Ladder Networks: Learning under Massive Label Deficit

Behroz Mirza and Tahir Syed
National University of Computer and Emerging Sciences,
Karachi, Pakistan

Jamshed Memon
Barrett Hodgson University,
Karachi, Pakistan

Yameen Malik
Smartlytics
Karachi, Pakistan

***Abstract*—Advancement in deep unsupervised learning are finally bringing machine learning close to natural learning, which happens with as few as one labeled instance. Ladder Networks are the newest deep learning architecture that proposes semi-supervised learning at scale. This work discusses how the ladder network model successfully combines supervised and unsupervised learning taking it beyond the pre-training realm. The model learns from the structure, rather than the labels alone transforming it from a label learner to a structural observer. We extend the previously-reported results by lowering the number of labels, and report an error of 1.27 on 40 labels only, on the MNIST dataset that in a fully supervised setting, uses 60000 labeled training instances.**

***Keywords*—Ladder networks; semi-supervised learning; deep learning; structure observer**

I. Introduction

Over the past decades, there has been an effort in machine learning theoretical research to move from supervised to unsupervised methods, for the reasons of 1. arduous effort in labeling data, and 2. the inherent aptitude of unsupervised approaches to discover the latent structure of data without the guiding (or misguiding) external influence of labels.

The paper discusses the opportunities and strengths in deep unsupervised learning and its implications towards unsupervised and weekly supervised learning in general. The model selected for this purpose is the recently-introduced *ladder network* designed by Valpola [1]. This work modifies the model configuration and reports an error on the extremely popular MNIST benchmark of 1.27 using 40 labels only. This is 10 labels fewer than previously reported results.

Ladder networks successfully combine supervised learning with unsupervised learning in deep neural networks models. Prior to this unsupervised learning was used for specialized pre-training task, followed by supervised learning. However ladder networks, are trained to simultaneously minimize the sum of supervised and unsupervised cost functions using backpropagation, thus eliminating the need for layer-wise pre-training.

The model has the distinctive feature of learning from the structure in the data instead of solely from the labels alone. This novelty results in minimizing the amount of labelled data required for training the network. As most of the data are unlabeled, the model learns principal features from the small set of labelled data and correlated features from the large set of unlabeled data concurrently [2]. This makes the machine learning process narrowly closer to natural learning.

The rest of the paper is organized as follows. Section II - 'Deep Unsupervised Learning' discusses the models namely RBM and Auto encoders. Section III - 'Semi Supervised Learning' discusses the Ladder Networks model followed by the experiments and results. Section IV - 'Conclusion' concludes the paper and discusses future research directions.

II. Deep Unsupervised Learning

A. Relaxing Supervision

Unsupervised learning forms a class of machine learning techniques of deducing a function to disentangle hidden structure from unlabeled data. What clearly distinguishes unsupervised learning from supervised learning is unlabeled samples are used during training so there is no error or reward signal to evaluate a potential solution. As unsupervised learning attempts to draw inferences from datasets consisting of input data without labeled responses it is closely related to the problem of density estimation in statistics [3].

Hinton and Salakudinov [4] proposed the idea of the stochastic RBM; symmetrical arrangement of binary stochastic neurons in a Boltzmann Machine where the two layers of the model for a bipartite graph. Later works by [5] suggested Auto Encoders for pre training; pre train each successive layers using unsupervised measure thus producing an enriched useful higher-level representation from the lower-level representation output. State of the art generalization can later be achieved by running Gradient descent on supervised format. Transitioning probability to unsupervised learning looks promising based on the fact - natural learning is unsupervised; we learn the structure around us by observing not by the names of the associated objects.

B. Greedy Unsupervised Pre-training

The year 2006 marks the breakthrough in training deep architectures as RBM were proposed followed by stacked autoencoders (SAEs) (Fig. 1). Both approaches used the notion of Greedy layer-wise unsupervised pre-training followed by supervised fine-tuning. The concepts Greedy layer wise pre training and Supervised fine tuning have profound impacts on Unsupervised Learning. Unsupervised pre-training leads to

- Pre-conditioning the model, whereby arranging the parameter values in suitable ranges later to be used in

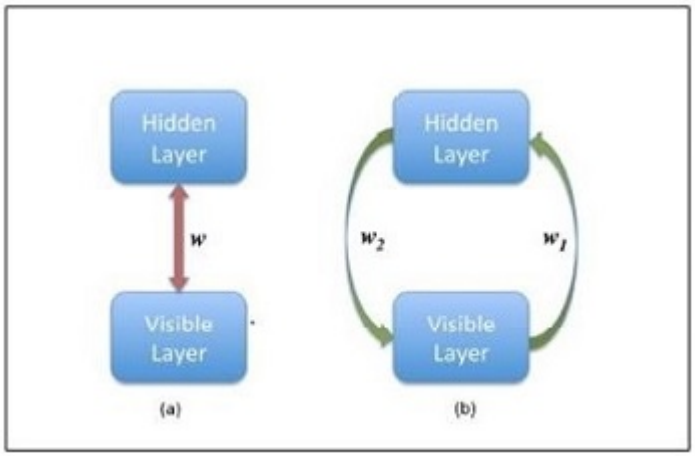

Fig. 1. RBM and Autoencoder architectures [6].

supervised training. This is completely different than a generic random initialization.

- Model initialization to an already close to optimal configuration in parameter space leads to an optimization of the optimization process. The appropriate range increases backpropagation efficiency and error minimization.

Greedy layer-wise unsupervised pre-training introduces a useful prior to the supervised fine-tuning training [7].

- Unsupervised learning is used to draw inferences from datasets consisting of input data without labeled responses, this makes it different from the supervised counterpart.
- Only use the inputs $x(t)$ for learning.
- Auto extracting meaningful features from data. This distinctive feature extraction capability without utilization of labels, makes unsupervised learning a prime candidate simulating near to human learning behavior.
- Leverage the availability of unlabeled data. Most data being unlabeled and natural intelligence deals with analyzing the bulk of objects via structures rather than labels, therefore unsupervised learning can utilize this massive data in training deep models exhibiting the distinction of learning via structures and not simply labels.

Two popular neural models for unsupervised learning are:

1) Restricted Boltzmann Machines
2) Autoencoders

C. Restriced Boltzmann Machines

Restricted Boltzmann Machines are autoencoder models that have the distinctive capability of transforming and reducing high dimensional data to low dimension. These use an effective way to initialize the weights of the model with calibrated values followed by gradient descent for fine tuning.

This model uses the sigmoid non-linearity, thus be called nonlinear generalization of the PCA. The model has outperformed the Linear PCA producing outstanding results, [8].

Trials confirm that weight optimization is challenging in Deep Non Linear Auto Encoders. Initialization with Large weight values leads to poor local minima problem, while initializing with small weights leads to very small gradients. However if an Auto Encoder initialized is with good calibrated weights, Gradient Descent performs well, but this initialization requires a novel algorithm which learns one layer of feature at a time. Each layer captures strong, high-order correlations b/w activities of units in layer below it.

As examples of unsupervised learning, RBM are used for pre-training phase extensively. The phase consists of learning a stacked RBM each having only one layer of feature detectors. The learned feature activations of one RBM are used as data for training the next RBM in the stack. After pre training these RBM are unfolded to create a deep auto encoder, which is later fine-tuned using back propagation.

D. Free Energy

The RBM model defines a distribution over x with latent variables via an energy function E. The function gives the probability distribution $P(v, h)$ where:

$$E(v,h) = -\sum_{iepixels} b_i v_i - \sum_{jefeatures} b_j v_j - \sum_{i,j} v_i h_j w_i j \tag{1}$$

If w is negative it leads to high energy and the probability decreases, if w is positive it leads to low energy and the probability increases. The challenge here: the function is divided by the partition function Z, the sum over all values of v and h. As v, h are binary, so Z can take many values leading to an exponential sum over the numerator, thus making computing it intractable. To overcome this challenge Hinton et al. proposed *contrastive divergence*.

E. Contrastive Divergence: The Negative Sample

Contrastive divergence [9] was proposed by Hinton. It uses Gibbs sampling to approximate joint distribution when direct sampling is difficult. Alternating between layers, given one unit in visible layer, all units are independent in hidden layer, values in one layer be sampled given a value in another layer (Fig. 2). Using contrastive Divergence we get the following interesting relations between Energy and Probability.

- Increase probability of observing xt at hidden layer, decrease the energy
- Decrease probability of observing x t at hidden layer increase the energy.
- Increase probability of observing digits from training set.
- Decrease probability of observing noise.
- Decreasing energy of things that looks like what is in training set.

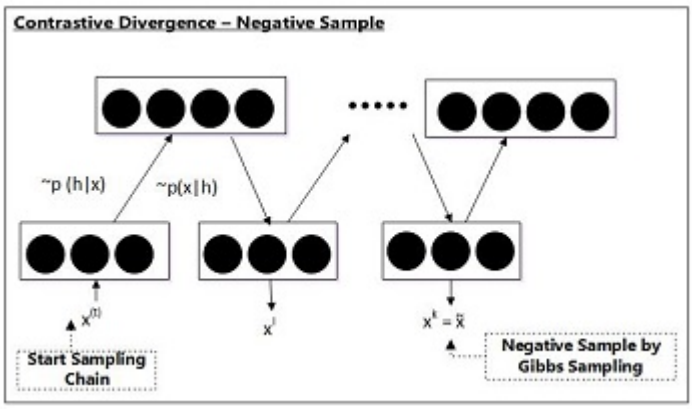

Fig. 2. Negative sample.

- Increase energy of things that are hallucinated or sampled by the model
- Ultimately model spits out things similar to the what in the model i.e. after a series of cycles x becomes closer to x t

F. Auto Encoders

An autoencoder model [10] attempts to regenerate its input. It has a hidden layer that describes a code used to represent the input. The model network comprises of an encoder function $h = f(x)$ and a decoder that produces a reconstruction $r = g(h)$. Auto encoders are designed to be unable to learn to copy perfectly. As the model is forced to prioritize which aspects of the input should be copied, it often learns useful properties of the data [11].

As discussed above the model has two parts namely the encoder and the decoder. Encoder takes an input and encodes it in a linear representation. Encoder is sigmoid of a linear transformation. Decoder takes latent representation h(x), passes it to non-linearity, generating the output. As the Auto encoder attempts to minimize the reconstruction error between actual and generated value during training, a typical Loss function is:

$$L(x, x') = \|x - x'\|^2 = \|x - \sigma 2(W'(\sigma 1(Wx + b) + b'))\|^2 \quad (2)$$

For deep autoencoders, their representational power, layer size and depth can be elaborated as:

- Universal Approximation Theorem [12] says feed forward network with a linear output layer and one hidden layer with non-linearity based squashing function can approximates any function. However the hidden layer may be large leading to generalization failure.
- Reduction in number of units can happen in deep models to represent the desired function get a better and generalization error.
- Exponential Reduction in Computational cost of representing a function and training data needed to learn can be done by increasing the depth of the model.

TABLE I. RESTRICTED BOLTZMANN MACHINE AND AUTO ENCODERS

RBM	Auto encoder
Stochastic model setting with symmetric connectivity b/w the visible and hidden layers.	Deterministic model with two weight matrices w1 and w2 representing the flow of data from the visible-to-hidden and hidden-to-visible layers.
Energy based models.	Achieved considerable success via de-noising the input.
Trained using contrastive divergence that performs Gibbs sampling and is used inside a gradient descent procedure.	Trained to perform optimal reconstruction of the visible layer by minimizing the mean-squared error in a reconstruction task.
Preferred in High Noise Scenarios & Speech Recognition.	Preferred in Low Noise Scenarios.

- Deep auto encoders yield better compression than shallow or linear auto encoders [13].

G. RBM and Auto Encoders

Restricted Boltzmann Machines and Auto encoder models; both are used for training deep architectures using an unsupervised greedy layer wise pre training followed by supervised fine tuning. Table I compares these two models.

III. SEMI-SUPERVISED LEARNING

Semi-supervised learning a class of machine learning techniques that has the capability of utilizing small volume of labeled data with a large volume of unlabeled data; leading to substantial improvement in learning precision. Thus it therefore falls between unsupervised learning and supervised learning realm. As the acquisition and labelling cost of labeled data for a specific learning problem is high making the dataset set infeasible, whereas acquisition of unlabeled data is relatively inexpensive. These settings make semi-supervised learning of great technical and practical value.

It is worth noting that supervised learning has achieved good results as opposed to unsupervised learning. The reason being that implementations of unsupervised learning are not are incompatible with supervised learning. Supervised Learning processes filter out non relevant information preserving only the important features where as unsupervised learning methods try retain and represent as much information about the original data as possible. This is where semi-supervised learning comes into play, integrating both supervised and unsupervised learning together in a novel architecture - The Ladder Networks.

A. Ladder Network An Autoencoder with shortcut connections

The Ladder Network model is an autoencoder with adjacent shortcut connections from the encoder to decoder at each layer. These connections let the higher layers to focus on abstract invariant and consistent features. Comparatively standard auto encoders are equivalent to latent variable models with a single layer of stochastic variables only; however the ladder network is equivalent in strength to hierarchically ranked latent variables models [14].

Ladder networks combine supervised learning with unsupervised learning in deep neural networks. As stated before, in

classical setting, unsupervised learning was used only for pre-training the network this was followed by supervised learning. However ladder networks integrates the two together. Similar to feedforward networks, learning occurs via minimizing the relevant cost function. Another important aspect is higher layers can focus on consistent features only leaving the details for the lower layers to represent. Classically unsupervised learning has been used for pre-training prior to supervised learning. However here, it continues to work after Supervised Learning has commenced. Relevant features are selected via supervised learning using labelled samples while unsupervised learning selects correlated features using bulk of unlabeled data. This improves generalization to new samples.

It is important to note that in this integrated model, once supervised learning starts selecting significant features, unsupervised learning only focuses on and selects co-related features which are useful for supervised learning. This characteristic is completely in contrast with the classic pre-training approach where unsupervised learning selected all the features via which input could be re generated.

B. Stochastic Latent Variable Models and Deterministic Autoencoders

Unsupervised learning methods where latent variables generate observed data are called Latent variable models. These have the capability of forecasting the mean of the observations. Minimizing the mismatch between the observed data and its reconstruction results in inference of the unknown latent variables.

Single layer latent models do not discard information and represent everything about the data; discarding information in a single layer model would increase the reconstruction error. For better reconstruction error management, subsidiary information along with the abstract consistent features is also required. The solution being using latent variables in a multi-layer hierarchy. This leads to higher layer levels to focus on abstract consistent features and leaving the details to lower levels. Finally higher-level latent variable models can easily represent the mean of the lower-level variables. This hierarchical arrangement provides stochasticity to latent variables but with a forewarning. Computing the posterior probability of the latent variables and their parameters is mathematically inflexible. Another problem with these models is the extensive usage of variational Bayesian probabilistic methods or training leading to a significant compromise in their performance.

Autoencoders are equivalent in representational power to single-layer latent variable models. Learning is based on minimizing the difference between the observation and its reconstruction. Similar to latent variable models, auto encoders have the capability of being stacked together. Training follows a layer wise approach as new layers are added to the previously trained network. After this, training later continues in a supervised manner, [15]. An important point to be noted is a hierarchical/stacked auto encoder is not equivalent in representational power to the hierarchical latent variable model counterpart. The difference being that the middle layers in an auto encoder are strongly deterministic while the hierarchical latent variable model has complete stochasticity. To elaborate more, regardless of whatever the priors are, stochastic variables have independent representational aptitude and thus can add information at the time of reconstruction. In stark contrast, deterministic variables cannot add any information at reconstruction time [16].

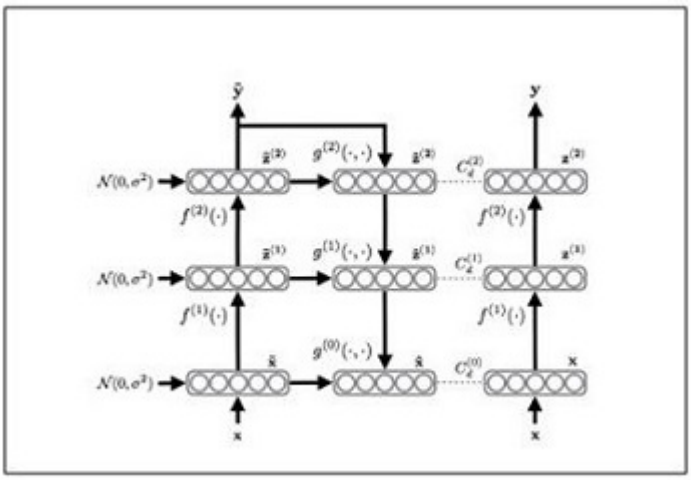

Fig. 3. A two layer ladder network [15].

This clearly shows that both autoencoders and latent variable models have their pros and cons. This paves way for a more robust model that has the capability of combining the strengths of the two models making it an ideal candidate for deep semi-supervised learning.

C. The Ladder model

The ladder network attempts to combine best of the breed from both Latent Variable Models and Stacked Auto encoder models. As is evident from the equations, $h(t)$ in an auto encoder depends only on top-down information and cannot add any new information to the representation because it does not receive information from the bottom-up path. On the contrary inference of $s(l)(t)$ in the latent model combines information from top-down priors and bottom-up likelihood.

The Hierarchical Latent Variable Model can therefore be expressed as:

$$p(s^l(t)|s^{l+1}(t), \xi^l) \tag{3}$$

The Stacked Auto encoder can be expressed as:

$$h^{l-1}(t) = g^l(h'^l(t)) \tag{4}$$

Likewise the Ladder Network can be expressed as:

$$h^{l-1}(t) = g^l(h'^l(t)), h^{(l-1)}(t)) \tag{5}$$

Transformation of an Autoencoder into a Latent Model when shortcut adjacent connections are added from the bottom-up encoder path to the modified top-down decoder path. This empowers the hidden layer to recover information which is missing from the higher layers, thus relieving higher layers to represent all the details. The mapping function g in the Ladder Model combines abstract information from higher layers with detailed information from lower layers (Fig. 3).

D. Distributed Layer-wise Learning

Deep networks have error functions only at the output layer. Keeping this in mind if similar training procedure is adopted for Ladder Networks as with Auto encoders; this leads to a considerable increase in the reconstruction error as the lateral connections also contributes to the reconstruction, leaving minimum error contribution from the higher layers which are the custodians of most consistent information. Auto encoder model has the inherent capability of routing all the signals through all layers and making learning difficult and slow. However latent variable models have cost functions for all stochastic variables. As the ladder network combines its structure from both these models, it introduces training signals at each level of the hierarchy exhibiting the novel feature of distributed learning.

The cost Function of the ladder network can be expressed as:

$$C = \frac{1}{T}\sum_{t=1}^{T} \|s(t) - s'(t)\|^2 = \frac{1}{T}\sum_{t=1}^{T} \left\|f^1 x(t) - s'(t)\right\|^2 \quad (6)$$

$$C^l = \frac{1}{T}\sum_{t=1}^{T} \left\|h^l(t) - h'^l(t)\right\|^2 \quad (7)$$

The first cost function of the ladder network expressed as (6) does not directly refer to the input but only to the latent variable and its corrupted counterpart. This leads to the fact that each layer in the hierarchical model contributes to the cost function, bringing the basis of training signals close to the parameters on the relevant layer. The 2nd function expressed as (7) shows that error term can be used at each layer, thus leading to Layered Learning novelty. Layer wise denoising is also used in the Model. The idea taken from [13] corrupting the input of auto encoders with noise and let the network attempt to reconstruct the original uncorrupted inputs. This forces the auto encoder to learn how to denoise the corrupted inputs. Bengio stated denoising not only the inputs but on all levels of the encoder path in a hierarchical model results in efficient sampling; and called such networks generative stochastic networks (GSN) [17]. Another important aspect is that a model with reconstruction capability of missing data can be turned into a probability density estimator.

E. Implementation

A typical implementation of the ladder network include:

1) A feedforward model which serves supervised learning with 2 encoder - clean and corrupted. The corrupted encoder adds Gaussian noise at all layers.
2) A decoder which has the capability to invert the mappings on each encoder layer and provisions unsupervised learning. Decoder uses a denoising function to reconstruct the activations of each layer. The target at each layer is the clean version of the activation where as the difference between the reconstruction and the clean version serves as the denoising cost of that layer.
3) The supervised cost is calculated from the output of the corrupted encoder and the output target. The unsupervised cost is the sum of denoising cost of all layers scaled by a hyper parameter. The final cost is the sum of supervised and unsupervised cost.
4) Train the whole network in a semi-supervised arrangement using standard optimization techniques.

F. Experiments and Results

TABLE II. ERROR REPORTED WITH 40 LABELS ON (SET 1)

	Initial Learning rate = 0.002 Annealed linearly to zero Decay Rate = 0.67 Layers 784-1000-500-250-250-250-10 Epochs = 150 / Dataset = MNIST			
Models/ No of Labels	40	100	1000	All
Full Model	1.27	1.07	0.71	0.6
Sigma (Top) Model	3.9	3.4	1.9	0.9
Bottom Model	1.30	1.09	1.0	0.71

TABLE III. DENOISING VALUES FOR 6 LAYERS (SET 1)

	Denoising Cost configured on Layer wise basis		
Models/ No of Labels	100	1000	All
Full Model	1000,10,0.1, 0.1,0.1,0.1, 0.1	2000,20,0.1, 0.1,0.1,0.1, 0.1	1000,1,0.01, 0.01,0.01,0.01, 0.01
Sigma (Top) Model	0,0,0,0,0,0,0.5	0,0,0,0,0,0,10	0,0,0,0,0,0,2
Bottom Model	5000,0,0,0,0, 0,0	2000,0,0,0,0, 0,0	2000,0,0,0,0, 0,0

TABLE IV. ERROR REPORTED WITH 100 LABELS (SET 2)

	Initial Learning rate = 0.002, Annealed linearly to zero Decay Rate = 0.67 Layers 784-1000-500-250-10 Epochs = 150 / Dataset = MNIST		
Models/ No of Labels	100	1000	All
Full Model	1.22	0.852	0.72
Sigma (Top) Model	4.08	1.28	1.08
Bottom Model	1.308	1.23	0.852

The Tables II, III, IV summarize the results of the experiments conducted in this research. Table II shows the results obtained using a 6 layers model while Table IV shows the results obtained using a 4 layers model. The Table III shows the layer wise denoising values used. These results are further elaborated in the Inferences section below.

G. Inferences

1) Table II shows, with 60000 labels the error rate is around 0.6, while with 1000 labels its 0.71, with 100 labels it is 1.06 and with merely 40 labels its 1.27.
2) The Top Model in Table II has also impressive results of 3.4 with 100 labels. It has cost function at only top layer, thus most of its denoising part can be bypassed. It has the capability of being plugged in any neural network.
3) Reducing the number of layers from 6 (Table II) to 4 (Table IV), increases the error by around 21
4) Top and bottom models have denoising cost at the top and bottom layer only as shown in Table III.
5) Applying noise to each layer and specifically to the first layer leads to regularization thus minimizing the generalization error as shown in Table III.

IV. Conclusion

This exploratory research highlights the impact ladder networks has brought upon deep unsupervised learning by alleviating pre-training and successfully synthesizing the two forms of learning which are usually considered under exclusivity. As for semi-supervised learning, it can be concluded that the most important contribution is made by the lateral adjacent connections. These connections being a vital component to the level that removing them declines the performance for all of the semi supervised tasks. The second important contribution is introduction of noise at each layer. As the number of labeled examples increases, the adjacent connections and the re construction criterion become less significant and the generalization enhancement coming from the induction of noise in each layer and vice versa [18]. Ladder networks have transformed the neural network model from a label learner to a structural observer, thus narrowing the gap between machine learning and machine intelligence.

In our opinion, the ladder network could be applied in these future directions: data generated in high velocity environments are unlabeled, which makes training models difficult; we envisage to use the Ladder model exhibiting-semi supervised learning in these settings. Semi-supervised models require large training time, optimizing it is a potential challenge. Ladder Networks have shown promising results with low dimensional data sets, so their use on high dimensional data sets is a possibility for exploration.

References

[1] Harri Valpola. From neural PCA to deep unsupervised learning. In Adv. in Independent Component Analysis and Learning Machines, pages 143171, Elsevier. arXiv:1411.7783, 2015

[2] Antti Rasmus, Tapani Raiko, and Harri Valpola. Denoising auto encoder with modulated lateral connections learns invariant representations of natural images. arXiv:1412.7210, 2015

[3] Krizhevsky, A., I Sutskever, and G. E Hinton. ImageNet classification with deep convolutional neural networks. In: Advances in Neural Information Processing Systems, pp. 11061114 , NIPS, 2012

[4] Hinton, G. and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. In: Science 313.5786, pp. 504507, 2006

[5] Pascal Vincent, Hugo Larochelle, Yoshua Bengio. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. In Journal of Machine Learning Research, 2010

[6] Temporal Auto encoding Restricted Boltzmann Machine. Chris Hausler and Alex Susemihl. arXiv:1210.8353v1, 2012

[7] Ian Goodfellow, Yoshua Bengio and Aaron Courville. The Deep Learning textbook, An MIT Press book. Under Publication, 2016

[8] Salakhutdinov, R. , Mnih, A. , Hinton, G. Restricted Boltzmann machines for collaborative filtering. Proceedings of the 24th international conference on Machine learning - ICML 07. p. 791. ISBN 9781595937933,doi:10.1145/1273496.1273596, 2007

[9] Miguel A. Carreira-Perpinan and Geoffrey Hinton. On contrastive divergence learning. Artificial Intelligence and Statistics, 2005

[10] Bengio, Y. Learning Deep Architectures for AI. Foundations and Trends in Machine Learning. 2. doi:10.1561/2200000006, 2009

[11] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In Advances in Neural Information Processing Systems 26, pages 899 907, 2013

[12] Cybenko., G. Approximations by superpositions of sigmoidal functions, Mathematics of Control, Signals, and Systems, 2 (4), 303-314, 1989

[13] Salah Rifai, Pascal Vincent, Xavier Muller. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In Proceedings of the Twenty-eight International Conference on Machine Learning, 2011

[14] Oja, E. The nonlinear PCA learning rule in independent component analysis. In: Neurocomputing 17.1, pp. 2546, 1997

[15] Antti Rasmus, Harri Valpola, Mikko Honkala, Mathias Berglund, and Tapani Raiko. Semi-supervised learning with ladder networks. arXiv preprint arXiv:1507.02672, 2015

[16] Ilin, A. and H. Valpola. On the effect of the form of the posterior approximation in variational learning of ICA models. In: Proc. of the 4th Int. Symp. on Independent Component Analysis and Blind Signal Separation (ICA2003). Nara, Japan, pp. 915920, 2003

[17] Yoshua Bengio, Eric Thibodeau-Laufer, Guillaume Alain, Jason Yosinski. Deep Generative Stochastic Networks Trainable by Backprop. arXiv:1306.1091, 2013

[18] Mohammad Pezeshki, Philemon Brakel, Aaron Courville, Yoshua Bengio. Deconstructing The Ladder Network Architecture. Under review as a conference paper at ICLR, 2016