Layerwise Pre-training with Autoencoders

Vanessa Jurtz

June 17, 2014

CENTERFORBIOLOGICALSEQUENCEANALYSIS CBS

Motivation

CENTERFO RBIOLOGI CALSEQU ENCEANA LYSIS **CBS**

Journal of Machine Learning Research 1 (2009) 1-40 Submitted 12/07; Revised 9/08; Published 1/09

Exploring Strategies for Training Deep Neural Networks

Hugo Larochelle LAROCHEH@IRO.UMONTREAL.CA
Yoshua Bengio BENGIOY@IRO.UMONTREAL.CA
Jérôme Louradour LOURADOJ@IRO.UMONTREAL.CA
Pascal Lamblin LAMBLINP@IRO.UMONTREAL.CA
Département d'informatique et de recherche opérationnelle
Université de Montréal
2920, chemin de la Tour
Montréal, Québec, Canada, H3T 1J8

Extracting and Composing Robust Features with Denoising Autoencoders

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, Pierre-Antoine Manzagol
Dept. IRO, Université de Montréal
C.P. 6128, Montreal, Qc, H3C 3J7, Canada
http://www.iro.umontreal.ca/~lisa

Technical Report 1316, February 2008

Contents

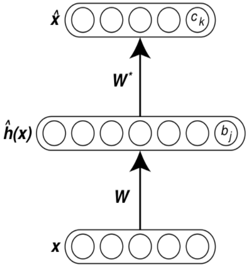
What is an autoencoder?

- Introduction
- Possible problems with autoencoders
- Tied Weights
- Denoising autoencoders

Some examples of deep learning

- Peptide-MHC training
- MNIST dataset training

What is an autoencoder?

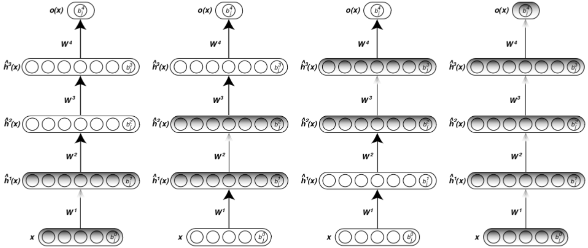


Neural network that is trained to reproduce its input in the output layer

Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research **2009**

What is an autoencoder?

Greedy unsupervised, layer wise pretraining ⇒ stack the autoencoders to initialize weights in deep net



(a) First hidden layer pre-training (b) Second hidden layer pre-training (c) Third hidden layer pre-training (d) Fine-tuning of whole network

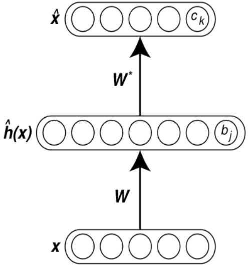


Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research **2009**

What is an autoencoder?

Idea: the layers of deep nets should separate the factors of variation, they should be high-level feature representations of the input

$\Rightarrow$ pre-training with autoencoders should initialize the weights closer to good solutions.

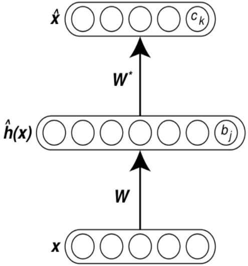


Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research **2009**

Possible problems with autoencoders

They can learn an uninteresting identity function!

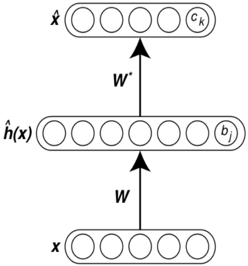
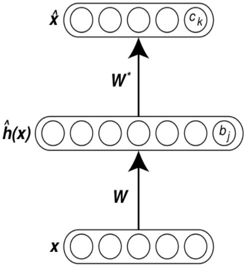


Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research **2009**

Tied weights

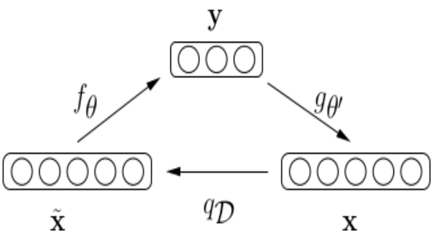


Avoid learning the identity function for continuous input by setting: $W^T = W^*$

Motivation: W^T and W^* tend to be similar after training

Figure adapted from Larochelle et al. *Exploring strategies for training Deep Neural Networks* Journal of Machine Learning Research **2009**

Denoising autoencoders



- corrupt part of the input of the autoencoder by setting values to 0
- this simulates removal of the neurons
- train the autoencoder to reproduce the original input from the corrupted version

Figure adapted from Vincent et al. *Extracting and Composing robust features with Denaoising Autoencoders* Technical Report 1316 **2008**

Some examples of deep learning

CENTERFO
RBIOLOGI
CALSEQU
ENCEANA
LYSIS **CBS**

Peptide MHC-ClassI binding:

- Dataset and protein sequence encoding
- Different training strategies
- Noise levels in DA
- Cost functions
- Pretraining rounds

MNIST dataset

- Introduction to the dataset
- Examples of training

MHC Class I binding data

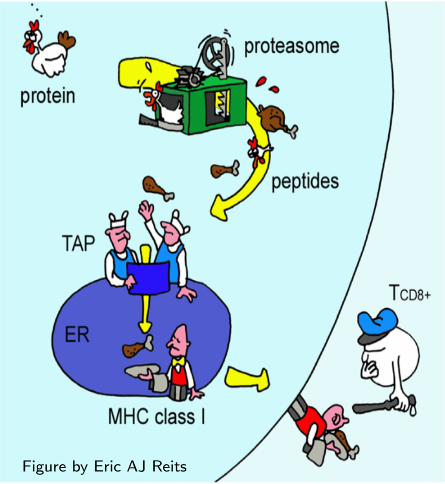


Figure by Eric AJ Reits

ARWLASTPL	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.589395
GMMGGLWKY	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.439136
KQWSWFSLL	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.451477
MMMPMFNAF	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.666267
MMYASWGVH	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.682619
MRMAWGGSY	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.344174
RMGAAVTPY	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.634813
RMMETWHPL	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	1.000000
RMMGKTNPL	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	1.000000
RRMATTFTF	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.474391
WMYNIQPYL	YYSEYREISENVYESNLYIAYSDYTWEYLNYRWY	0.935937
AAKKKGASL	YHTTYREISENWYEANLYLEYEYYSMAAFNYTWY	0.728109
ALLNIKVKL	YHTTYREISENWYEANLYLEYEYYSMAAFNYTWY	0.453792
FLRRRRAAL	YHTTYREISENWYEANLYLEYEYYSMAAFNYTWY	0.797442

IC 50 binding values are log transformed using $1 - \frac{log(affinity)}{log(50000)}$

Protein sequence encoding

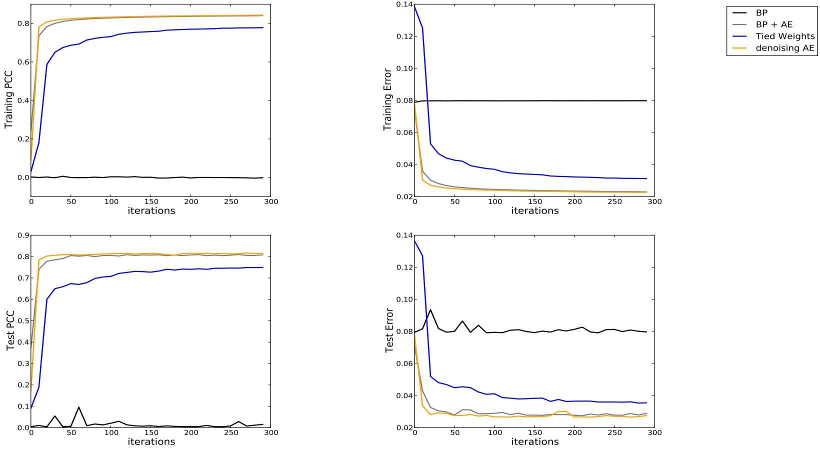
Each amino acid is encoded by a sequence of 20 numbers. One of them is 0.9 and the position of 0.9 indicates which animo acid is encoded. The other numbers are all 0.05.

Example:

A $\Rightarrow$ 0.9 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05 0.05

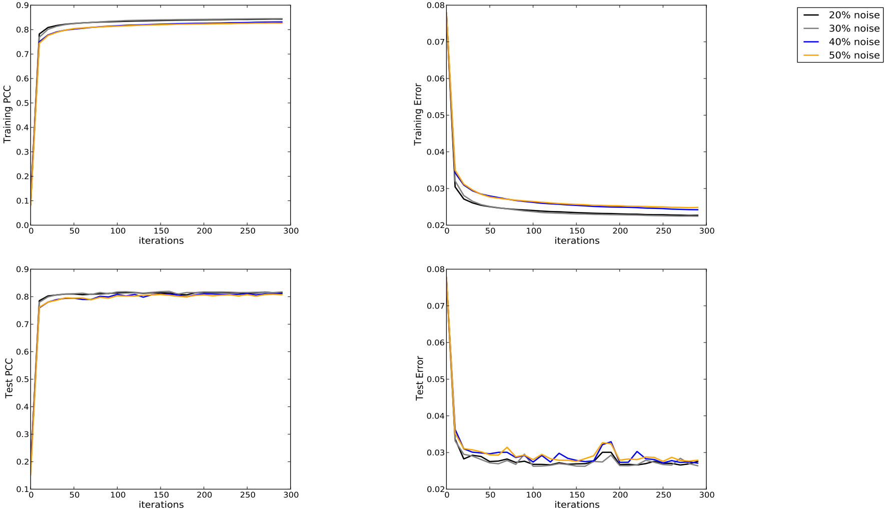
Peptide training

Peptide and receptor sequence, sparse encoding, 4 hidden layers 20 neurons in each



Peptide training - different noise levels

Peptide + receptor sequence, sparse encoding, 4 hidden layers 20 neurons in each



MSE cost function:

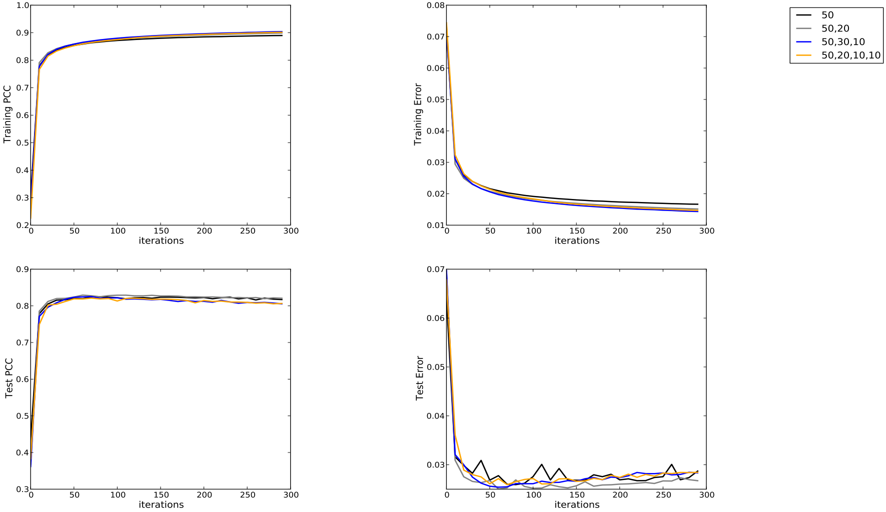
$$E = \sum_k (t_k - O_k)^2 \Rightarrow \delta_{k_O} = (O_k - t_k) \cdot O_k(1 - O_k)$$

CE cost function:

$$E = -\sum_k [t_k ln(O_k) + (1 - t_k) ln(1 - O_k)] \Rightarrow \delta_{k_O} = (O_k - t_k)$$

Peptide - MHC training

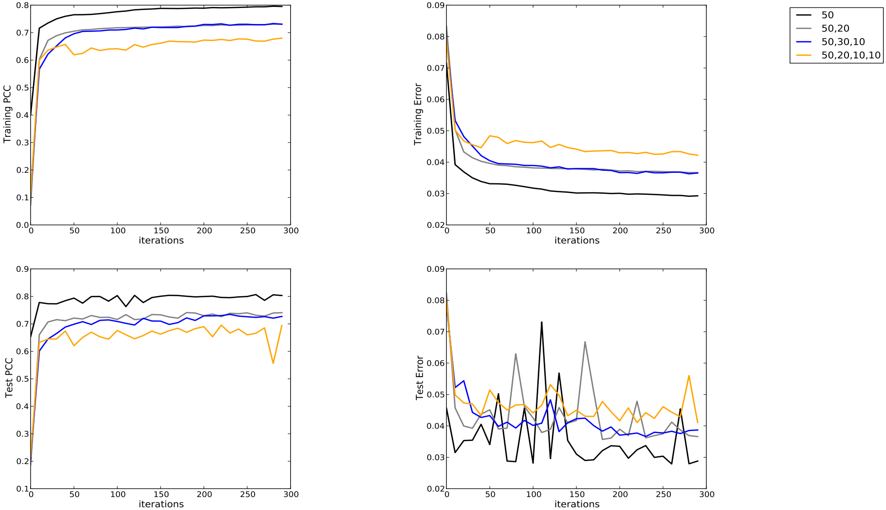
Denoising Autoencoder MSE, sparse encoded peptide + receptor



Peptide - MHC training

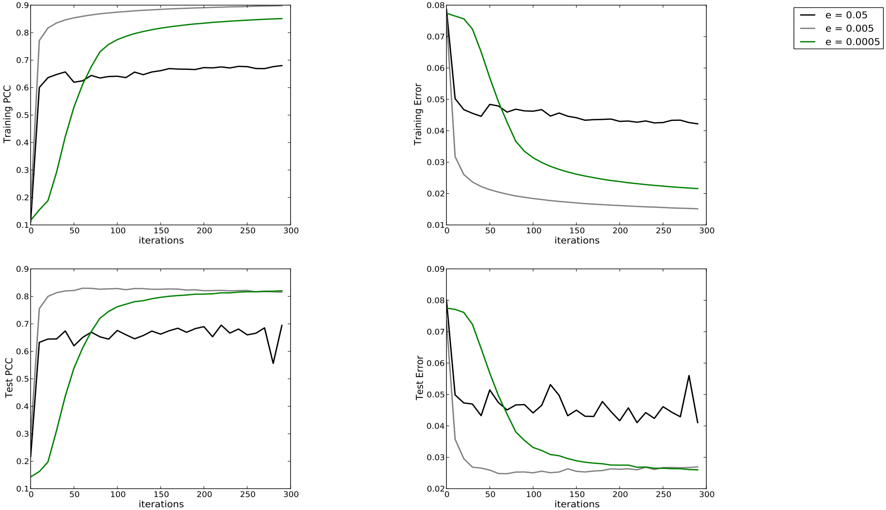
CENTERFORBIOLOGICALSEQUENCEANALYSIS CBS

Denoising Autoencoder CE, sparse encoded peptide + receptor



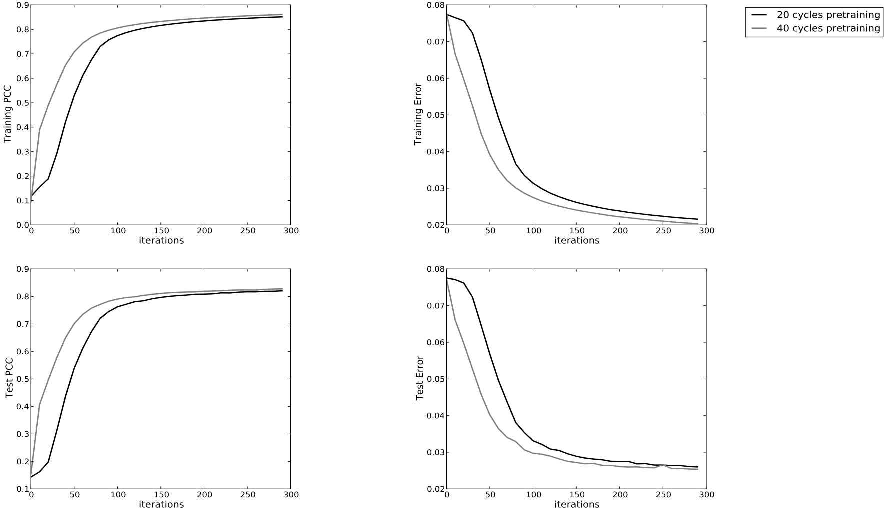
Peptide - MHC training

Denoising autoencoder CE, sparse encoded peptide + receptor 50,20,10,10



Peptide - MHC training

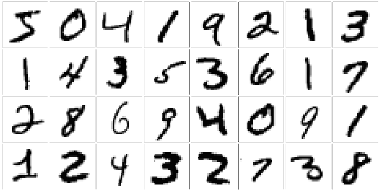
DA CE, sparse encoded peptide + receptor 50,20,10,10 e=0.0005



Conclusions:

- Training deep networks with autoencoder pre-training is possible
- Denoising autoencoders work better than tied weights for sparse encoded peptid-receptor data
- Training parameters need to be further optimized

MNIST Dataset



Handwritten digits (written by different people)

- 10 000 training examples
- 5 000 validation examples (for early stopping)
- 50 000 test examples

MNIST Dataset training

Network Type	Depth	MNIST-small classif. test error	MNIST-rotation classif. test error
Neural network (random initialization, + fine-tuning)	1	**4.14** % ± 0.17	15.22 % ± 0.31
	2	**4.03** % ± 0.17	**10.63** % ± 0.27
	3	**4.24** % ± 0.18	11.98 % ± 0.28
	4	4.47 % ± 0.18	11.73 % ± 0.29
SAA network (autoassociator learning + fine-tuning)	1	3.87 % ± 0.17	11.43% ± 0.28
	2	**3.38** % ± 0.16	9.88 % ± 0.26
	3	**3.37** % ± 0.16	**9.22** % ± 0.25
	4	**3.39** % ± 0.16	**9.20** % ± 0.25
SRBM network (CD-1 learning + fine-tuning)	1	3.17 % ± 0.15	10.47 % ± 0.27
	2	**2.74** % ± 0.14	9.54 % ± 0.26
	3	**2.71** % ± 0.14	**8.80** % ± 0.25
	4	**2.72** % ± 0.14	**8.83** % ± 0.24

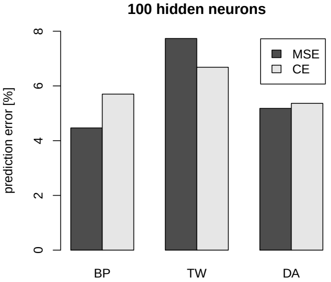
Table 3: Classification performance on MNIST-small and MNIST-rotation of different networks for different strategies to initialize parameters, and different depths (number of layers).

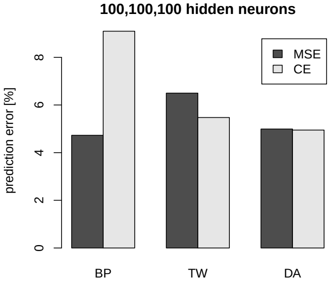
Network	Depth	Test performance
Backpropagation + MSE	1	4.332
Backpropagation + CE	1	4.204
Tied weights + MSE	1	7.428
Tied weights + CE	1	4.588
Denoising Autoencoder + MSE	1	8.41
Denoising Autoencoder + CE	1	4.936

We used 500 hidden neurons in each hidden layer, a learning rate of 0.005 and 20 rounds of pre-training.

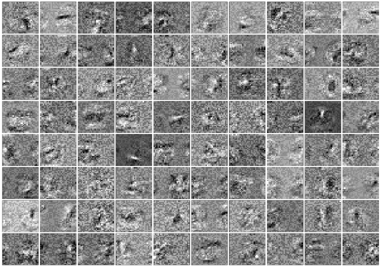
MNIST Dataset training

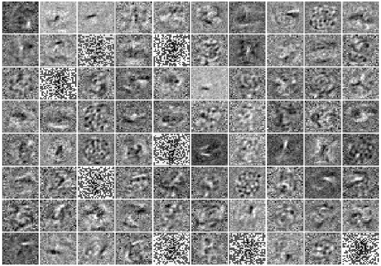
- 0s in MNIST data replaced by 0.1
- learning rate: 0.005
- pretraining: 100 rounds + early stopping
- fine-tuning: 300 rounds + early stopping





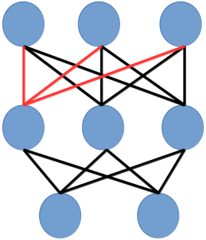
MNIST Dataset training





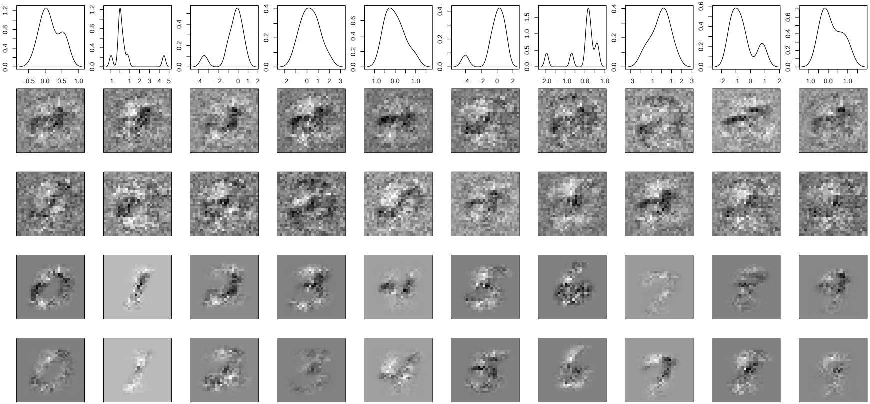
Weight images
all weights in 1st hidden layer
leading to one hidden neuron

top: BP 1 hidden layer 100 hidden neurons
bottom: DA 1 hidden layer 100 hidden neurons



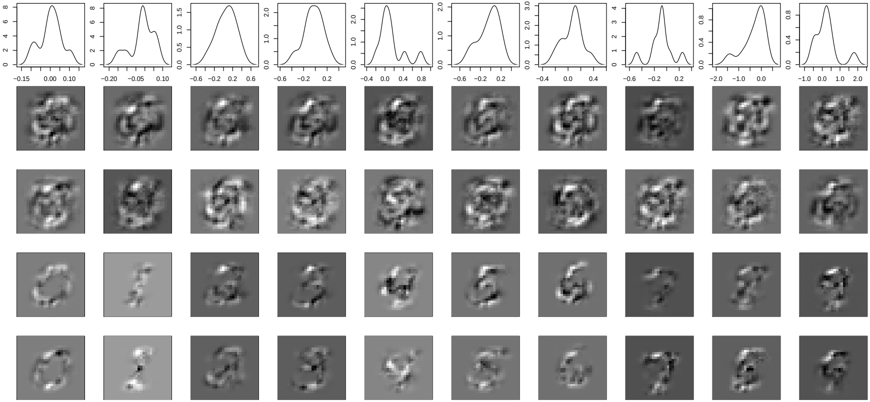
MNIST Dataset training

Backpropagation, 1 hidden layer with 100 hidden neurons, MSE cost function



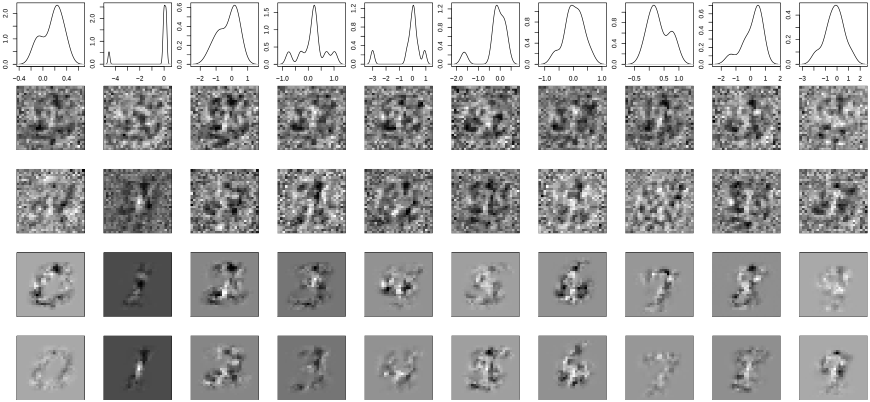
MNIST Dataset training

Tied weights, 1 hidden layer with 100 hidden neurons, CE cost function



CENTERFORBIOLOGICALSEQUENCEANALYSIS CBS

Denoising autoencoder, 1 hidden layer with 100 hidden neurons, CE cost function



Conclusions

Layerwise pretraining with autoencoders was implemented and makes training deep nets possible

Training procedure must be refined to improve performance:

- decreasing learning rate
- automized detection of training success
- regularization and weight decay

Acknowledgements

Morten Nielsen
Henrike Zschach and Pascal Timshel
PISP group