Rectified Factor Networks and Dropout

Djork-Arné Clevert Thomas Unterthiner
Sepp Hochreiter
Institute of Bioinformatics
Johannes Kepler University, Linz, Austria
{okko,unterthiner,hochreit}@bioinf.jku.at

Abstract

The success of deep learning techniques is based on their robust, effective and abstract representations of the input. In particular, sparse representations that are obtained from rectified linear units and dropout increased classification performance at various tasks. Deep architectures are often constructed by unsupervised pretraining and stacking of either restricted Boltzmann machines (RBMs) or autoencoders. We propose rectified factor networks (RFNs) for pretraining of deep networks. In contrast to RBMs and autoencoders, RFNs (1) estimate the noise of each input component, (2) aim at decorrelating the hidden units (factors), (3) estimate the precision of hidden units by the posterior variance. In the E-step of an EM algorithm, RFN learning (i) enforces non-negative posterior means, (ii) allows dropout of hidden units, and (iii) normalizes the signal part of the hidden units. In the M-step, RFN learning applies gradient descent along the Newton direction to allow rectifying, dropout, and fast GPU implementations. RFN learning can be considered as a variational EM algorithm with unknown prior which is estimated during maximizing the likelihood. Using a fixed point analysis, we show RFNs explain the data variance like factor analysis.

RFNs produce sparse and non-linear input representations for new data by a linear mapping and subsequent rectification, therefore can be readily used for pretraining of deep networks. It is tailored to making full use of large hidden layers with respect to both using all of them to code the input and computational complexity.

We tested and compared RFNs for unsupervised pretraining of deep learning on nine different benchmark datasets: MNIST, basic MNIST, bg-rand MNIST, bg-img MNIST, rect (tall vs. wide rectangles), rect-img, convex (convex vs. concave shapes), NORB and CIFAR-10. Competitors where support vector machines, deep learning with stacking denoising autoencoders, deep learning with stacking regular autoencoders, and deep learning with stacking restricted Boltzmann machines. Only on bg-rand RFN pretraining was significantly worse than the best performing method, though it was second. On basic MNIST, bg-img MNIST, rect-img, rect, convex and NORB, RFNs pretraining outperformed all other methods — in four cases significantly.

1 Introduction

The advent of deep learning [17, 4] and its success in both academic challenges and industrial applications [6, 22, 12, 5] is based on better input representations compared to previous approaches. Input codes at higher levels have more abstract representations that capture complex and non-linear explanatory factors underlying the observed input [3]. Rectified linear units (ReLU) [26, 10, 32] and dropout [18, 31] improved the performance of deep learning methods in several supervised benchmark datasets. ReLU lead to sparse representations [10] which are further sparsified by dropout [1].

Therefore, beyond complex and abstract representation on different levels, sparse representations enhance the performance of subsequent supervised techniques. Sparse representations were originally motivated by findings in sensory neural systems and are well known objectives in machine learning [28, 16, 9]. In bioinformatics sparse codes excelled in biclustering of gene expression data [20] and in finding DNA sharing patterns between humans, Neanderthals and Denisovans [19]. However constructing a sparse code can be computationally expensive [13], while ReLU and dropout are fast and, therefore, are perfectly suited for deep learning with many representational units.

Pretraining deep networks with unsupervised sparse coding methods is very appealing because unlabeled data can be exploited to enhance data representation and to extract robust structures in the data. Conventional sparse coding methods adjust the code of new data and, therefore, are no appropriate for pretraining deep networks, which have weights that are independent of the input. These methods either construct a code for the observed data but not for new data or require an iterative update to compute a code for new data [28, 13]. Also generative models with sparse priors have for each input a different posterior variance which again makes the coding weights dependent on the input [9, 29]. Furthermore, generative models with sparse priors do not ensure sparse posteriors, though the latter represent the inputs. For example, rectified factor analysis, which rectifies Gaussian priors and selects models using a variational Bayesian learning procedure, does not yield sparse posteriors [14, 15].

ReLU and dropout produce sparse codes for the test data without iterative code updates. Currently, ReLU and dropout are used in connection with restricted Boltzmann machines (RBMs) [26, 18] and neural networks [10, 32, 31]. However during unsupervised pretraining, neither RBMs nor neural autoencoders (1) estimate the noise of the visible units, (2) explicitly decorrelate the hidden (code) units, (3) estimate the precision of hidden units. First, without input noise estimates, very noisy inputs are forced to be reconstructed via large errors in autoencoders or large input-hidden coactivations in RBMs. Furthermore, errors of highly predictive inputs with low noise levels should be up-weighted to detect more structures at which they participate. Consequently, input noise estimations can hint at interesting parts in the input. Secondly, many hidden units may capture the same, most dominant input structures while rare or small structures are missed. Decorrelation of hidden units will force more variety into hidden units and helps to detect rare or small input structures. Thirdly, RBMs and autoencoders estimate neither the precision nor the information content of hidden units. Thus, larger activations of the ReLU hidden units need not indicate more information, more precision, or more consistent input structures. We suggest to address issues (1)–(3) by a factor analysis model as an alternative to RBMs and autoencoders.

2 Rectified Factor Network

We propose to use factor analysis with many hidden units (factors) for unsupervised pretraining of deep networks to estimate the noise of each visible unit, to decorrelate the hidden units, and to estimate the information content, precision, and consistency of input structures captured by the hidden units. The factor analysis model is

$$\boldsymbol{v} \;=\; \boldsymbol{W}\boldsymbol{h} \;+\; \boldsymbol{\epsilon}\,. \tag{1}$$

The prior $\boldsymbol{h} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$ on the hidden units $\boldsymbol{h} \in \mathbb{R}^l$, i.e. the factors, and the noise $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Psi})$ on visible units $\boldsymbol{v} \in \mathbb{R}^m$, i.e. the observations, are independent. The model parameters are the weight (factor loading) matrix $\boldsymbol{W} \in \mathbb{R}^{m \times l}$ and the noise covariance matrix $\boldsymbol{\Psi} \in \mathbb{R}^{m \times m}$. $\boldsymbol{\Psi}$ is assumed to be diagonal in order to explain correlations between input components by the hidden units (signal) and not by correlated noise. The linear factor analysis model with independent additive Gaussian noise is depicted in Fig. 1.

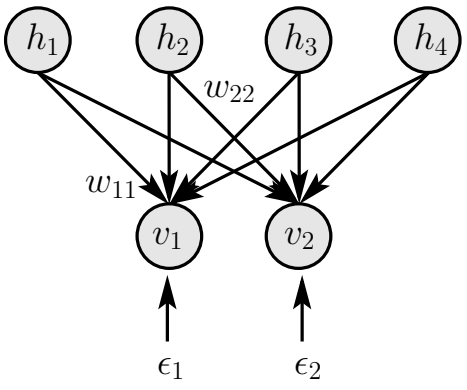


Figure 1: Factor analysis model with hidden units (factors) $\boldsymbol{h}$, visible units (observations) $\boldsymbol{v}$, weight (factor loading) matrix $\boldsymbol{W}$, and noise $\boldsymbol{\epsilon}$.

Given the mean-centered data $\{\boldsymbol{v}\} = \{\boldsymbol{v}_1, \ldots, \boldsymbol{v}_n\}$ and the covariance matrix $\boldsymbol{C} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v}_i \boldsymbol{v}_i^T$, the

expectation maximization (EM) update for maximizing the likelihood of the factor analysis model is

$$
\begin{aligned}
\textbf{E-step:} \quad & \boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i} \;=\; \boldsymbol{W}^T \left(\boldsymbol{W}\,\boldsymbol{W}^T \;+\; \boldsymbol{\Psi}\right)^{-1} \boldsymbol{v}_i \;=\; \left(\boldsymbol{I} \;+\; \boldsymbol{W}^T\boldsymbol{\Psi}^{-1}\boldsymbol{W}\right)^{-1} \boldsymbol{W}^T\boldsymbol{\Psi}^{-1}\,\boldsymbol{v}_i \;, \\
& \boldsymbol{\Sigma}_{\boldsymbol{h}_i|\boldsymbol{v}_i} \;=\; \boldsymbol{I} \;-\; \boldsymbol{W}^T \left(\boldsymbol{W}\,\boldsymbol{W}^T \;+\; \boldsymbol{\Psi}\right)^{-1} \boldsymbol{W} \;=\; \left(\boldsymbol{I} \;+\; \boldsymbol{W}^T\boldsymbol{\Psi}^{-1}\boldsymbol{W}\right)^{-1} \;, \\
& \mathrm{E}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\left(\boldsymbol{h}_i\right) \;=\; \boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i} \;, \qquad \mathrm{E}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\left(\boldsymbol{h}_i\,\boldsymbol{h}_i^T\right) \;=\; \boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\,\boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i}^T \;+\; \boldsymbol{\Sigma}_{\boldsymbol{h}_i|\boldsymbol{v}_i} \\
& \boldsymbol{U} \;=\; \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{v}_i\,\mathrm{E}_{\boldsymbol{h}_i|\boldsymbol{v}_i}^T\left(\boldsymbol{h}_i\right) \;, \qquad \boldsymbol{S} \;=\; \frac{1}{n}\sum_{i=1}^{n} \mathrm{E}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\left(\boldsymbol{h}_i\,\boldsymbol{h}_i^T\right) \\
\textbf{M-step:} \quad & \boldsymbol{W} \;=\; \boldsymbol{U}\,\boldsymbol{S}^{-1} \;, \qquad \boldsymbol{\Psi} \;=\; \mathrm{diag}\left(\boldsymbol{C} \;-\; \boldsymbol{U}\,\boldsymbol{W}^T \;-\; \boldsymbol{W}\,\boldsymbol{U}^T \;+\; \boldsymbol{W}\,\boldsymbol{S}\,\boldsymbol{W}^T\right) \;.
\end{aligned}
$$

In this update rule the hidden unit posterior $p(\boldsymbol{h}_i \mid \boldsymbol{v}_i)$ is Gaussian with mean $\boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i}$ and covariance matrix $\boldsymbol{\Sigma}_{\boldsymbol{h}_i|\boldsymbol{v}_i}$. In the EM algorithm, the posterior is computed in the E-step for the old parameters $(\boldsymbol{W}, \boldsymbol{\Psi})$ which are then updated in the M-step. $\boldsymbol{U}$ is the basic update term for weight matrix $\boldsymbol{W}$ which is essentially Hebb's rule between hidden and visible units. As mentioned in the introduction, factor analysis (1) estimates the noise $\boldsymbol{\Psi}$ of visible units, (2) decorrelates the hidden units by multiplying $\boldsymbol{U}$ with $\boldsymbol{S}^{-1}$, (3) estimates the information content and the precision of hidden units via $\boldsymbol{\Sigma}_{\boldsymbol{h}_i|\boldsymbol{v}_i}$.

We aim to introduce rectifying and dropout of the hidden unit posterior into this EM framework to construct sparse codes. Besides that, we also need the rectifier to create a non-linear input representations, which allows for stacking RFNs, as without any non-linear activation function on the hidden unit every stacked linear model would collapse to one layer.

Rectifying, that is non-negative posterior means, can be implemented by the posterior constraint method [7, 11]. However dropout and normalization, which are essential for our approach, cannot be treated by posterior constraints. Therefore, we modify above EM algorithm with respect to four items:

(i) **rectifying**: $\left[\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}\right]_j = \max\left\{0, \left[\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}\right]_j\right\}$ to enforce non-negative posterior means.

(ii) **dropout**: $\left[\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}\right]_j = \delta \;\left[\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}\right]_j$ with $\delta \in \{0, 1\}$ as dropout variable.

(iii) **normalizing**: $\frac{1}{n}\sum_i \left[\boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\right]_j^2 = 1$.

(iv) **gradient descent**: the M-step is a gradient descent step in the Newton direction.

Rectifying and dropout in steps (i) and (ii), respectively, enforce sparse codes $\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}$. Normalizing the signal part $\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}$ of the hidden units in step (iii) is an important step for models with many hidden units. If the signal-to-noise ratio is varying across samples or if the noise is correlated with the signal, then factor analysis tends to explain signals away by noise. Normalizing the hidden unit's signal part, forces the model to explain correlation among visible units by signal which otherwise would be explained by noise and thereby finds a wider representation of the visible units. Step (iv), the gradient descent update, is important to keep the current solution and to avoid that extreme cases of rectifying or dropout corrupt it. Furthermore, this step is advantageous for using stochastic gradient and fast GPU implementations. The RFN learning algorithm with learning rates η_Ψ and η_W, dropout rate d, and the lower bound $\Psi_{\min}$ on Ψ_{kk} is

$$
\begin{aligned}
\textbf{E-step (1):} \quad & \boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}} \;=\; \left(\boldsymbol{I} \;+\; \boldsymbol{W}^T\boldsymbol{\Psi}^{-1}\boldsymbol{W}\right)^{-1} \boldsymbol{W}^T\boldsymbol{\Psi}^{-1}\,\boldsymbol{v} \;, && (2) \\
\textbf{Dropout:} \quad & \Pr(\delta = 0) \;=\; d \;\;, \;\; \Pr(\delta = 1) \;=\; 1 \;-\; d \\
\textbf{Rectifier:} \quad & \left[\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}\right]_j \;=\; \delta\;\max\left\{0, \left[\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}\right]_j\right\} && (3) \\
\textbf{Normalizer:} \quad & \frac{1}{n}\sum_i \left[\boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\right]_j^2 \;=\; 1
\end{aligned}
$$

E-step (2): $\boldsymbol{\Sigma}_{\boldsymbol{h}|\boldsymbol{v}} \;=\; \left(\boldsymbol{I} \;+\; \boldsymbol{W}^T \boldsymbol{\Psi}^{-1} \boldsymbol{W}\right)^{-1} \;,$ (4)

$$\mathrm{E}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\left(\boldsymbol{h}_i\right) \;=\; \boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i} \quad , \quad \mathrm{E}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\left(\boldsymbol{h}_i\, \boldsymbol{h}_i^T\right) \;=\; \boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\, \boldsymbol{\mu}_{\boldsymbol{h}_i|\boldsymbol{v}_i}^T \;+\; \boldsymbol{\Sigma}_{\boldsymbol{h}_i|\boldsymbol{v}_i} \tag{5}$$

$$\boldsymbol{U} \;=\; \frac{1}{n} \sum_{i=1}^{n} \boldsymbol{v}_i\, \mathrm{E}_{\boldsymbol{h}_i|\boldsymbol{v}_i}^T\left(\boldsymbol{h}_i\right) \quad , \quad \boldsymbol{S} \;=\; \frac{1}{n} \sum_{i=1}^{n} \mathrm{E}_{\boldsymbol{h}_i|\boldsymbol{v}_i}\left(\boldsymbol{h}_i\, \boldsymbol{h}_i^T\right) \tag{6}$$

M-step: $\boldsymbol{C} \;=\; \frac{1}{n} \sum_{k=1}^{n} \boldsymbol{v}_k\, \boldsymbol{v}_k^T \quad , \quad c_k \;=\; \left[\boldsymbol{C} \;-\; \boldsymbol{U}\, \boldsymbol{W}^T \;-\; \boldsymbol{W}\, \boldsymbol{U}^T \;+\; \boldsymbol{W}\, \boldsymbol{S}\, \boldsymbol{W}^T\right]_{kk}$

$$\boldsymbol{W} \;=\; \boldsymbol{W} \;+\; \eta_W\, \Delta \boldsymbol{W} \quad , \quad \Delta \boldsymbol{W} \;=\; \boldsymbol{U}\, \boldsymbol{S}^{-1} \;-\; \boldsymbol{W} \tag{7}$$

$$\Psi_{kk} \;=\; \max\{\Psi_{\min} \,,\, \Psi_{kk} \;+\; \eta_\Psi\, \Delta \Psi_{kk}\} \quad , \quad \Delta \Psi_{kk} \;=\; c_k \;-\; \Psi_{kk} \,. \tag{8}$$

3 Analysis of RFN Learning

In the next subsection RFN learning is formulated as a variational EM algorithm that maximizes the data likelihood for an unknown prior. In the following subsection we show via a fixed point analysis that RFNs explain the data variance like factor analysis. Finally, we derive the Newton update rule of the RFN learning.

3.1 Variational EM

The variational EM framework [8, 27, 21, 2, 9, 29] is utilized to describe RFN learning. We parametrize a Gaussian prior by the variational parameter $\boldsymbol{\xi}$:

$$p(\boldsymbol{h}; \boldsymbol{\xi}) \;=\; \mathcal{N}(\boldsymbol{h}; \boldsymbol{\xi}, \boldsymbol{I}) \;=\; (2\pi)^{-\frac{l}{2}}\, \exp\left(-\frac{1}{2}\, (\boldsymbol{h} \;-\; \boldsymbol{\xi})^T\, (\boldsymbol{h} \;-\; \boldsymbol{\xi})\right) \,. \tag{9}$$

Using a variational distribution $Q(\boldsymbol{h} \mid \boldsymbol{v})$, the data likelihood can be bounded via the following equation:

$$\log p(\boldsymbol{v}) \;-\; D_{\mathrm{KL}}(Q(\boldsymbol{h} \mid \boldsymbol{v}) \parallel p(\boldsymbol{h} \mid \boldsymbol{v})) \;=\; -\int Q(\boldsymbol{h} \mid \boldsymbol{v})\, \log \frac{Q(\boldsymbol{h} \mid \boldsymbol{v})}{p(\boldsymbol{h}, \boldsymbol{v})}\, d\boldsymbol{h} \tag{10}$$

$$= \int Q(\boldsymbol{h} \mid \boldsymbol{v})\, \log p(\boldsymbol{v} \mid \boldsymbol{h})\, d\boldsymbol{h} \;-\; D_{\mathrm{KL}}(Q(\boldsymbol{h} \mid \boldsymbol{v}) \parallel p(\boldsymbol{h})) \;=\; \mathcal{F}(\boldsymbol{W}, \boldsymbol{\Psi}, \boldsymbol{\xi} \mid \hat{\boldsymbol{W}}, \hat{\boldsymbol{\Psi}}, \hat{\boldsymbol{\xi}}) \,,$$

where D_{KL} denotes the Kullback-Leibler divergence. $(\hat{\boldsymbol{W}}, \hat{\boldsymbol{\Psi}}, \hat{\boldsymbol{\xi}})$ are the actual parameter estimates that describe $Q(\boldsymbol{h} \mid \boldsymbol{v})$, $(\boldsymbol{W}, \boldsymbol{\Psi})$ are the parameters of the posterior $p(\boldsymbol{v} \mid \boldsymbol{h})$, and $\boldsymbol{\xi}$ are the parameters of the prior $p(\boldsymbol{h})$. The variational EM maximizes $\mathcal{F}$ in its M-step with respect to the parameters and in its E-step with respect to Q. The E-step is equivalent to minimizing $D_{\mathrm{KL}}(Q(\boldsymbol{h} \mid \boldsymbol{v}) \parallel p(\boldsymbol{h} \mid \boldsymbol{v}))$, therefore it tightens the lower bound $\mathcal{F}$ on the log-likelihood $\log p(\boldsymbol{v})$. For the M-step, the objective can be divided into $\int Q(\boldsymbol{h} \mid \boldsymbol{v}) \log p(\boldsymbol{v} \mid \boldsymbol{h}) d\boldsymbol{h}$ which only depends on $(\boldsymbol{W}, \boldsymbol{\Psi})$ and must be maximized and $D_{\mathrm{KL}}(Q(\boldsymbol{h} \mid \boldsymbol{v}) \parallel p(\boldsymbol{h}))$ which only depends on $\boldsymbol{\xi}$ and must be minimized.

Instead of optimizing Q as in the variational EM, we consider Q as being given via rectifying, dropout, and normalization. However, neither the variational parameter nor the posterior are known. Therefore we minimize $D_{\mathrm{KL}}(Q(\boldsymbol{h} \mid \boldsymbol{v}) \parallel p(\boldsymbol{h} \mid \boldsymbol{v}))$ with respect to $\boldsymbol{\xi}$ for the given $Q(\boldsymbol{h} \mid \boldsymbol{v})$. The posterior $p(\boldsymbol{h} \mid \boldsymbol{v})$ is a Gaussian with mean and covariance matrix:

$$\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}} \;=\; \left(\boldsymbol{I} \;+\; \boldsymbol{W}^T \boldsymbol{\Psi}^{-1} \boldsymbol{W}\right)^{-1} \left(\boldsymbol{W}^T \boldsymbol{\Psi}^{-1}\, \boldsymbol{v} \;+\; \boldsymbol{\xi}\right) \quad , \quad \boldsymbol{\Sigma}_{\boldsymbol{h}|\boldsymbol{v}} \;=\; \left(\boldsymbol{I} \;+\; \boldsymbol{W}^T \boldsymbol{\Psi}^{-1} \boldsymbol{W}\right)^{-1} \,. \tag{11}$$

Minimizing $D_{\mathrm{KL}}(Q(\boldsymbol{h} \mid \boldsymbol{v}) \parallel p(\boldsymbol{h} \mid \boldsymbol{v}))$ gives

$$\boldsymbol{\xi} \;=\; (\boldsymbol{W}^T \boldsymbol{\Psi}^{-1} \boldsymbol{W} \;+\; \boldsymbol{I})\, \boldsymbol{\mu}_Q \;-\; \boldsymbol{W}^T \boldsymbol{\Psi}^{-1} \boldsymbol{v} \,, \tag{12}$$

where $\boldsymbol{\mu}_Q$ is the modified $\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}}$ according to above steps (i)—(iii) (rectifying, dropout, normalizing). $Q(\boldsymbol{h} \mid \boldsymbol{v})$ and $p(\boldsymbol{h} \mid \boldsymbol{v})$ have the same covariance matrix and we enforced $\boldsymbol{\mu}_{\boldsymbol{h}|\boldsymbol{v}} = \boldsymbol{\mu}_Q$, thus the Kullback-Leibler divergence between them is zero. Therefore $\boldsymbol{\xi}$ makes the bound $\mathcal{F}$ on $\log p(\boldsymbol{v})$

tight and can even be used to estimate the model prior. The performance of the model in explaining the data depends on the amount of information in $\boldsymbol{v}$ that is conveyed to $\boldsymbol{\mu}_Q$. Dropout is covered by this framework if $\boldsymbol{\xi}$ is viewed as a random variable. Consequently, RFN learning maximizes the likelihood for a prior that is constructed during learning. Besides the architecture, the complexity of the model class depends on how much input information is conveyed to the hidden units after the posterior modification.

3.2 Fixed Point

For rectified factor networks (RFNs) the objective is unknown during learning because the prior and, therefore, also the likelihood are not given. We derive a fixed point of the algorithm which is independent of the posterior modification. The fixed point solution explains the variation in the data like factor analysis.

The fixed point equation for the $\boldsymbol{W}$ update is

$$\Delta \boldsymbol{W} \ = \ \boldsymbol{U} \, \boldsymbol{S}^{-1} \ - \ \boldsymbol{W} \ = \ \boldsymbol{0} \ \Rightarrow \ \boldsymbol{U} \ - \ \boldsymbol{W} \, \boldsymbol{S} \ = \ \boldsymbol{0} \ \Rightarrow \ \boldsymbol{U} \, \boldsymbol{W}^T \ - \ \boldsymbol{W} \, \boldsymbol{S} \, \boldsymbol{W}^T \ = \ \boldsymbol{0} \, , \quad (13)$$

where for the last equation we multiplied the previous from the right hand side by $\boldsymbol{W}^T$. The fixed point equation for the full (not only diagonal) $\boldsymbol{\Psi}$ update is

$$\boldsymbol{\Psi} \ = \ \boldsymbol{C} \ - \ \boldsymbol{U} \, \boldsymbol{W}^T \ - \ \boldsymbol{W} \, \boldsymbol{U}^T \ + \ \boldsymbol{W} \, \boldsymbol{S} \, \boldsymbol{W}^T \ = \ \boldsymbol{C} \ - \ \boldsymbol{W} \, \boldsymbol{U}^T \, , \quad (14)$$

where we inserted Eq. (13). Since $\boldsymbol{W} \ \boldsymbol{S} \ \boldsymbol{W}^T$ in Eq. (13) is symmetric, we can insert $\boldsymbol{W} \ \boldsymbol{U}^T = \boldsymbol{U} \ \boldsymbol{W}^T = \boldsymbol{C} - \boldsymbol{\Psi}$ in last Eq. (13):

$$\boldsymbol{C} \ = \ \boldsymbol{\Psi} \ + \ \boldsymbol{W} \, \boldsymbol{S} \, \boldsymbol{W}^T \, . \quad (15)$$

Therefore the model that corresponds to the fixed point explains the data covariance matrix $\boldsymbol{C}$ by a noise part $\boldsymbol{\Psi}$ and a signal part $\boldsymbol{W} \ \boldsymbol{S} \ \boldsymbol{W}^T$. Like factor analysis the data variance is explained by the model via the parameters $\boldsymbol{\Psi}$ (noise) and $\boldsymbol{W}$ (signal). $\boldsymbol{\Psi} = \frac{1}{n} \sum_{i=1}^n \boldsymbol{\epsilon}_i \boldsymbol{\epsilon}_i^T + \boldsymbol{W} \boldsymbol{\Sigma}_{\boldsymbol{h}_i | \boldsymbol{v}_i} \boldsymbol{W}^T$ explains both reconstruction errors $\boldsymbol{\epsilon}_i = \boldsymbol{v}_i - \boldsymbol{\mu}_{\boldsymbol{h}_i | \boldsymbol{v}_i} \boldsymbol{W}$ and uncertainties of hidden units captured by $\boldsymbol{\Sigma}_{\boldsymbol{h}_i | \boldsymbol{v}_i}$. If $\boldsymbol{\mu}_{\boldsymbol{h}_i | \boldsymbol{v}_i}$ contains sufficient information on the visible units $\boldsymbol{v}_i$, then the fixed point equation holds also for restricting $\boldsymbol{\Psi}$ to a diagonal matrix. In this case the reconstruction errors approach zero (forced by the M-step) and $\boldsymbol{W} \boldsymbol{\Sigma}_{\boldsymbol{h}_i | \boldsymbol{v}_i} \boldsymbol{W}^T$ is diagonal dominant and explained by $\boldsymbol{\Psi}$. Dropout is covered by this fixed point analysis. For dropout $\boldsymbol{\mu}_{\boldsymbol{h}_i | \boldsymbol{v}_i}$ become random variables, thus $\boldsymbol{U}$ and $\boldsymbol{S}$ are replaced by their expectations. How much data variation is explained by signal $\boldsymbol{W} \ \boldsymbol{S} \ \boldsymbol{W}^T$ and how much is explained by noise $\boldsymbol{\Psi}$ depends on (a) the model complexity like the number of factors, (b) the measurement noise on the visible units, and (c) the amount of information about the visible units $\boldsymbol{v}$ that is coded in the hidden units $\boldsymbol{\mu}_{\boldsymbol{h} | \boldsymbol{v}}$.

3.3 Newton Direction

We now derive the Newton update rule and address step (iv) from above. The objective to maximize is

$$\mathcal{L} \ = \ \int_{\mathbb{R}^l} Q_i(\boldsymbol{h}_i \mid \boldsymbol{v}_i) \ \log \left(p(\boldsymbol{v}_i \mid \boldsymbol{h}_i; \boldsymbol{W}, \boldsymbol{\Psi}) \right) \ d\boldsymbol{h}_i \, , \quad \log \mathcal{L} \ = \ - \ \frac{m \, n}{2} \log \left(2 \pi \right) \ - \ \frac{n}{2} \log |\boldsymbol{\Psi}|$$
$$- \ \frac{1}{2} \ \sum_{i=1}^n \boldsymbol{v}_i^T \boldsymbol{\Psi}^{-1} \ \boldsymbol{v}_i \ + \ \sum_{i=1}^n \mathrm{E}_{\boldsymbol{h}_i | \boldsymbol{v}_i} \left(\boldsymbol{v}_i^T \boldsymbol{\Psi}^{-1} \boldsymbol{W} \boldsymbol{h}_i \right) \ \frac{1}{2} \ \mathrm{E}_{\boldsymbol{h}_i | \boldsymbol{v}_i} \left(\sum_{i=1}^n \boldsymbol{h}_i^T \ \boldsymbol{W}^T \boldsymbol{\Psi}^{-1} \ \boldsymbol{W} \ \boldsymbol{h}_i \right) \quad (16)$$

The Hessian $\boldsymbol{H}_{\boldsymbol{W}}$ of $(\frac{2}{n} \log \mathcal{L})$ with respect to $\boldsymbol{W}$ as a vector is:

$$\boldsymbol{H}_{\boldsymbol{W}} \ = \ \frac{\partial \mathrm{vec} \left(\frac{2}{n} \nabla_{\boldsymbol{W}} \log \mathcal{L} \right)}{\partial \mathrm{vec}(\boldsymbol{W})^T} \ = \ \frac{\partial \mathrm{vec} \left(\boldsymbol{\Psi}^{-1} \, \boldsymbol{U} \ - \ \boldsymbol{\Psi}^{-1} \, \boldsymbol{W} \, \boldsymbol{S} \right)}{\partial \mathrm{vec}(\boldsymbol{W})^T} \ = \ - \, \boldsymbol{S} \ \otimes \ \boldsymbol{\Psi}^{-1} \, , \quad (17)$$

where $\otimes$ is the Kronecker product of matrices. For the product of the negative inverse Hessian with the gradient we have:

$$- \ \boldsymbol{H}_{\boldsymbol{W}}^{-1} \ \mathrm{vec} \left(\boldsymbol{\Psi}^{-1} \, \boldsymbol{U} \ - \ \boldsymbol{\Psi}^{-1} \, \boldsymbol{W} \, \boldsymbol{S} \right) \ = \ \left(\boldsymbol{S}^{-1} \ \otimes \ \boldsymbol{\Psi} \right) \ \mathrm{vec} \left(\boldsymbol{\Psi}^{-1} \, \boldsymbol{U} \ - \ \boldsymbol{\Psi}^{-1} \, \boldsymbol{W} \, \boldsymbol{S} \right) \quad (18)$$
$$= \ \mathrm{vec} \left(\boldsymbol{\Psi} \ \left(\boldsymbol{\Psi}^{-1} \, \boldsymbol{U} \ - \ \boldsymbol{\Psi}^{-1} \, \boldsymbol{W} \, \boldsymbol{S} \right) \boldsymbol{S}^{-1} \right) \ = \mathrm{vec} \left(\boldsymbol{U} \, \boldsymbol{S}^{-1} \ - \ \boldsymbol{W} \right) \, .$$

Thus, if we apply a Newton update then the update direction for $\boldsymbol{W}$ in the M-step is

$$\Delta \boldsymbol{W} \ = \ \boldsymbol{U} \ \boldsymbol{S}^{-1} \ - \ \boldsymbol{W} \ . \tag{19}$$

This is the exact EM update if the stepsize is 1. Since the objective is a quadratic function in $\boldsymbol{W}$, one Newton update would lead to the exact solution. However for rectified linear posteriors and for dropout we want to have small changes of the matrix $\boldsymbol{W}$. The Newton update converges fast, but comes at the cost of inverting $\boldsymbol{S}$, which is computational expensive, thus for large $\boldsymbol{S}$ matrices $\Delta \boldsymbol{W} \ = \ \boldsymbol{U} \ - \ \boldsymbol{W} \ \boldsymbol{S}$ is a reasonable alternative update rule.

The Hessian $\boldsymbol{H}_{\boldsymbol{\Psi}}$ of $(\frac{2}{n} \log \mathcal{L})$ with respect to $\boldsymbol{\Psi}$ as a vector is a diagonal matrix with $[H_{\boldsymbol{\Psi}}]_{ij} = 0$ for $i \neq j$ and for $i = j$:

$$[H_{\boldsymbol{\Psi}}]_{ii} \ = \ \frac{1}{\Psi_{ii}^2} \ - \ \frac{2 \, c_i}{\Psi_{ii}^3} \ = \ \frac{1}{\Psi_{ii}^2} \left(1 \ - \ \frac{2 \, c_i}{\Psi_{ii}} \right) \ , \ c_i \ = \ \left[\boldsymbol{C} \ - \ \boldsymbol{U} \ \boldsymbol{W}^T \ - \ \boldsymbol{W} \ \boldsymbol{U}^T \ + \ \boldsymbol{W} \ \boldsymbol{S} \ \boldsymbol{W}^T \right]_{ii} \tag{20}$$

Since we maximize the bound, we have to ensure that the Hessian is negative definite, that is, $[H_{\boldsymbol{\Psi}}]_{ii} \ < \ 0$ or $\Psi_{ii} \ < \ 2 \ c_i$. Therefore we replace c_i by Ψ_{ii} and get $[H_{\boldsymbol{\Psi}}]_{ii} \ = \ -1/\Psi_{ii}^2$. The Newton update direction is:

$$\Delta \Psi_{ii} \ = \ - \ \frac{[\nabla_{\boldsymbol{\Psi}} \log \mathcal{L}]_{ii}}{[H_{\boldsymbol{\Psi}}]_{ii}} \ = \ c_i \ - \ \Psi_{ii} \ . \tag{21}$$

This is the EM update for learning rate equal to 1.

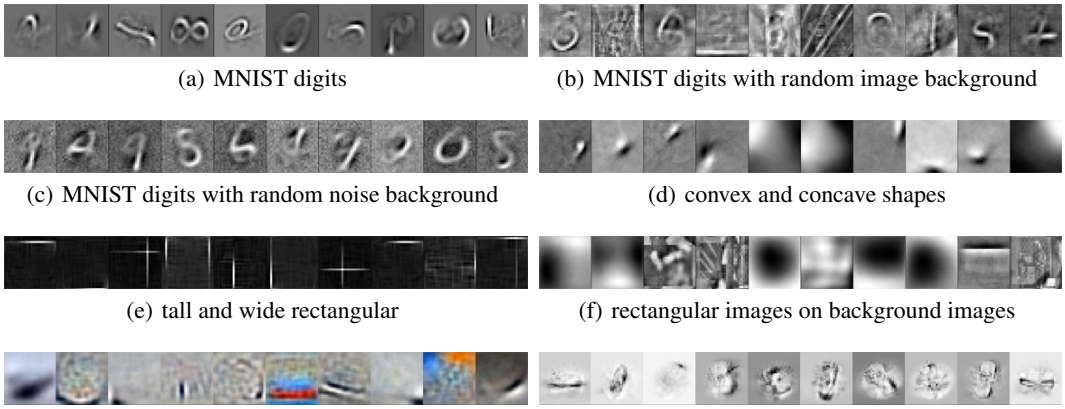
4 Experiments

In this section, we assess the performance of rectified factor networks (RFNs) as a pretraining procedure to construct deep networks. We stacked RFNs in the same way as described by Vincent et al.[30], namely first training a single layer RFN and then passing the resulting representation as input for training the next RFN.

We conducted experiments with two deep network architectures constructed from pretrained RFNs: a 1-hidden layer network (RFN-1) and a 3-hidden layer network. The classification performance of the RFN pretrained deep networks was compared to (i) support vector machines, (ii) deep networks pretrained by stacking denoising autoencoders (SDAE), (iii) stacking regular autoencoders (SAE), (iv) restricted Boltzmann machines (RBM), and (v) stacking restricted Boltzmann machines (DBN).

The benchmark datasets are from previous publications [25, 30, 24, 23] and contain: (i) *MNIST* (original MNIST), (ii) *basic* (a smaller subset of MNIST for training), (iii) *bg-rand* (MNIST digits with random noise background), (iv) *bg-img* (MNIST digits with random image background), (v) *rect* (discrimination between tall and wide rectangles), (vi) *rect-img* (discrimination between tall and wide rectangular images overlayed on different background images), (vii) *convex* (discrimination between convex and concave shapes), (viii) *CIFAR-10* (60k 32x32 colour images in 10 classes, with 6k images per class), and (ix) *NORB* (29,160 stereo image pairs of 50 toys belonging to 5 generic categories). The dataset characteristics in terms of size of training, validation and test set is given in the second column of Tab. 1. In all experiments we apply no further data preprocessing except median centering and learn the representation in an unsupervised fashion. We followed Vincent et al. [30] and selected the models based on the validation set performance. The RFNs have as possible hyperparameters: (i) the number of units in the layers from $\{1024, 2048, 4096\}$ and (ii) the dropout rate from $\{0.0, 0.25, 0.5, 0.75\}$. The learning rates were fixed to $\eta_W = 0.1$ and $\eta_\Psi = 0.01$ which we found to be well suited for RFN learning on separate artificial toy datasets, where the task was to find biclusters [20]. For supervised fine-tuning with stochastic gradient descent, we selected the learning rate from $\{0.1, 0.01, 0.001\}$, the masking noise from $\{0.0, 0.25\}$, and the number of layers from $\{1, 3\}$. Again following Vincent et al. [30], fine-tuning was stopped early, where the stopping time was selected based on the validation set performance.

The results of the comparison of deep networks pretrained with RFNs and other models are given in Tab. 1, while Fig. 2 shows some examples of the learnt filters. The test error rate is reported for seven classification problems. A 95% confidence interval is computed according to Vincent et al. [30]. The result of the best performing method is given in bold, as well as the result of those



(a) MNIST digits (b) MNIST digits with random image background

(c) MNIST digits with random noise background (d) convex and concave shapes

(e) tall and wide rectangular (f) rectangular images on background images

(g) CIFAR-10 images (best viewed in color) (h) NORB images

Figure 2: Various randomly selected filters trained on benchmark data sets using a RFN with 1024 hidden units. The panels (a)-(c) show filters for MNIST variations, while (d) and (e)-(f) show filters for convex and rectangle, respectively. Filters learnt from CIFAR-10 and NORB are shown in panel (g)-(h). RFN learnt various kinds of filters, such as stroke, local and global blob detectors. RFN extracts complex features and as in panel (c) can be seen, are the RFN filters unaffected by the background noise.

Table 1: Comparison of deep networks pretrained with RFNs and other models. Test error rate on all considered classification problems is reported together with a 95% confidence interval. The result of the best performing method is given in bold, as well as those for which confidence intervals overlap. The first column gives the data set, the second the size of training, validation and test set, the last column indicates the number of hidden layers of the RFN pretrained deep network which is chosen according to the validation set performance. In only one case RFN pretraining was significantly worse than the best method but still the second best method. In six out of the nine experiments RFN pretraining performed best, where in four cases it was significantly better than all other methods.

Dataset		SVM	RBM	DBN	SAE	SDAE	RFN
MNIST	50k-10k-10k	**1.40**$\pm$**0.23**	**1.21**$\pm$**0.21**	**1.24**$\pm$**0.22**	**1.40**$\pm$**0.23**	**1.28**$\pm$**0.22**	**1.27**$\pm$**0.22** (1)
basic	10k-2k-50k	3.03$\pm$0.15	3.94$\pm$0.17	3.11$\pm$0.15	3.46$\pm$0.16	**2.84**$\pm$**0.15**	**2.66**$\pm$**0.14** (1)
bg-rand	10k-2k-50k	14.58$\pm$0.31	9.80$\pm$0.26	**6.73**$\pm$**0.22**	11.28$\pm$0.28	10.30$\pm$0.27	7.94$\pm$0.24 (3)
bg-img	10k-2k-50k	22.61$\pm$0.37	**16.15**$\pm$**0.32**	**16.31**$\pm$**0.32**	23.00$\pm$0.37	**16.68**$\pm$**0.33**	**15.66**$\pm$**0.32** (1)
rect	1k-0.2k-50k	2.15$\pm$0.13	4.71$\pm$0.19	2.60$\pm$0.14	2.41$\pm$0.13	1.99$\pm$0.12	**0.63**$\pm$**0.06** (1)
rect-img	10k-2k-50k	24.04$\pm$0.37	23.69$\pm$0.37	22.50$\pm$0.37	24.05$\pm$0.37	21.59$\pm$0.36	**20.77**$\pm$**0.36** (1)
convex	10k-2k-50k	19.13$\pm$0.34	19.92$\pm$0.35	18.63$\pm$0.34	18.41$\pm$0.34	19.06$\pm$0.34	**16.41**$\pm$**0.32** (1)
NORB	19k-5k-24k	11.6$\pm$0.40	8.31$\pm$0.35	-	10.10$\pm$0.38	9.50$\pm$0.37	**7.00**$\pm$**0.32** (1)
CIFAR	40k-10k-10k	62.7$\pm$0.95	**40.39**$\pm$**0.96**	43.38$\pm$0.97	43.25$\pm$0.97	-	**41.29**$\pm$**0.95** (1)

methods for which confidence intervals overlap. In only one case RFN pretraining was significantly worse than the best method but still the second best method. In six out of the nine experiments RFN pretraining performed best, where in four cases it was significantly better than all other methods.

5 Conclusion

We have introduced rectified factor networks (RFNs) for pretraining deep networks. RFNs produce sparse and non-linear input representations of the data on different levels and extract factors underlying the data. RFNs are tailored to pretraining of deep networks and to many hidden units. We have shown that RFN learning is a variational EM algorithm with unknown prior and that the fixed point of RFN training explains the data variance like factor analysis.

On nine different benchmark datasets we compare deep networks with RFN pretraining with support vector machines and deep networks pretrained with regular / denoising autoencoders and restricted Boltzmann machines. Only on one dataset RFN pretraining was significantly worse than the best performing method. On six datasets RFNs pretraining outperformed all other methods — in four cases significantly.

RFNs are an interesting alternative to RBMs and autoencoders in the context of deep learning. RFNs have high potential as unsupervised methods for semi-supervised learning and producing sparse codes as they are geared to large datasets and many representational units.

Acknowledgment

The Tesla K40 used for this research was donated by the NVIDIA Corporation.

References

[1] P. Baldi and P. Sadowski. The dropout learning algorithm. *Artificial Intelligence*, 210C:78–122, 2014.

[2] M. J. Beal. *Variational algorithms for approximate Bayesian inference*. PhD thesis, University College London, The Gatsby Computational Neuroscience Unit, 17 Queen Square London WC1N 3AR, 2003.

[3] Y. Bengio, A. Courville, and P. Vincent. Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828, 2013.

[4] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle. Greedy layer-wise training of deep networks. In *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages 153–160. MIT Press, 2007.

[5] D. C. Ciresan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition CVPR 2012*, 2012. Long preprint arXiv:1202.2745v1 [cs.CV].

[6] G. E. Dahl, D. Yu, L. Deng, and A. Acero. Context-dependent pre-trained deep neural networks for large vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):33–42, 2012.

[7] K. Ganchev, J. Graca, J. Gillenwater, and B. Taskar. Posterior regularization for structured latent variable models. *Journal of Machine Learning Research*, 11:2001–2049, 2010.

[8] Z. Ghahramani and G. E. Hinton. Variational learning for switching state-space models. *Neural Computation*, 12:963–996, 1998.

[9] M. Girolami. A variational method for learning sparse and overcomplete representations. *Neural Comput.*, 13(11):2517–2532, 2001.

[10] X. Glorot, A. Bordes, and Y. Bengio. Deep sparse rectifier neural networks. In G. Gordon, D. Dunson, and M. Dudk, editors, *JMLR W&CP: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS 2011)*, volume 15, pages 315–323, 2011.

[11] J. V. Graca, , K. Ganchev, and B. Taskar. Expectation maximization and posterior constraints. In J.C. Platt, D. Koller, Y. Singer, and S.T. Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20, pages 569–576, 2007.

[12] A. Graves, A.-R. Mohamed, and G. E. Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pages 6645–6649. IEEE, 2013.

[13] K. Gregor and Y. LeCun. Learning fast approximations of sparse coding. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 399–406. Omnipress 2010, ISBN 978-1-60558-907-7, 2010.

[14] M. Harva and A. Kaban. A variational bayesian method for rectified factor analysis. In *Proc. Int. Joint Conf. on Neural Networks (IJCNN'05)*, pages 185–190, 2005.

[15] M. Harva and A. Kaban. Variational learning for rectified factor analysis. 2006.

[16] G. E. Hinton and Z. Ghahramani. Generative models for discovering sparse distributed representations. *Philos. T. R. Soc. B*, 352:1177–1190, 1997.

[17] G. E. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[18] G. E. Hinton, N. Srivastava, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580, 2012.

[19] S. Hochreiter. HapFABIA: Identification of very short segments of identity by descent characterized by rare variants in large sequencing data. *Nucleic Acids Res.*, 41(22):e202, 2013.

[20] S. Hochreiter, U. Bodenhofer, M. Heusel, A. Mayr, A. Mitterecker, A. Kasim, S. VanSanden, D. Lin, W. Talloen, L. Bijnens, et al. FABIA: factor analysis for bicluster acquisition. *Bioinformatics*, 26(12):1520–1527, 2010.

[21] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul. An introduction to variational methods for graphical models. *Machine Learning*, 37:183–233, 1999.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton. ImageNet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS 2012)*, page 4, 2012.

[23] Alex Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Master's thesis, Dept. of Comp. Sci., University of Toronto, 2009.

[24] H. Larochelle, D. Erhan, A. Courville, J. Bergstra, and Y. Bengio. An empirical evaluation of deep architectures on problems with many factors of variation. In *Proceedings of the 24th International Conference on Machine Learning*, ICML '07, pages 473–480. ACM, 2007.

[25] Yann LeCun, Fu-Jie Huang, and Leon Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *Proceedings of CVPR'04*. IEEE Press, 2004.

[26] V. Nair and G. E. Hinton. Rectified linear units improve restricted Boltzmann machines. In J. Fürnkranz and T. Joachims, editors, *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814. Omnipress 2010, ISBN 978-1-60558-907-7, 2010.

[27] R. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse, and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. MIT Press, Cambridge, MA, 1998.

[28] B. A. Olshausen and D. J. Field. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–609, 1996.

[29] J. Palmer, D. Wipf, K. Kreutz-Delgado, and B. Rao. Variational EM algorithms for non-Gaussian latent variable models. In *Advances in Neural Information Processing Systems*, volume 18, pages 1059–1066, 2006.

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

[31] D. Warde-Farley, I. J. Goodfellow, A. Courville, and Y. Bengio. An empirical analysis of dropout in piecewise linear networks. *ArXiv e-prints*, 2013.

[32] M. D. Zeiler, M. Ranzato, R. Monga, M. Mao, K. Yang, Q. V. Le, P. Nguyen, A. Senior, V. Vanhoucke, J. Dean, and G. E. Hinton. On rectified linear units for speech processing. In *38th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 3517–3521. IEEE, 2013.