty(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta)}{\partial \Theta_{kl}} = \frac{\frac{\partial}{\partial \Theta_{kl}} p^\infty(\boldsymbol{v}^{\text{out}}, \boldsymbol{v}^{in};\Theta)}{p^\infty(\boldsymbol{v}^{in};\Theta)} - p^\infty(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta) \frac{\frac{\partial}{\partial \Theta_{kl}} p^\infty(\boldsymbol{v}^{in};\Theta)}{p^\infty(\boldsymbol{v}^{in};\Theta)}. \tag{36}$$

Using definition $p^\infty(\boldsymbol{s};\Theta) := p^\infty(\boldsymbol{v}^{\text{out}}, \boldsymbol{v}^{in}, \boldsymbol{h};\Theta) = (1/Z(\Theta)) \exp\left(-H(\boldsymbol{v}^{\text{out}}, \boldsymbol{v}^{in}, \boldsymbol{h};\Theta)\right)$, a short calculation gives

$$\frac{\frac{\partial}{\partial \Theta_{kl}} p^\infty(\boldsymbol{v}^{\text{out}}, \boldsymbol{v}^{in};\Theta)}{p^\infty(\boldsymbol{v}^{in};\Theta)} = -\frac{1}{Z(\Theta)} \frac{\partial Z(\Theta)}{\partial \Theta} \cdot p^\infty(\boldsymbol{v}^{\text{out}}|\boldsymbol{v}^{in};\Theta) - \int_{\mathbb{R}^r} \frac{\partial H(\boldsymbol{s};\Theta)}{\partial \Theta_{kl}} p^\infty(\boldsymbol{v}^{\text{out}}, \boldsymbol{h}|\boldsymbol{v}^{in};\Theta) \, d\boldsymbol{h} \tag{37}$$

$$\frac{\frac{\partial}{\partial \Theta_{kl}} p^\infty(\boldsymbol{v}^{in};\Theta)}{p^\infty(\boldsymbol{v}^{in};\Theta)} = -\frac{1}{Z(\Theta)} \frac{\partial Z(\Theta)}{\partial \Theta} - \int_{\mathbb{R}^r} \int_{\mathbb{R}^{N_{\boldsymbol{v}^{\text{out}}}}} \frac{\partial H(\boldsymbol{s};\Theta)}{\partial \Theta_{kl}} p^\infty(\boldsymbol{v}^{\text{out}}, \boldsymbol{h}|\boldsymbol{v}^{in};\Theta) \, d\boldsymbol{v}^{\text{out}} \, d\boldsymbol{h}. \tag{38}$$

Applying (36)–(38) in (32) yields the approximate derivative of CD_n^{IO} (24).

B. Backpropagation Using Discriminant Function Q_h^δ

For fine-tuning a GerDA DNN with back-propagation, the partial derivatives have to be derived from the discriminant criterion Q_h^δ w.r.t. all features $\boldsymbol{h}_i$, $1 \le i \le N$. Therefore, let be $\mathbf{H} := (\boldsymbol{h}_1, \ldots, \boldsymbol{h}_{N_1}, \boldsymbol{h}_{N_1+1}, \ldots, \boldsymbol{h}_{N-N_C+1}, \ldots, \boldsymbol{h}_N)$ a matrix of $N = N_1 + N_2 + \cdots + N_C$, $N_i \neq 0$, labeled features $\boldsymbol{h}_i \in \mathbb{R}^r$ sorted in ascending order with respect to their class-membership $\omega(\boldsymbol{h}_i) = \omega_k$, $1 \le k \le C$.

Then, we define the weighted discriminant criterion

$$J : \mathbb{R}^{r \times N} \to \mathbb{R}, \;\; J(\mathbf{H}) := trace\left\{ (\mathbf{S}_T^\delta)^{-1} \mathbf{S}_B^\delta \right\} \tag{39}$$

with the weighted total scatter matrix

$$\mathbf{S}_T^\delta := \mathbf{S}_W + \mathbf{S}_B^\delta \tag{40}$$

and $\mathbf{S}_W$, $\mathbf{S}_B^\delta$ as defined in Section II-C.

In the following, $\boldsymbol{A}(\cdot, j)$ denotes the j th column vector and $A(i, j)$ denotes the (i, j)th element of any $\boldsymbol{A} \in \mathbb{R}^{n \times m}$.

For $\boldsymbol{H}(., s)$ belonging to class ω_q, one gets

$$\frac{\partial S_W(u,v)}{\partial H(p,s)} = \frac{1}{N} \begin{cases} \big(H(v,s) - m_q(v)\big) & \text{if } u = p, v \neq p \\ \big(H(u,s) - m_q(u)\big) & \text{if } u \neq p, v = p \\ 2(H(p,s) - m_q(p)) & \text{if } u = p, v = p \\ 0 & \text{if } u \neq p, v \neq p \end{cases},$$

$$\frac{\partial S_B^\delta(u,v)}{\partial H(p,s)} = \frac{1}{N^2} \sum_{l=1, l\neq q}^{C} N_l \delta_{l,q} \cdot \begin{cases} \big(m_q(v) - m_l(v)\big) & \text{if } u = p, v \neq p \\ \big(m_q(u) - m_l(u)\big) & \text{if } u \neq p, v = p \\ 2 \cdot (m_q(p) - m_l(p)) & \text{if } u = p, v = p \\ 0 & \text{if } u \neq p, v \neq p \end{cases}.$$

From

$$\frac{\partial J(\mathbf{H})}{\partial H(p,s)} = \sum_{k,j=1}^{d} \frac{\partial (S_T^\delta)^{-1}(k,j) \cdot S_B^\delta(k,j)}{\partial H(p,s)} \tag{41}$$

$$= \sum_{k,j=1}^{d} \frac{\partial (S_T^\delta)^{-1}(k,j)}{\partial H(p,s)} S_B^\delta(k,j) \tag{42}$$

$$+ \sum_{k,j=1}^{d} (S_T^\delta)^{-1}(k,j) \frac{\partial S_B^\delta(k,j)}{\partial H(p,s)} \tag{43}$$

with

$$\frac{\partial (S_T^\delta)^{-1}(k,j)}{\partial H(p,s)} = -\sum_{v,u=1}^{d} (S_T^\delta)^{-1}(k,u) \frac{\partial S_T^\delta(u,v)}{\partial H(p,s)} (S_T^\delta)^{-1}(v,j) \tag{44}$$

and

$$\frac{\partial S_T^\delta(u,v)}{\partial H(p,s)} = \frac{\partial S_W(u,v)}{\partial H(p,s)} + \frac{\partial S_B^\delta(u,v)}{\partial H(p,s)} \tag{45}$$

it follows (assuming $\delta_{l,l} = 0$):

$$\frac{\partial J(\mathbf{H})}{\partial \mathbf{H}} = \frac{2}{N^2} \left[(\mathbf{S}_T^\delta)^{-1} \mathbf{S}_B^\delta - \mathbf{I} \right] \cdot (\mathbf{S}_T^\delta)^{-1} \cdot \left(\sum_{l=1}^{C} \mathbf{M}_l \right) - \frac{2}{N} (\mathbf{S}_T^\delta)^{-1} \cdot \mathbf{S}_B^\delta \cdot (\mathbf{S}_T^\delta)^{-1} \cdot (\mathbf{H} - \mathbf{M}) \quad \text{with}$$

$$\mathbf{M}_l := \left(N_l \cdot \delta_{l,q} \cdot (\boldsymbol{m}_l - \boldsymbol{m}_q) \cdot \mathbf{1}_{N_q}^T \right)_{1 \le q \le C} \in \mathbb{R}^{d \times N}$$

$$\mathbf{M} := \left(\boldsymbol{m}_q \cdot \mathbf{1}_{N_q}^T \right)_{1 \le q \le C} \in \mathbb{R}^{d \times N}. \tag{46}$$

Acknowledgment

The authors would like to thank H.-G. Meier for his contribution to some theoretical discussions and L. van der Maaten for supporting their experiments with his implementation of t-SNE.

References

[1] F. Nie, S. Xiang, and C. Zhang, "Neighborhood minmax projections," in *Proc. 20th Int. Joint Conf. Artif. Intell.*, 2007, pp. 1–6.

[2] M. Sugiyama, "Dimenionality reduction of multimodal labeled data by local Fisher discriminant analysis," *J. Mach. Learn. Res.*, vol. 8, pp. 1027–1061, May 2007.

[3] S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller, "Fisher discriminant analysis with kernels," in *Proc. IEEE Neural Netw. Signal Process. Workshop*, Aug. 1999, pp. 41–48.
[4] F. Nie, D. Xu, I. Tsang, and C. Zhang, "Flexible manifold embedding: A framework for semi-supervised and unsupervised dimension reduction," *IEEE Trans. Image Process.*, vol. 19, no. 7, pp. 1921–1931, Jul. 2010.
[5] D. Cai, X. He, and J. Han, "Efficient kernel discriminant analysis via spectral regression," in *Proc. 7th IEEE Int. Conf. Data Mining*, Oct. 2007, pp. 427–432.
[6] J. Ye and S. Ji, "Discriminant analysis for dimensionality reduction: An overview of recent developments," in *Biometrics: Theory, Methods, and Applications* (Computational Intelligence). New York: Wiley, 2009, pp. 1–19.
[7] C. Zhang, F. Nie, and S. Xiang, "A general kernelization framework for learning algorithms based on kernel PCA," *Neurocomputing*, vol. 73, nos. 4–6, pp. 959–967, Jan. 2010.
[8] R. Salakhutdinov and G. Hinton, "Learning a nonlinear embedding by preserving class neighbourhood structure," in *Proc. Int. Conf. Artif. Intell. Stat.*, vol. 11. 2007, pp. 1–8.
[9] R. A. Fisher, "The use of multiple measurements in taxonomic problems," *Ann. Eugen.*, vol. 7, no. 2, pp. 179–188, Sep. 1936.
[10] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *Science*, vol. 313, no. 5786, pp. 504–507, Jul. 2006.
[11] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in Neural Information Processing Systems*, vol. 19. Cambridge, MA: MIT Press, 2007, pp. 153–160.
[12] H. Larochelle, Y. Bengio, J. Loradour, and P. Lamblin, "Exploring strategies for training deep neural networks," *J. Mach. Learn. Res.*, vol. 1, pp. 1–40, Jan. 2009.
[13] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, and P. Vincent, "Why does unsupervised pre-training help deep learning?" *J. Mach. Learn. Res.*, vol. 11, pp. 625–660, Feb. 2010.
[14] A. Stuhlsatz, J. Lippel, and T. Zielke, "Discriminative feature extraction with deep neural networks," in *Proc. Int. Joint Conf. Neural Netw.*, Barcelona, Spain, Jul. 2010, pp. 1–8.
[15] A. Stuhlsatz, J. Lippel, and T. Zielke, "Feature extraction for simple classification," in *Proc. 20th Int. Conf. Pattern Recognit.*, Instanbul, Turkey, Aug. 2010, pp. 1525–1528.
[16] D. H. Ackley, G. E. Hinton, and T. J. Sejnowski, "A learning algorithm for Boltzmann machines," *Cognit. Sci.*, vol. 9, no. 1, pp. 147–169, Jan.–Mar. 1985.
[17] G. Hinton, S. Osindero, and Y. W. Teh, "A fast learning algorithm for deep belief nets," *Neural Comput.*, vol. 18, no. 7, pp. 1527–1554, Jul. 2006.
[18] M. Welling, M. Rosen-Zvi, and G. Hinton, "Exponential family harmoniums with an application to information retrieval," in *Advances in Neural Information Processing Systems*, vol. 17. Cambridge, MA: MIT Press, 2005, pp. 1481–1488.
[19] H. Osman and M. M. Fahmy, "On the discriminatory power of adaptive feed-forward layered networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 16, no. 8, pp. 837–842, Aug. 1994.
[20] S. Geman and D. Geman, "Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 6, no. 6, pp. 721–741, Nov. 1984.
[21] G. E. Hinton, "Training products of experts by minimizing contrastive divergence," *Neural Comput.*, vol. 14, no. 8, pp. 1771–1800, 2002.
[22] C. Wolf, D. Gaida, A. Stuhlsatz, S. McLoone, and M. Bongards, "Organic acid prediction in biogas plants using UV/vis spectroscopic online-measurements," in *Life System Modeling and Intelligent Computing* (Communications in Computer and Information Science), vol. 97, K. Li, X. Li, S. Ma, and G. W. Irwin, Eds. Berlin, Germany: Springer-Verlag, 2010, pp. 200–206.
[23] M. Sonka, V. Hlavac, and R. Boyle, *Image Processing, Analysis, and Machine Vision*. Boston, MA: PWS Publishing, 1998.
[24] Y. LeCun and C. Cortes. *The Mnist Database of Handwritten Digits*. New York Univ., New York [Online]. Available: http://yann.lecun.com/exdb/mnist
[25] L. van der Maaten, "Learning a parametric embedding by preserving local structure," in *Proc. 12th Int. Conf. Artif. Intell. Stat.*, vol. 5. 2009, pp. 384–391.
[26] M. Sips, B. Neubert, J. P. Lewis, and P. Hanrahan, "Selecting good views of high-dimensional data using class consistency," *Comput. Graph. Forum*, vol. 28, no. 3, pp. 831–838, 2009.
[27] P. Li, "Robust logitboost and adaptive base class (ABC) logitboost," in *Proc. 26th Conf. Uncertain. Artif. Intell.*, 2010, pp. 1–10.
[28] M. Islam, A. Sattar, F. Amin, X. Yao, and K. Murase, "A new adaptive merging and growing algorithm for designing artificial neural networks," *IEEE Trans. Syst., Man, Cybern., Part B: Cybern.*, vol. 39, no. 3, pp. 705–722, Jun. 2009.
[29] A. Stuhlsatz, C. Meyer, F. Eyben, T. ZieIke, G. Meier, and B. Schuller, "Deep neural networks for acoustic emotion recognition: Raising the benchmarks," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process.*, May 2011, pp. 5688–5691.
[30] J. Cook, V. Chandran, and S. Sridharan, "Multiscale representation for 3-D face recognition," *IEEE Trans. Inf. Forens. Security*, vol. 2, no. 3, pp. 529–536, Sep. 2007.
[31] A.-A. Bhuiyan and C. Liu, "On face recognition using Gabor filters," *World Acad. Sci., Eng. Technol.*, vol. 28, no. 4, pp. 51–56, Apr. 2007.
[32] P. Kovesi. *MATLAB and Octave Functions for Computer Vision and Image Processing*. School Computer Science Software Eng., Univ. Western Australia, Perth, Australia [Online]. Available: http://www.csse.uwa.edu.au/∼pk/research/matlabfns/
[33] K. Jarrett, K. Kavukcuoglu, M. Ranzato, and Y. LeCun, "What is the best multi-stage architecture for object recognition?" in *Proc. IEEE Int. Conf. Comput. Vis.*, Sep.–Oct. 2009, pp. 2146–2153.
[34] M. Ranzato, C. Poultney, S. Chopra, and Y. LeCun, "Efficient learning of sparse representations with an energy-basedmodel," in *Advances in Neural Information Processing Systems*. Cambridge, MA: MIT Press, 2006.
[35] K. Labusch, E. Barth, and T. Martinetz, "Simple method for high-performance digit recognition based on sparse coding," *IEEE Trans. Neural Netw.*, vol. 19, no. 11, pp. 1985–1989, Nov. 2008.
[36] A. Kolb, E. Barth, and R. Koch, "Tof-sensors: New dimensions for realism and interactivity," in *Proc. Conf. Comput. Vis. Pattern Recog., Workshop ToF Camera Based CV*, 2008, pp. 1–6.
[37] M. Böhme, M. Haker, K. Riemer, T. Martinetz, and E. Barth, "Face detection using a time-of-flight camera," in *Dyn3-D* (Lecture Notes in Computer Science), vol. 5742, A. Kolb and R. Koch, Eds. New York: Springer-Verlag, 2009, pp. 167–176.
[38] S. C. Brubaker, J. Wu, J. Sun, M. D. Mullin, and J. M. Rehg, "On the design of cascades of boosted ensembles for face detection," *Int. J. Comput. Vis.*, vol. 77, nos. 1–3, pp. 65–86, 2008.
[39] K. Sung and T. Poggio, "Example-based learning for view-based human face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 1, pp. 39–51, Jan. 1998.
[40] C. Garcia and M. Delakis, "Convolutional face finder: A neural architecture for fast and robust face detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 11, pp. 1408–1423, Nov. 2004.
[41] H. Kashima, T. Ide, T. Kato, and M. Sugiyama, "Recent advances and trends in large-scale kernel methods," *IEICE Trans. Inf. Syst.*, vol. E92-D, no. 7, pp. 1338–1353, 2009.
[42] W. Zheng, Z. Lin, and X. Tang, "A rank-one update algorithm for fast solving kernel Foley–Sammon optimal discriminant vectors," *IEEE Trans. Neural Netw.*, vol. 21, no. 3, pp. 393–403, Mar. 2010.
[43] P. K. Mallapragada, R. Jin, A. K. Jain, and Y. Liu, "SemiBoost: Boosting for semi-supervised learning," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 11, pp. 2000–2014, Nov. 2009.
[44] Y. Zhang and D.-Y. Yeung, "Semisupervised generalized discriminant analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 8, pp. 1207–1217, Aug. 2011.
[45] C. S. Dhir and S.-Y. Lee, "Discriminant independent component analysis," *IEEE Trans. Neural Netw.*, vol. 22, no. 6, pp. 845–857, Jun. 2011.

André Stuhlsatz received the Ph.D. degree in electrical engineering from Otto-von-Guericke University, Magdeburg, Germany, in 2010.

Since 2011, he has been a Development Manager for instrumentation with SMS Siemag AG, Düsseldorf. He was with the Department of Mechanical and Process Engineering, and the Laboratory for Pattern Recognition, Department of Electrical Engineering, Düsseldorf University of Applied Sciences, Düsseldorf, Germany, from 2004 to 2010. His current research interests include machine learning, statistical pattern recognition, neural networks, computer vision, and industrial applications.

Dr. Stuhlsatz was a member of the Cognitive Systems Group, Otto-von-Guericke-University. He was the winner of the Best Paper Award from the International Conference on Machine Learning and Applications in 2008.

Jens Lippel received the B.Eng. degree in product development and manufacturing from the Department of Mechanical and Process Engineering, Düsseldorf University of Applied Sciences, Düsseldorf, Germany. He is currently pursuing the M.Sc. degree in simulation and experimental techniques with the same university, where he is working as a Graduate Teaching Assistant.

His current research interests include machine learning, computer vision, and face and object recognition.

Thomas Zielke received the Dipl.-Ing. degree in computer engineering from the Technical University of Darmstadt, Darmstadt, Germany, the M.Sc. degree in knowledge-based systems from Heriot-Watt University, Edinburgh, U.K., and the Ph.D. (Dr.-Ing) degree in neurocomputing from Ruhr-University Bochum, Bochum, Germany.

He was the President and CEO of C-VIS Computer Vision und Automation GmbH. From 2006 to 2009, he served as CTO, Facial Recognition, Cross Match Technologies, Inc., Palm Beach Gardens, FL. He is currently a Professor of computer science with the Düsseldorf University of Applied Sciences, Düsseldorf, Germany. His current research interests include pattern recognition, neural networks, computer vision, face recognition, and other biometric applications.

Dr. Zielke was awarded the European IST Prize 1996 for pioneering face recognition applications.