Online learning and generalization of parts-based image representations by Non-Negative Sparse Autoencoders

Andre Lemme[a,*], René Felix Reinhart[a], Jochen Jakob Steil[a]

[a]*Research Institute for Cognition and Robotics (CoR-Lab), Bielefeld University, Universitätsstr. 25, 33615 Bielefeld*

Abstract

We present an efficient online learning scheme for non-negative sparse coding in autoencoder neural networks. It comprises a novel synaptic decay rule that ensures non-negative weights and an intrinsic self-adaptation rule that optimizes sparseness of the non-negative encoding. We show that non-negativity constrains the space of solutions such that overfitting is prevented and very similar encodings are found irrespective of the network initialization and size. We benchmark the novel method on real-world datasets of handwritten digits and faces. The autoencoder yields higher sparseness and lower reconstruction errors than related batch algorithms based on matrix factorization. It generalizes to new inputs both accurately and without costly computations, which is fundamentally different from the classical matrix factorization approaches.

Keywords: autoencoder, non-negativity, sparse coding

1. Introduction

Unsupervised learning techniques that can find filters for relevant parts in images are particularly important for visual object classification [1]. The goal is to learn latent causes such that simple addition of these causes can explain a set of images. There is strong evidence that such parts-based representations exist in the mammalian brain and strongly contribute to object recognition [2]. In addition, parts-based representations are easy to interpret for humans which is an important advantage in several application areas [3]. We model the learning of parts-based representations in a novel autoencoder neural network. Technically speaking, *non-negativity* of the neural encodings is required, i.e. network parameters and states are constrained to be positive only. Beside non-negativity, neural physiological evidence as well as energy constraints [4] motivate to encode sensory inputs with only a few neurons active at a specific

[*]Corresponding author

Email addresses: alemme@CoR-Lab.Uni-Bielefeld.de (Andre Lemme), freinhar@CoR-Lab.Uni-Bielefeld.de (René Felix Reinhart), jsteil@CoR-Lab.Uni-Bielefeld.de (Jochen Jakob Steil) point in time. This strategy of encoding is commonly referred to as *sparse coding*. Describing patterns with less active neurons minimizes the probability of destructive cross talk [4] and enhances the probability of linear separability, a crucial feature for object classification [5]. To address these requirements, several matrix factorization approaches have been developed that decompose a set of patterns into parts and their configuration which both are non-negative and sparse. We discuss these approaches in Sec. 2. Unfortunately, matrix factorization turns out to have serious drawbacks: On the one hand, generalization to new patterns is either computationally expensive or does not respect non-negativity. On the other hand, reconstruction accuracy is naturally impaired by the additional non-negativity constraints in contrast to unconstrained methods that allow negative components. The interest in factorization methods therefore has somewhat declined in recent years though still successful applications are frequently reported.

Since the groundbreaking introduction of layer-wise pretraining to deep architectures [6, 7], stacked autoencoder networks have proven to be excellent models for pattern recognition (see [8] for a thorough discussion). In particular, autoencoders with tied weights, i.e. utilizing the same weights for pattern encoding and decoding, can be trained efficiently by contrastive divergence or simple error-correction learning rules without backpropagation of errors. Moreover, autoencoders with tied weights provide a generic model for efficient encoding and decoding of new inputs by simply propagating activities through the network. These feed-forward networks achieve very good generalization performance but have not yet been combined with non-negativity and sparseness constraints.

We combine non-negativity and sparseness in a novel autoencoder model that was first introduced in [9]. Non-negativity of input connection weights is enforced by a novel asymmetric weight decay function that punishes negative weights more than positive ones. Logistic activation functions in the hidden layer map neural activities to strictly positive outputs. Sparseness of the encodings is achieved by an unsupervised self-adaptation rule, which adapts the non-linear activation functions based on the principle of intrinsic plasticity [10]. Both synaptic and intrinsic online learning rules work in parallel, use only local information and are very efficient to compute. The autoencoder enables fast encoding of new inputs by simple forward propagation of activities without costly and iterative constraint satisfaction.

We first show the basic features of the non-negative autoencoder on a synthetic dataset. In a next step, we compare the reconstruction performance as well as the generated encodings and basis images of the introduced model to the non-negative offline algorithms NMF and NMFSC on a benchmark dataset of handwritten digits. The efficient online implementation outperforms the offline algorithms with respect to reconstruction errors and sparseness of the basis images as well as the encodings, which underlines the excellent generalization abilities of the autoencoder. Finally, we show that the autoencoder robustly identifies distinct parts-based representations for real-world datasets of handwritten digits and faces irrespective of the network size and initialization.

2. Related work on learning parts-based representations

Lee and Seung proposed a non-negative matrix factorization (NMF) in [3] to produce non-negative encodings of input images by combining a linear matrix factorization approach with non-negativity constraints. In [11], Hoyer introduced the non-negative sparse coding (NNSC) approach which learns non-negative and sparse representation. Later, Hoyer also added a sparseness constraint to the non-negative matrix factorization [12]. The difference between NMF with sparseness constraint (NMFSC) and NNSC is that NMFSC provides two parameters that control the degree of sparseness of the basis images and the encodings, respectively. The batch algorithms NMF and NMFSC are based on the matrix factorization equation $\mathbf{X} \approx \mathbf{WH}$, where $\mathbf{X}$ is a non-negative data matrix. The factorization process starts from the initial basis[1] $\mathbf{W}$ and code matrix $\mathbf{H}$. Then, iterative updates of the encodings and basis images are conducted in order to reduce the reconstruction error while complying with non-negativity and sparseness constraints for both the basis $\mathbf{W}$ and the encodings $\mathbf{H}$. The main drawback of the matrix factorization approach is that generalization to new input patterns is not easily possible: The encoding of novel inputs with respect to the previously trained basis $\mathbf{W}$ requires either to compute the pseudo-inverse of the basis $\mathbf{W}^+$ [13] which, however, does not preserve the non-negativity constraint of the encodings [14], or to conduct the constrained optimization of the encodings $\mathbf{H}$ for the new inputs with fixed basis $\mathbf{W}$ [12]. The latter approach does preserve non-negativity but introduces considerable computational costs and does not guarantee the unique mapping from inputs to encodings due to the iterative process which depends on initial conditions. Therefore, NMF is not suited for problems that require efficient real-time processing of inputs with encodings based on the same basis and encodings that fulfill the non-negativity and sparseness constraints. Despite these massive drawbacks of the NMF approach, NMF and NNSC are successfully applied in hierarchical models for object recognition [15, 16] and yield superior results on various datasets and conditions for face recognition compared to other state-of-the-art feature extraction methods [13]. We conclude that non-negativity is beneficial for pattern recognition, but the commonly applied matrix factorization methodology to achieve non-negativity suffers from serious shortcomings, in particular the inefficient and iterative encoding of novel inputs.

The problem of non-negativity and computational efficiency is addressed by the non-negative and online version of the principle component analysis introduced in [17]. However, non-negative PCA does not produce a sparse encoding of the inputs, which is generally a drawback for later object classification layers.

In the following, we introduce a novel autoencoder that combines sparseness and non-negativity with efficient encoding of new stimuli.

[1]Though the term basis is slightly abusive because the basis images are not linearly independent and can be over-complete, we stick to this common terminology.

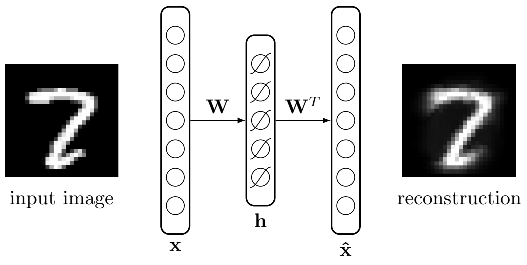


Fig. 1: Autoencoder network with input image $\mathbf{x}$ (left) and reconstructed image $\hat{\mathbf{x}}$ (right). $\mathbf{W} \equiv \mathbf{W}^T$ is the tied weight matrix and $\mathbf{h}$ the hidden network state.

3. Non-negative sparse autoencoder (NNSAE)

We modify an autoencoder network in order to obtain non-negative and sparse encodings with only positive network weights from non-negative input data. The network architecture is shown in Fig. 1. We denote the input by $\mathbf{x} \in \mathbb{R}^D$ and the network reconstruction by $\hat{\mathbf{x}} = \mathbf{W}^T\mathbf{f}(\mathbf{W}\mathbf{x})$, where $\mathbf{W} \in \mathbb{R}^{N\times D}$ is the weight matrix and $\mathbf{f}(\cdot)$ are parameterized activation functions

$$f_i(g_i, a_i, b_i) = \frac{1}{1 + e^{(-a_i g_i - b_i)}} \quad \in [0, 1] \tag{1}$$

with slopes a_i and biases b_i that are applied component-wise to the neural activities $\mathbf{g} = \mathbf{W}\mathbf{x}$ ($\mathbf{g} \in \mathbb{R}^N$). Non-negative encodings $\mathbf{h}$ are already assured by the logistic activation functions $f_i(g_i, a_i, b_i)$.

3.1. Learning to reconstruct

Learning of the autoencoder is based on minimizing the reconstruction error

$$E = \frac{1}{K}\sum_{i=1}^{K} ||\mathbf{x}_i - \hat{\mathbf{x}}_i||^2 \tag{2}$$

on the training set. Note that $\mathbf{W} \equiv \mathbf{W}^T$ is the same matrix for encoding and decoding, i.e. the autoencoder has *tied* weights, which reduces the number of parameters to be adapted and is in general motivated by the idea that a neuron should contribute to the reconstruction only the parts it detects in the input. Tied weights result in a dual role of the network weights: On the one hand, each row of the weight matrix $\mathbf{w}_i$ serves as filter when encoding the input (left part in Fig. 1). On the other hand, each weight vector contributes as basis image to the reconstruction in the decoding step (right part in Fig. 1). We refer to rows of the weight matrix $\mathbf{W}$ as *basis images* in the remaining text in order to facilitate the discussion. Moreover, using tied weights means that learning does not require backpropagation of errors through the hidden layer. With tied

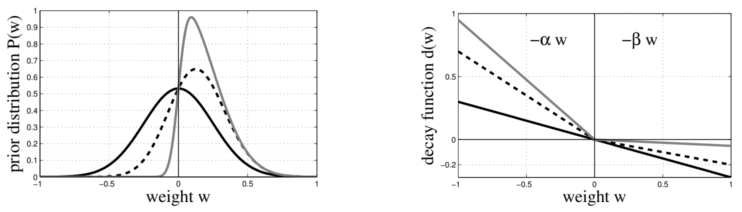


Fig. 2: Skewed prior probabilities for a non-negative weight distribution (left). The corresponding decay functions (right) with two parameters α and β. These parameters control the weight decay depending on the sign of w.

weights, learning reduces to the following online error correction rule applied after presentation of example $\mathbf{x}$:

$$\Delta w_{ij} = \eta\,(x_i - \hat{x}_i)\,h_j + d(\tilde{w}_{ij}), \tag{3}$$

where w_{ij} is the connection from hidden neuron j to the output neuron i and h_j is the activation of neuron j. We use an adaptive learning rate $\eta = \tilde{\eta}\,(||\mathbf{h}||^2 + \epsilon)^{-1}$ that scales the gradient step size $\tilde{\eta}$ by the inverse network activity and is similar to backpropagation-decorrelation learning in reservoir networks [18]. The decay function $d(\tilde{w}_{ij})$ will be discussed in the next section.

3.2. Asymmetric decay enforces non-negative network parameters

In order to enforce non-negative weights, we introduce a novel asymmetric, piecewise linear decay function:

$$d(\tilde{w}_{ij}) = \begin{cases} -\alpha\,\tilde{w}_{ij} & \text{if } \tilde{w}_{ij} < 0 \\ -\beta\,\tilde{w}_{ij} & \text{else,} \end{cases} \tag{4}$$

where $\tilde{w}_{ij} = w_{ij} + \Delta w_{ij}$ is the new computed weight after error correction according to (Eq. 3). Typically, a quadratic cost term $\lambda||\mathbf{W}||^2$ is added to the error function (Eq. 2) which serves as regularization and is equivalent to a Gaussian prior distribution of the network weights. Using gradient descent, the Gaussian prior results in a so-called decay term $d(w_{ij}) = -\lambda w_{ij}$. We assume a virtually deformed Gaussian prior that is skewed with respect to the sign of the weight. Gradient descent then yields the asymmetric decay function (Eq. 4). The principal relation between skewed prior distribution for the weights and the asymmetric decay function is illustrated in Fig. 2. The main advantage of (Eq. 4) is the parameterized decay behavior depending on the sign of the weight: It is possible to allow a certain degree of negative weights ($0 \leq \alpha \ll 1$) or to prohibit negative weights completely ($\alpha = 1$). The decay function (Eq. 4) reduces to the standard case of a symmetric prior for $\alpha = \beta$. Setting $\alpha = 1$ and applying the decay on the updated weights $\tilde{w}_{ij}$ ensures that the weight matrix $\mathbf{W}$ has strictly non-negative values.

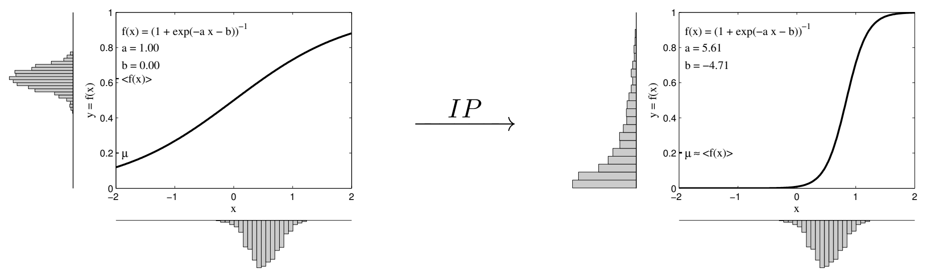


Fig. 3: IP learning maximizes a neuron's information transmission properties by adapting the slope a and bias b of the activation function: Initial activation function with input and output distributions (left). Optimized activation function after IP training with approximately exponential output distribution (right).

3.3. Adaptation of non-linearities achieves sparse encodings

To enforce sparseness of the encodings, we apply the intrinsic plasticity (IP) mechanism introduced by Triesch in [10]. IP is a biologically inspired learning rule and produces energy-efficient, sparse encodings. The idea is to adjust the parameters a_i, b_i of the logistic activation function (Eq. 1) in order to optimize information transmission of the neurons. More formally, the Kullback-Leibler divergence of the neuron's output distribution with respect to a desired, exponential output distribution is minimized. The mean activity level μ of the desired output distribution appears as global parameter in the learning process. We yield the following online gradient rule with learning rate η_{IP}:

$$\Delta b_i = \eta_{IP} \left(1 - \left(2 + \frac{1}{\mu} \right) h_i + \frac{1}{\mu} h_i^2 \right), \tag{5}$$

$$\Delta a_i = \eta_{IP} \frac{1}{a_i} + g_i \, \Delta b_i. \tag{6}$$

This unsupervised self-adaptation rule is local in time and space and therefore efficient to compute. Note that IP optimizes the lifetime sparseness of a neuron (sparse activation of one neuron over time), which is different from the typical approach to increase the population sparseness. But by coupling synaptic and intrinsic plasticity, the population sparseness is effectively optimized if diverse basis images are learned. We use a small learning rate $\eta_{IP} = 0.0001$ if not otherwise stated and set the desired mean activity $\mu = 0.2$ throughout the experiments. We initialize the parameters of the activation functions $\mathbf{a} = \mathbf{1}$ and $\mathbf{b} = -\mathbf{3}$ in order to accelerate convergence.

In the context of the non-negative autoencoder, IP is not only important for increased sparseness of the encodings. The strictly non-negative input distribution caused by the non-negative weight and data matrices is particularly challenging and requires special attention for the activation function selection. For instance, an unbiased logistic function degrades to half of its output range for non-negative inputs (compare Fig. 3 (left)). IP adapts the activation functions to these non-negative input distributions and ensures optimal information transmission (compare Fig. 3 (right)).

4. Identifying a parts-based representation

We first use a synthetic dataset which is constructed out of known parts and show that the non-negative sparse autoencoder (NNSAE) identifies the underlying image components.

4.1. The BARS dataset and network training

The non-linear BARS problem was introduced by Földiák [19]. The dataset contains 9×9 pixel images with randomly positioned horizontal and vertical bars and comprises 10.000 trainings and 50.000 test images. We scale the pixel intensities into the interval $[0.00, 0.25]$. There are at least two bars present per image, but maximally two horizontal and two vertical bars. Hence, 18 basis images each containing one bar are needed to perfectly describe the data. The unsupervised learning problem is to compute a basis comprising all independent components, i.e. bars. Some examples of the dataset are shown in Fig. 4(a).

The networks have $D = 9 \times 9 = 81$ input neurons. We select different sizes N for the hidden layer in the experiments. The weight matrix $\mathbf{W}$ is initialized randomly according to a uniform distribution in $[0.00, 0.05]$. We set the learning rate in (Eq. 3) to $\tilde{\eta} = 0.01$. Each autoencoder is trained for 100 epochs, where the entire training set is presented to the model in each epoch. We train strictly non-negative autoencoders in this section using $\alpha = 1$ and $\beta = 0$.

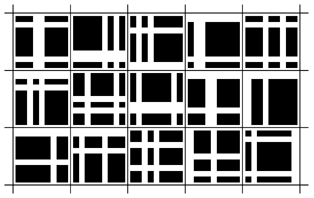
4.2. Parts-based representation of the BARS dataset

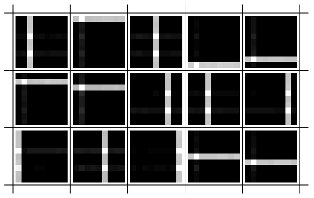
We systematically change the size of the hidden layer to investigate whether all basis images are found by the model and how a too large hidden layer with $N > 18$ affects the learned representation.

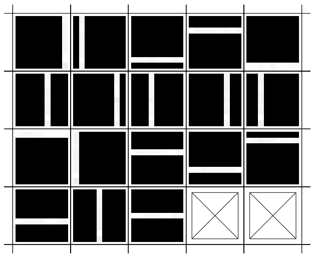
In the first experiment, we use less than the required 18 neurons in the hidden layer. The resulting basis images are shown in Fig. 4(b) for a network with $N = 15$ neurons. The model finds basis images corresponding to the bars the data is constructed from, but the basis images contain more than one bar (see Fig. 4(b)). The network is forced to find a suboptimal solution with multiple bars per neuron in order to minimize the reconstruction error.

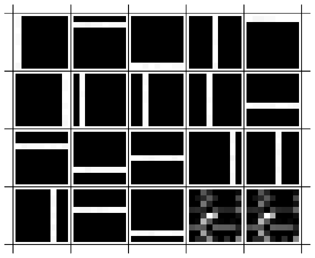
In the next experiment, we use 18 neurons in the hidden layer which matches the number of basis images used for data construction. The basis images learned by the model exactly represent the 18 bars underlying the data (see Fig. 4(c)). Though the dataset is rather simple, this result is remarkable for an online learning algorithm. Does the NNSAE also identify these 18 basis images when equipped with more than 18 hidden neurons?

In the next experiments, we equip the hidden layer with 20 and 50 neurons. The resulting basis images are shown in Fig. 4(d) and Fig. 4(e). All 18 basis images are found by the model irrespective of the hidden layer size. The remaining basis vectors show a rather random structure (see Fig. 4(d) and Fig. 4(e)). We observed that these randomly structured weight vectors $\mathbf{w}_i$ have only small entries which are only pronounced by the normalized display and do not contribute to the reconstruction, i.e. are unused. We can identify "used" and "unused" basis images by using a simple threshold criterion: If $\max(\mathbf{w}) < 0.001$, the basis


(a) Example images with bars.


(b) $N = 15$


(c) $N = 18$


(d) $N = 20$

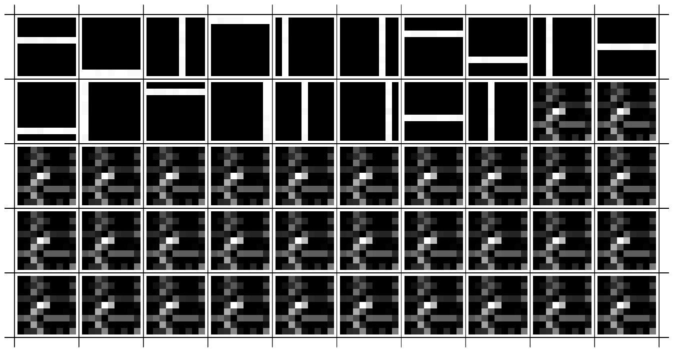

(e) $N = 50$

Fig. 4: Some examples of the BARS dataset (a). Learned basis images for different hidden layer sizes: network with $N = 15$ (b), $N = 18$ (c), $N = 20$ (d) and $N = 50$ (e) neurons in the hidden layer. In this toy example, 18 neurons are enough to represent each bar in the image dataset. The NNSAE finds all basis images when equipped with enough neurons, where excessive neurons form unused basis images with $max(\mathbf{w}) < 0.001$. Note that these basis images are displayed in a normalized manner to show their structure.

image is judged as unused. Applying this thresholding to the hidden layer, we obtain the order of basis images as shown in Fig. 4(d) and Fig. 4(e).

Interestingly, the unused neurons have very similar structured basis images. This is a result of the constrained learning dynamics: The addition of basis images to form the reconstruction $\hat{\mathbf{x}} = \sum_{i=1}^{N} \mathbf{w}_i h_i$ with $\mathbf{w}_i > 0$ and $h_i > 0$ introduces a competition between the neurons for contributions to the reconstruction. Initially, some basis images match inputs better than others and win the competition for contribution to the reconstruction and specialize to these inputs. IP activates not yet specialized neurons from time to time in order to fulfill the lifetime sparseness constraint. However, if there is no useful contribution to add, the error $(\mathbf{x} - \hat{\mathbf{x}}) < 0$ becomes negative and the weight decreases. As a consequence, if too many neurons are present in the network, the weights of some neurons decay: The unused basis images converge to the same structure with all entries close to zero because of the same error and similar activities. These competitive dynamics of the learning process self-regularize the network resources and can be utilized to estimate the intrinsic, parts-based dimensionality of the data which we further discuss in Sec. 6.

We conclude that the NNSAE model is able to find all basis images in the data if we equip the hidden layer with enough neurons. In contrast, the unconstrained autoencoder ($\alpha = \beta = 0$) and PCA produce dense encodings regardless of the basis size. The reconstruction improves with increasing network size, but the basis images will not reflect the parts-based data structure. The NNSAE identifies the parts that create the data irrespective of the hidden layer's size.

5. Implications of the non-negativity constraint

We analyze the effect of non-negativity in the non-negative sparse autoencoder and compare the non-negative bases across the NNSAE and batch algorithms NMF and NMFSC on a real-world dataset of handwritten digits.

5.1. Encoding and decoding of handwritten digits

The MNIST dataset is commonly used to benchmark image reconstruction and classification methods. For the following experiments the "MNIST-basic" set[2] is used, which comprises 10.000 trainings and 50.000 test images of handwritten digits from 0 to 9 in a centered and normalized 28×28 pixel format. Some example digits from the test set are shown in the top row of Fig. 5.

We use autoencoders with 784 input neurons and 100 hidden neurons to encode and decode the input images. Learning rates, initialization ranges and number of training epochs are applied as before. To account for the random network initialization, we present results averaged over 10 trials. The following variates are calculated to quantify the model properties:

[2] `http://www.iro.umontreal.ca/~lisa/twiki/bin/view.cgi/Public/DeepVsShallowComparisonICML2007`

	$\alpha=0$	$\alpha=0.1$	$\alpha=0.2$	$\alpha=0.3$	$\alpha=1.0$
$PMSE$	0.0107	0.0149	0.0149	0.0150	0.0151
$STD \cdot 10^{-4}$	0.5966	0.4079	0.7153	0.7143	0.6893
$s(\mathbf{H})$	0.3954	0.3465	0.3474	0.3460	0.3512
$s(\mathbf{W})$	0.5334	0.9379	0.9363	0.9397	0.9322

Tab. 1: $PMSE$ with standard deviation on the test set and sparseness of the encodings $\mathbf{H}$ and network weights $\mathbf{W}$ depending on asymmetric decay rate α. Note that only for $\alpha=1$ a fully non-negative weight matrix will be enforced.

(a) Pixel-wise mean square error

$$PMSE = \frac{1}{KD} \sum_{i=1}^{K} \sum_{j=1}^{D} (x_j^i - \hat{x}_j^i)^2 \tag{7}$$

of the reconstructions, where D is the dimension of the data and K the number of samples. (b) Average sparseness of the code matrix $\mathbf{H}$ and basis images $\mathbf{W}$, where we estimate the sparseness of a vector $\mathbf{v} \in \mathbb{R}^n$ by

$$s(\mathbf{v}) = (\sqrt{n} - (\Sigma_{i=1}^n |v_i|)/\sqrt{\Sigma_{i=1}^n v_i^2}))(\sqrt{n} - 1)^{-1}. \tag{8}$$

This function was introduced by Hoyer [12] and rates the energy of a vector $\mathbf{v}$ with values between zero and one, where sparse vectors $\mathbf{v}$ are mapped to $s(\mathbf{v}) \gg 0$.

5.2. Impact of the asymmetric decay

We first investigate the impact of the asymmetric decay function (Eq. 4) by increasing α stepwise. Tab. 1 shows the $PMSE$ and the average sparseness of hidden states as well as connection weights depending on α while $\beta \equiv 0$. The results in Tab. 1 for the case without decay ($\alpha=0$) show that a certain amount of negative weights seems to be useful in terms of the reconstruction error. The non-negativity constraint introduces additional costs to the optimization process which consequently increases the reconstruction error slightly. However, if we commit to non-negativity, it does not matter whether there is just a small set of negative entries ($\alpha = 0.1$) or only positive entries ($\alpha = 1$) in the weight matrix. This indicates that a rather moderate non-negativity constraint causes the weight dynamics and the final solution to change completely, i.e. non-negativity narrows the search space for learning critically. We therefore set α either to zero or to unity in the following experiments.

5.3. Online NNSAE versus offline NMF and NMFSC

We compare the offline algorithms NMF and NMFSC with the online NNSAE regarding the reconstruction ability, the sparseness of the encodings and the generated basis images. We show that the sparse autoencoder is superior to the offline methods in generalizing to novel inputs and with respect to the sparseness of the encodings despite the purely local and online learning rules. Finally,

	$PMSE$		Sparseness	
	Test	Train	$sp(\mathbf{H})$	$sp(\mathbf{W})$
NMF	$0.047 \pm 1 \cdot 10^{-4}$	$0.011 \pm 3 \cdot 10^{-4}$	0.26	0.89
NMFSC	$0.109 \pm 6 \cdot 10^{-6}$	$0.014 \pm 2 \cdot 10^{-4}$	0.29	0.90
NNSAE	$0.015 \pm 6 \cdot 10^{-5}$	$0.012 \pm 7 \cdot 10^{-5}$	0.35	0.93

Tab. 2: Pixel-wise mean square reconstruction errors for training and test set as well as sparseness of code and weight matrices for NMF, NMFSC and NNSAE.

we compare the learned bases across the methods and show that non-negativity enforces a specific coding scheme.

To make the comparison between online and offline learning fair, we iterate the factorization algorithms 100 times such that the batch algorithms see the training data 100 times like the NNSAE. We further set the sparseness parameters provided by NMFSC for the weight and code matrix to $s(\mathbf{W}) = 0.9$ and $s(\mathbf{H}) = 0.35$ according to the values obtained for the NNSAE. To evaluate the generalization of NMF and NMFSC on the test set, we keep the trained basis images $\mathbf{W}$ fixed and update only the code matrix $\mathbf{H}$ iteratively as in the training. The NNSAE processes new data by propagating activity from the input to the hidden layer, i.e. calculating $\mathbf{h} = f(\mathbf{W}\mathbf{x})$, without further iterations to satisfy non-negativity and sparseness constraints.

5.3.1. Comparison of reconstruction errors and sparseness

Tab. 2 gives a review of the performance of NMF/NMFSC and compares it to the NNSAE. The NNSAE outperforms NMF/NMFSC with respect to the $PMSE$ on the test set, sparseness of the code matrix $\mathbf{H}$ and sparseness of the weight matrix $\mathbf{W}$. It is surprising that even NMFSC does not reach the sparse encodings that we achieve with the NNSAE, although the constraint is set to be equal ($s(\mathbf{H}) = 0.35$). We show for some examples of the test set the corresponding code histograms and reconstructions by NMFSC and NNSAE in Fig. 5. The sparseness of the basis images $\mathbf{w}_i$ does not differ significantly for the different methods, which seems to be the effect of the non-negative constraint itself. We show in the next section that the basis images are indeed very similar. Consequently, the performance on the training set is comparable between all tested methods. However, NNSAE clearly outperforms the matrix factorization methods on the test set (compare Tab. 2). The NNSAE utilizes the same weight matrix for encoding and decoding and provides a model to encode new data, whereas NMF and NMFSC need to process new data iteratively to preserve the non-negative constraint and can not draw on an underlying model. Also the non-linearity is part of the NNSAE model, whereas in NMF non-linearity of the encodings is generated by the iterative matrix updates, i.e. is an algorithmic non-linearity.

5.3.2. Comparison of basis images

The quantitative comparison of code sparseness and basis images indicates that the encodings of the non-negative methods are very similar. In Fig. 5 we

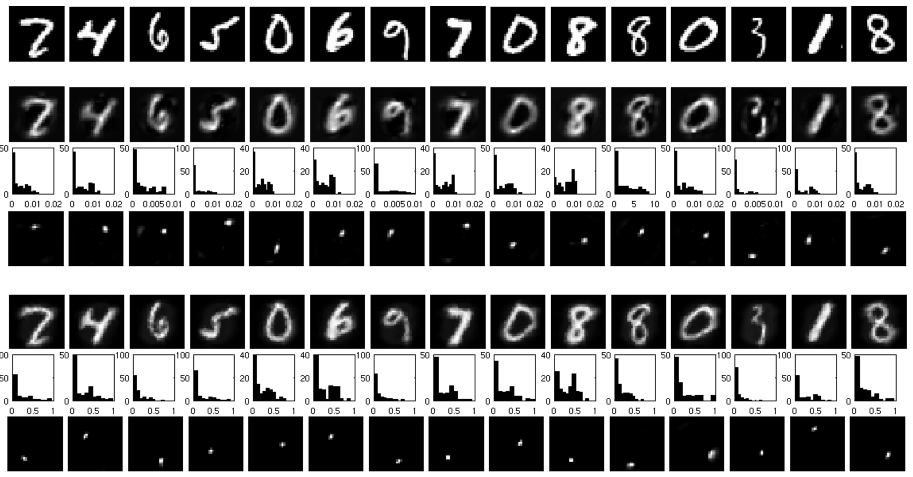


Fig. 5: Some example images of the MNIST dataset from the test set (top row). Reconstructions (2nd row), histograms of the encodings (3rd row) and some basis images (4th row) of the NMFSC. Reconstructions (5th row), histograms of the encodings (6th row) and some basis images (7th row) of the NNSAE.

show in the 4th and the last row some basis images of NMFSC and NNSAE. The basis images of both methods cover only blob-like areas, i.e. are local feature detectors, which corresponds to a sparse representation.

By applying non-negativity and sparse coding constraints, we force the model to find basis images covering a large area without degrading reconstructions by polluting details and sharp edges. However, the comparison on this level is limited to a behavioristic level. We face the key question whether NNSAE generates similar basis images $\mathbf{w}_i$ as the batch algorithms NMF and NMFSC. To compare the weight matrices of each method quantitatively, we use the Matlab[3] implementation of the Hungarian algorithm [20]. The Hungarian algorithm solves a weighted bipartite graph matching problem by calculating the minimal cost for matching the basis images of method A with method B. The comparison is based on the angle between two basis vectors. A cost matrix $\mathbf{C}_{N\times N}$ stores the cosine of the angle between the basis images of two weight matrices $\mathbf{W}^A = (\mathbf{w}_1^A, ..., \mathbf{w}_N^A)$ and $\mathbf{W}^B = (\mathbf{w}_1^B, ..., \mathbf{w}_N^B)$. Therefore, the cost matrix is given by

$$\mathbf{C}_{ij} = 1 - \frac{\mathbf{w}_i^A \cdot \mathbf{w}_j^B}{||\mathbf{w}_i^A||\;||\mathbf{w}_j^B||} \quad \forall\, i, j = 1, \dots, N,$$

where a smaller angle between the vectors means that less costs are stored in the matrix. The cost matrix $\mathbf{C}$ is used by the graph matching algorithm to find the best possible match between both sets of basis images.

[3]http://www.mathworks.de/

	NMFSC	NMF	NNSAE	AE
NMFSC	25.5790	-	-	-
NMF	29.4371	26.8124	-	-
NNSAE	29.6129	30.9014	25.3334	-
AE	63.7129	61.3591	64.7979	64.9006

Tab. 3: Minimal costs for matching the basis images of two methods averaged over 5 initializations. The non-negative methods find very similar basis images compared to the standard autoencoder (AE) with $\alpha = \beta = 0$.

Tab. 3 shows the mean costs of matching trained basis images of NMF, NMFSC and NNSAE averaged over five initializations. As a quantitative baseline, the last row of Tab. 3 shows the costs for matching the basis vectors to a standard autoencoder (AE) without decay, i.e. $\alpha = \beta = 0$. Apparent from Tab. 3, the non-negative models produce comparable basis images, whereas the AE produces basis images that cannot be easily matched even to other trials of the same method. That means that the standard AE converges to different solutions depending on the initialization. Non-negativity constrains the solution space drastically, which allows only a very specific set of solutions. This is reflected by the very similar encoding schemes of the non-negative methods irrespective of the optimization details and initial conditions.

6. Estimating the number of latent causes

Based on the idea to cluster neurons into used and unused categories, we show that the NNSAE can be utilized to estimate the intrinsic dimensionality of a dataset. Similar to the introductory example in Sec. 4 but for real-world datasets, we show that the number of used basis images saturates when the size of the hidden layer is increased. This self-regularization mechanism also prevents overfitting of the model to the training data.

6.1. Identifying important parts of digits

In Fig. 6 we show the $PMSE$ on the trainings and test set of the MNIST dataset depending on the number of neurons N in the hidden layer. For comparison, also the error of the standard autoencoder (AE, $\alpha = \beta = 0$) is shown. The reconstruction error decreases monotonously with increasing number of neurons for the NNSAE. The AE shows a superior reconstruction performance for most network sizes. Non-negativity additionally constrains the basic optimization problem which impairs the reconstruction performance slightly. However, the additional constraints also reduce overfitting in comparison to the AE: The offset between trainings and test errors is marginal for the NNSAE, whereas the AE displays – though only mild – overfitting for medium-sized networks (see magnified area in Fig. 6). Moreover, the performance of the AE decreases again for larger networks with $N > 400$ (see Fig. 6), whereas the $PMSE$ saturates for NNSAE networks with more than approximately 200 neurons. This indicates

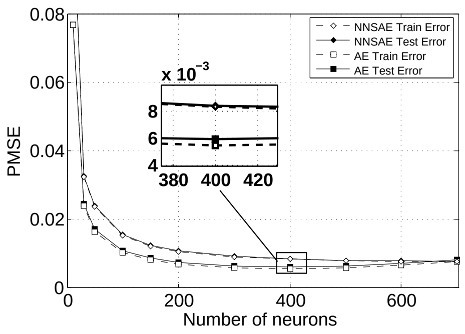
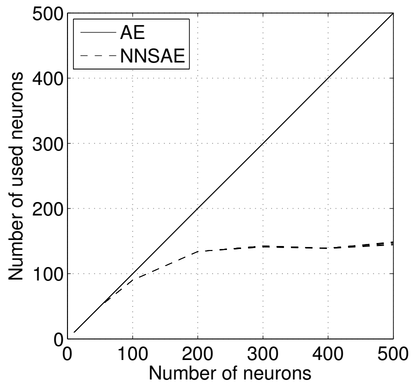


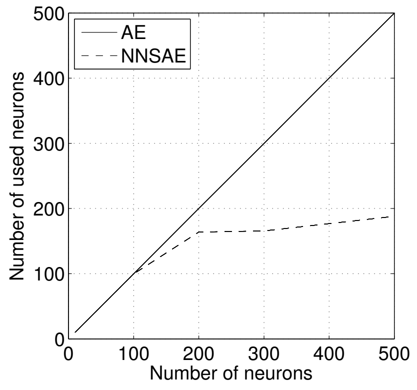
Fig. 6: Pixel-wise reconstruction error $PMSE$ on the MNIST dataset in logarithmic scale as function of the number of neurons in the hidden layer. Starting from $N = 10$ hidden neurons, the hidden layer size is increased up to $N = 700$ neurons. For comparison, we show the reconstruction error also for the standard autoencoder (AE) with $\alpha = \beta = 0$.

that the unconstrained optimization of the AE can not utilize the additional degrees of freedom anymore: Overparameterization makes it difficult for gradient learning to find a good local minimum and hence the network performance degrades. The NNSAE instead can still learn a suitable representation due to the strongly constrained parameter space. Non-negativity and sparseness of the NNSAE self-regularize the effective number of model parameters which prevents overfitting and helps learning in case of an overparameterized model.

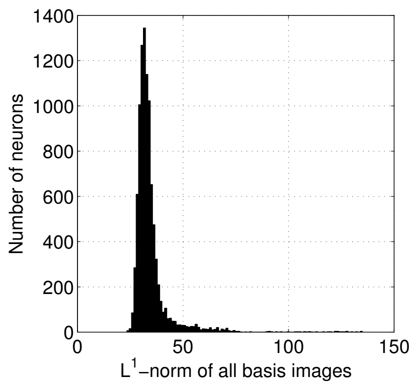
We proceed in more principle ways and estimate the intrinsic, parts-based data dimensionality by clustering the neurons of the NNSAE into used and unused categories following the observation in Sec. 4: Basis images of unused neurons have a very small maximum entry. We therefore cluster a neuron into the used or unused category based on the L^1-norm of its basis image. In Fig. 7(a) the number of used neurons is plotted versus the initial network size using different thresholds ($t \in \{1.5, 1.6, \ldots, 2.5\}$) for the clustering. The same set of thresholds is used for both method variants (NNSAE and AE). The solid and dashed lines in Fig. 7(a) are the resulting number of used neurons for the AE and the NNSAE, respectively. For the MNIST dataset, the number of used neurons in the NNSAE networks saturates approximately at 150, whereas in case of the AE the number of used neurons does not saturate, i.e. all neurons are used. This result confirms the self-regularizing effect we already observed in Sec. 4: The NNSAE develops a basis comprising a suitable amount of used neurons and a remaining set of unused neurons. Note further that the number of neurons recruited by the NNSAE can be interpreted as estimate of the intrinsic data dimensionality. For comparison, we conduct a principle component analysis on the MNIST dataset and estimate the intrinsic data dimensionality by the number of components needed to cover most of the data's variance. According to the eigenvalue spectrum, the biggest 150 (200) eigenvalues cover approximately 95% (97%) of the variance present in the data. This is conform with the estimate by the NNSAE, which allocates approximately 150 components to explain the data.



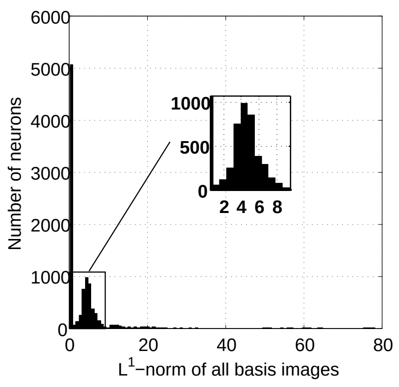
(a) MNIST



(b) CBCL faces



(c) L^1-norm of basis images for the AE.



(d) L^1-norm of basis images for the NNSAE.

Fig. 7: Evaluation of used and unused neurons for different network sizes. The amount of used basis images versus the initial network size is shown in Fig. 7(a) for the MNIST dataset and in Fig. 7(b) for the CBCL face dataset. Note that the number of used neurons saturates with increasing hidden layer size in case of the NNSAE (dashed lines), while the AE further recruits all neurons (solid lines). The basis images are classified as used or unused based on their L^1-norm. Histograms of the L^1-norm of the basis images for the MNIST dataset: Fig. 7(c) for the AE and Fig. 7(d) for the NNSAE.

We choose different thresholds t for the clustering in Fig. 7(a) to show the robustness of the resulting clusters against this parameter. To further illustrate and justify the thresholding into used and unused neurons, we show the distribution of the L^1-norm of basis images $\mathbf{w}_i$ over all neurons and networks of either the AE or NNSAE in Fig. 7(c) and Fig. 7(d). There are two major differences in the two histograms: First, the larger average L^1-norm of the AE compared to the NNSAE. The NNSAE has to distribute entries in the basis images sparely in order to not overload and blur the reconstructions. The AE in contrast produces densely occupied basis images with large positive and negative entries which consequently have a large L^1-norm. Second, in Fig. 7(d) a great amount of basis images have a L^1-norm close to zero building up a cluster of unused neurons. Automatic thresholding methods like the Otsu method [21] can easily separate this cluster of unused neurons from the used ones which makes pruning robust. Otsu's threshold selection method was originally intended to operate on histograms of gray-scale images. In the context of the NNSAE, the threshold is defined based on the histogram of L^1-norm values of the network's basis images $\mathbf{w}_i$ with $i=1,\ldots,N$. This technique is applied in the next section to prune the network and estimate the intrinsic data dimensionality automatically.

6.2. Identifying important parts of faces

We finally demonstrate the functioning of the NNSAE on the CBCL face dataset including the estimation of the data dimensionality.

6.2.1. Dataset and network training

The CBCL face dataset is extensively used by the MIT Center For Biological and Computation Learning to test the performance of face-detection systems[4]. The dataset provides 19×19 gray-scale images including 2.429 faces for training and 472 faces for testing. We apply a histogram normalization to the images in a preprocessing step before training and scale the pixel intensities into the range $[0.00, 0.25]$. Because of the small training set we increase the learning rates and set $\tilde{\eta}=0.05$ and $\eta_{IP}=0.0005$ in (Eq. 3) and (Eq. 5). Some example images from the test set are shown in the top row of Fig. 8. We use autoencoders with 361 input neurons and a variable size in the hidden layer to encode and decode the images. All other parameters are used as before.

6.2.2. Additive construction of faces

We first show the principle functioning of the NNSAE on the CBCL face dataset. We train a NNSAE with $N=200$ neurons for 100 epochs and then prune unused neurons utilizing Otsu's automatic thresholding method as mentioned in 6.1. The remaining network is trained two more epochs to fine-tune the network after pruning. Fig. 8 shows the reconstructions of some example images from the test set together with the encodings (rows two and three). The bottom panel in Fig. 8 displays some basis images learned by the NNSAE. The basis images represent typical parts of faces such as eyes, nose, cheeks and so on. The original images are accurately reconstructed by combining the basis images according to the sparse encodings.

We again measure the number of used neurons as function of the hidden layer size where we apply the Otsu method to automatically determine the threshold for parameter-free clustering of the basis images into used and unused ones. Fig. 7(b) shows that the NNSAE allocates approximately 170 neurons to represent the face images. This estimate of the intrinsic data dimensionality

[4]http://www.ai.mit.edu/projects/cbcl

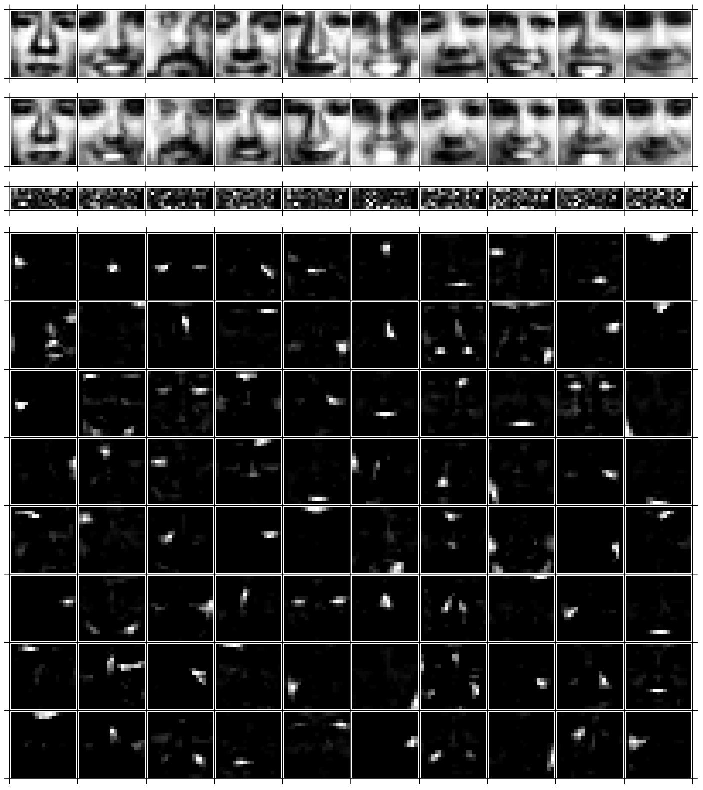


Fig. 8: Example images of the CBCL face dataset (top row). Reconstructions of the originals using a NNSAE with $N = 200$ neurons (second row). Encodings corresponding to the reconstructions (third row). Some basis images of the trained network (bottom panel). The basis images reflect parts of faces which are linearly combined to reconstruct the input images.

is in line with the principle component analysis, which covers 98% of the data variance if 170 components are used. Again, the number of used neurons saturates for larger hidden layer sizes which confirms the self-regularizing effect of the NNSAE learning also on a rather complex real-world face dataset. On the contrary, the AE uses all neurons to represent the data (solid line in Fig. 7(b)): The thresholding method provides a confidence value which is zero for all AE networks and indicates that a discrimination of neurons into used and unused based on their weight norm is not meaningful. In case of the NNSAE, the threshold confidence is very high for networks with $N > 100$ which emphasizes the robustness of the discrimination.

We conclude that the NNSAE can identify important parts of handwritten digits as well as faces. The learned basis sets have a rather stable size and form irrespective of initial conditions and hidden layer sizes. This makes the NNSAE a very robust and reliable learning technique. The synergy of non-negativity and sparseness results in a self-regularization of the network with respect to the allocated neural resources. Identification of recruited neurons is simple, robust and allows to estimate the latent, parts-based dimensionality of the data. Finally, self-regularization keeps the effective network parameters small which consequently prevents overfitting of the model.

7. Conclusion

We present an autoencoder for efficient online learning of sparse and non-negative encodings. Therefor, we combine a mechanism that enhances the sparseness of encodings with an asymmetric decay function that enforces non-negative network weights. The online trained autoencoder outperforms the offline techniques NMF and NMFSC when reconstructing images from the test set, underlining its generalization capabilities. Additionally, the autoencoder produces sparser encodings compared to the offline methods. We point out the general effect of non-negativity, i.e. restricting the solution space, which results in similar basis images that represent the latent causes of the data and are robustly found irrespective of the initialization. Moreover, the particular combination of IP and non-negative, error-driven learning in the NNSAE results in an optimal exploitation of activation versus reconstruction constraints. As a consequence, the autoencoder is robust to parameter variations and allocates always a similar number of neurons which develop a similar set of basis images. This self-regularization effect also prevents overfitting of the model to the training data.

In comparison to the common batch algorithms for non-negative coding, the main advantage of the NNSAE is the efficient encoding of novel inputs: the state of the network has simply to be updated with the new input, whereas the matrix factorization approaches in contrast require a full optimization step on the new data.

It is of particular interest to use the NNSAE in a stacked autoencoder network for pattern recognition in the future: Does the non-negative representation improve classification performance, and how is fine-tuning of the hierarchy, e.g. by backpropagation learning, affected by the asymmetric decay function? In this context, layer-wise pretraining with a subsequent pruning stage will reduce the dependence on hyper-parameters.

References

[1] M. W. Spratling, Learning image components for object recognition, Journal of Machine Learning Research 7 (2006) 793–815.

[2] K. Tanaka, Columns for complex visual object features in the inferotemporal cortex: Clustering of cells with similar but slightly different stimulus selectivities, Cerebral Cortex 13 (1) (2003) 90–99. doi:10.1093/cercor/13.1.90.

[3] D. D. Lee, H. S. Seung, Learning the parts of objects by non-negative matrix factorization, Nature 401 (6755) (1999) 788–791. doi:10.1038/44565.

[4] B. Olshausen, D. Field, Sparse coding of sensory inputs, Current Opinion in Neurobiology 14 (2004) 481–487. doi:10.1016/j.conb.2004.07.007.

[5] M. Ranzato, Y.-L. Boureau, Y. LeCun, Sparse Feature Learning for Deep Belief Networks, in: Proc. NIPS, 2008, pp. 1185–1192.

[6] G. E. Hinton, S. Osindero, Y.-W. Teh, A fast learning algorithm for deep belief nets, Neural Computation 18 (2006) 1527–1554.

[7] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: NIPS, 2007, pp. 153–160.

[8] H. Larochelle, P. Lamblin, Exploring strategies for training deep neural networks, Journal of Machine Learning Research 1 (2009) 1–40.

[9] A. Lemme, R. F. Reinhart, J. J. Steil, Efficient online learning of a non-negative sparse autoencoder, in: Proc. ESANN, 2010, pp. 1–6.

[10] J. Triesch, A gradient rule for the plasticity of a neuron's intrinsic excitability, Neural Computation (2005) 65–70.

[11] P. O. Hoyer, Non-negative sparse coding, in: Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing, pp. 557–565.

[12] P. O. Hoyer, Non-negative matrix factorization with sparseness constraints, Journal of Machine Learning Research 5 (2004) 1457–1469.

[13] B. Shastri, M. Levine, Face recognition using localized features based on non-negative sparse coding, Machine Vision and Applications 18 (2) (2007) 107–122.

[14] D. Dornbusch, R. Haschke, S. Menzel, H. Wersing, Correlating shape and functional properties using decomposition approaches, FLAIRS-23 (2010) 398–403.

[15] H. Wersing, E. Körner, Learning optimized features for hierarchical models of invariant object recognition, Neural computation 15 (7) (2003) 1559–1588.

[16] I. Bax, G. Heidemann, H. Ritter, Hierarchical feed-forward network for object detection tasks, Optical Engineering 45 (6).

[17] M. D. Plumbley, E. Oja, A ”nonnegative PCA” algorithm for independent component analysis, IEEE Transactions on Neural Networks 15 (2004) 66–76. doi:10.1109/TNN.2003.820672.

[18] J. J. Steil, Backpropagation-decorrelation: recurrent learning with O(N) complexity, in: Proc. IJCNN, Vol. 1, 2004, pp. 843–848.

[19] P. Földiák, Forming sparse representations by local anti-Hebbian learning, Biological Cybernetics 64 (1990) 165–170.

[20] A. Frank, On Kuhn's Hungarian Method - A tribute from Hungary, Naval Research Logistics 52 (2005) 2–5. doi:10.1002/nav.20056.

[21] N. Otsu, A threshold selection method from gray-level histograms, Automatica 11 (1975) 285.