Not all signals are created equal: Dynamic Objective Auto-Encoder for Multivariate Data

Martin Längkvist
Applied Autonomous Sensor Systems
School of Science and Technology
Örebro University
SE-701 82, Örebro, Sweden
martin.langkvist@oru.se

Amy Loutfi
Applied Autonomous Sensor Systems
School of Science and Technology
Örebro University
SE-701 82, Örebro, Sweden
amy.loutfi@oru.se

Abstract

There is a representational capacity limit in a neural network defined by the number of hidden units. For multimodal time-series data, there could exist signals with various complexity and redundancy. One way of getting a higher representational capacity for such input data is to increase the number of units in the *hidden* layer. We propose a step towards dynamically change the number of units in the *visible* layer so that there is less focus on signals that are difficult to reconstruct and more focus on signals that are easier to reconstruct with the goal to improve classification accuracy and also better understand the data itself. A comparison with state-of-the-art architectures show that our model achieves a slightly better classification accuracy on the task of classifying various styles of human motion.

1 Introduction

A common challenge in a classification task for multivariate data is to extract relevant information, capture long-term dependencies, and remove redundancies. However, due to the *curse of dimensionality* and the representational capacity in the model, such dependencies are often difficult to capture. The representational capacity in a neural network, which is defined by the number of hidden units (and is further reduced by introducing regularization), is spent on attempting to reconstruct every input signal. One solution for multivariate data is to manually remove signals that are suspected to contain noise or redundancy. This would decrease the number of dimensions in the input data for the purpose of compensating for the desired increase in model order. We propose an automatic method that will partially ignore signals by assigning each signal an individual model order instead of manually remove signals.

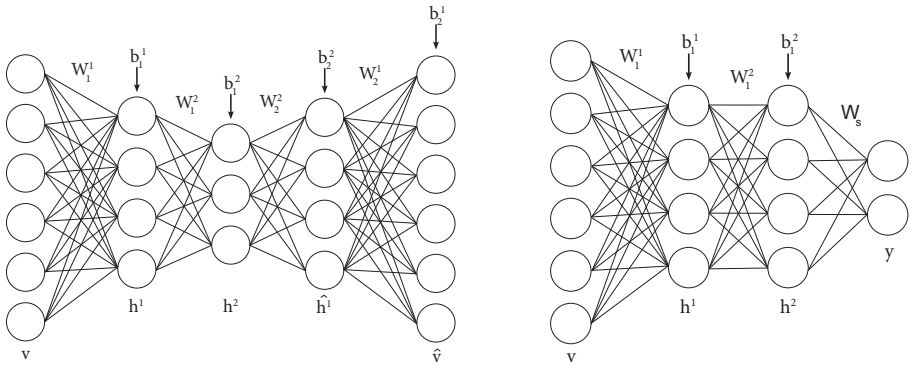
The process of automating certain aspects of learning has gained a lot of focus in recent years in unsupervised feature learning and deep learning [7, 4, 13], such as automatically learning feature representations (see [3] for a recent review), automatically set learning rates [14], hyper parameters [5], and number of hidden units [17]. Various multivariate data applications using deep learning have shown to be successful: modeling human motions [15], audio-visual speech classification [11], symbolic sequences of polyphonic music [6], EEG-Based prediction of epileptic seizures [10], sleep stage classification [8] and bacteria identification with an electronic nose [9]. However, little work on signal selection or automatically set the model order has been done.

One difficult aspect is the criterion for evaluating what is a good signal. In unsupervised learning, one common criterion is the reconstruction error while in supervised learning the classification result is a good indicator. One of the advantages of working with unsupervised learning is that the model can be trained on lots of unlabeled data, which is mostly plentiful and easy to obtain. Therefore, our method will evaluate the input signals during the unsupervised phase of the learning.

Our method is presented in Section 3 and is a variation of a sparse auto-encoder, which will be described in Section 2. The evaluation of our model and classification results on four different styles of motion is presented in Section 4.

2 Background

2.1 Auto-Encoder



(a) (b)

Figure 1: (a) Unsupervised and (b) supervised finetuning of a auto-encoder with 2 layers.

An auto-encoder [1] consists of one input layer, one or more hidden layers, and one output layer. Each hidden layer is first trained individually, followed by a fine-tuning step, see Figure 1. The first hidden layer, h^1, is calculated as:

$$h^1 = \sigma(W_1^1 v + b_1^1) \tag{1}$$

where σ is the logistic function, $\sigma(x) = \frac{1}{1+e^{-x}}$. For input data that is not between 0 and 1, the last layer could have a linear activation function $\sigma(x) = x$. The second hidden layer, h^2, is trained in the same way as the first layer but with h^1 as input. The cost function to be minimized is:

$$J = \frac{1}{2N}\sum_i (v^{(i)} - \hat{v}^{(i)})^2 + \frac{\lambda}{2}\sum_i\sum_j W_{ij}^2 + \beta\sum_j \rho \log\frac{\rho}{p_j} + (1-\rho)\log\frac{1-\rho}{1-p_j} \tag{2}$$

where p_j is the mean activation for unit j. The first term is the reconstruction error term, the second term is the weight decay term and the third term is the sparsity penalty term. A finetuning step of all layers is performed after all layers have been pre-trained. First unsupervised and then supervised. The cost function for supervised finetuning is the same as for unsupervised training except for the reconstruction error term which becomes the cross-entropy loss:

$$J_1 = -\frac{1}{N}\sum_i (1 - y^{(i)})\log(1 - \widetilde{y}^{(i)}) + y^{(i)}\log(\widetilde{y}^{(i)}) \tag{3}$$

where y is the correct label and $\widetilde{y}$ is the predicted label.

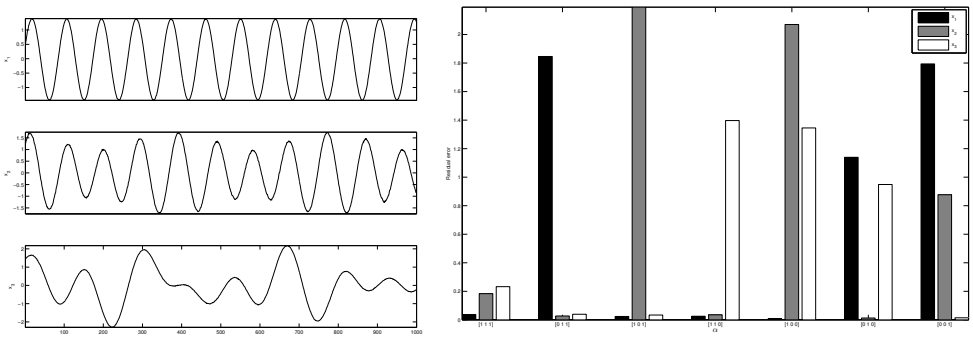
One variation of the auto-encoder is the denoising auto-encoder [16]. The idea behind a denoising auto-encoder is to train a model that is robust to noise in the input data by randomly introduce noise or occlude some of the training data and maintaining a good reconstruction of the original training data.

3 Dynamic Objective Auto-Encoder

3.1 Motivation

Consider a 2-layered neural network with model order 10 in the first layer and model order 5 in the second layer that has been trained on multivariate data. In order to obtain the activations in the top

hidden layer, it requires 10+5 samples of input data. That means that the current top layer hidden activations have a memory of 15 seconds. In order to increase this memory, it is necessary to increase the model order of either layer. The computational cost to increase the model order in the first layer is the number of dimensions in the input data times the increase in model order. The computational cost in the second layer is the number of hidden units in the first layer times the increase in model order. For both cases the computational cost is expensive. However, for most multi-dimensional data, it is not necessary to increase the model order for every signal. The proposed method will attempt to find an automatic way of choosing which signals should have an increased model order and, for compensation to maintain computational complexity, which signals could have a decreased model order. This would increase the model's memory without adding computational complexity.

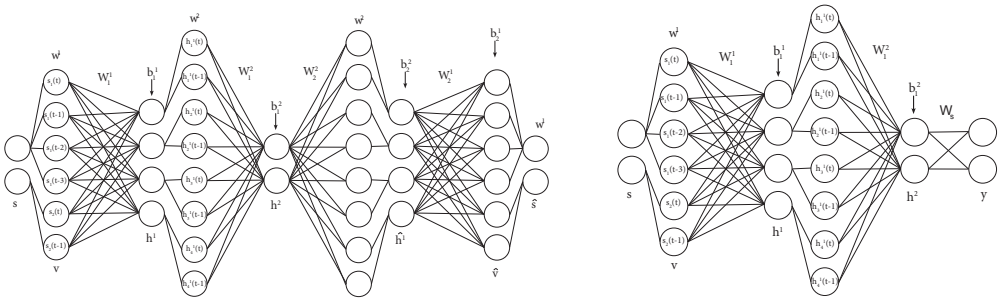


(a) (b)

Figure 2: (a) Three signals of test data (b) Bar plot of reconstruction error of test data for different values of α.

To illustrate, we construct a test signal with 3 dimensions which consist of a sum of sinusoids: $x_i = \sum_{i \in \{1,2,5\}} A \sin(Bt+C)$ where the parameters A, B and C are chosen randomly in each term. The number of hidden units are set to only 3 for demonstration purposes. We set the model order for each signal to 5 and introduce a signal focusing vector $\alpha \in \mathbb{R}^3$, which scales the reconstruction error in the cost function for each signal. The reconstruction error for the different values of α can be seen in Figure 2. When $\alpha = [1, 1, 1]^T$, the reconstruction is lowest for the signal containing only one sinusoid and highest for the signal that is a sum of 5 sinusoids. When one signal is omitted, the reconstruction error of the other two signals is decreased. The error is even further decreased if two of the signals are omitted. This shows that for a model with limited model complexity, the reconstruction of one signal can be improved by limiting the influence other signals have on the model.

3.2 Model



(a) (b)

Figure 3: (a) Unsupervised and (b) supervised finetuning of a dynamic objective auto-encoder with 2 layers.

The proposed model, see Figure 3, uses a variation of sparse auto-encoder with two main differences. The first is how the visible layer is formed by the inclusion of an individual model order vector, $\eta \in \mathbb{R}^S$, where S is the number of signals. The second main difference is the addition of a residual weight vector, $\alpha \in \mathbb{R}^{\sum \eta}$. If $\alpha = [1 \dots 1]^T$ and $\eta = [1 \dots 1]^T$, the model generalizes to a regular sparse auto-encoder. With $\alpha = [1 \dots 1]^T$ and $\eta = [n \dots n]^T$, the model resembles a conditional Restrictive Boltzmann Machine (cRBM) [15] with model order n but without the auto-regressive weights. For multivariate data, $x_s(t)$, the visible layer at time t becomes:

$$v(t) = \begin{bmatrix} x_1(t-\eta_1) \\ \vdots \\ x_1(t-1) \\ \vdots \\ x_S(t-\eta_S) \\ \vdots \\ x_S(t-1) \end{bmatrix} \tag{4}$$

The cost function becomes:

$$\begin{aligned} J = &\frac{1}{2N} \sum_i \sum_j (v_j^{(i)} - \hat{v_j}^{(i)})^2 \cdot \alpha_j + \frac{\lambda}{2} \sum_i \sum_j W_{ij}^2 + \\ &\beta \sum_j \rho \log \frac{\rho}{p_j} + (1-\rho) \log \frac{1-\rho}{1-p_j} + \\ &+ \sum_i \gamma_i (\alpha_p \log(\frac{\alpha_p}{\alpha_i}) + (2-\alpha_p) \log(\frac{2-\alpha_p}{2-\alpha_i})) \end{aligned} \tag{5}$$

The difference in the cost function from a regular sparse auto-encoder is the addition of α in the first term (square error term) and the fourth term (α-penalty term). Any deviation from the value 1 in alpha will increase the α-penalty term, however, the square error term will decrease if $|\alpha| < 1$. The goal is to find an optimal α that minimizes the total cost function. Each layer has a separate η and α. For supervised finetuning, α is not used.

There are therefore two sets of parameters to be optimized: the network parameters, $\theta = \{W, b\}$, and the residual weight vector, α. Training is done in two steps similar to how training is done in sparse coding [12]; first fixate α and update θ, and then update α with the new θ.

The inclusion of α has an effect on the learning. In particular, for the update of θ the error term, δ, in the final layer is:

$$\delta_j = \frac{1}{N} \sum_i (v_j^{(i)} - \hat{v_j}^{(i)}) \alpha_j \tag{6}$$

which means that a lower α will decrease the error term and thus decrease the amount by which the weights and biases in the previous layers are changed during back-propagation. This makes the model "lose interest" in trying to reconstruct signals that have been assigned a low α. Due to the weight decay term, weights that are not updated will decay and eventually reach zero. For the update of α, the gradient becomes:

$$\Delta\alpha = \frac{1}{2N} \sum (v - \hat{v})^2 + \gamma(-\frac{\alpha_p}{\alpha} + \frac{2-\alpha_p}{2-\alpha}) \tag{7}$$

which describes the balance between the reconstruction error and how far α deviates from α_p, which is initially set to 1. Signals with large residual error will generate a lower α.

With the addition of the α-penalty term comes a new parameter γ. One intuitive way of thinking about γ (and also λ or β) is that they act like the spring constant in a flexible spring system. A lower value of γ results in a more flexible system and makes α more open for parameter changes while a higher value of γ will tighten the system and make α less prone for changes. The constant γ can also be set to a vector in order to introduce a kind of forgetting factor and make the system more time-dependant. We use a non-linearity to decrease the γ-value for reconstruction of signals longer

back in time. This will make units for away from the current time frame more likely to generate a low α-value (ignoring that signal at that time) compared to units closer to the current time frame. In particular, the new γ value for unit $n \in [0, \ldots, \eta_i]$ of signal i is $\gamma_n^i = \gamma e^{-n}$.

Increasing or decreasing η is based on α^{L+1} and is iteratively changed after all layers have been trained and finetuned. In particular, each signal which have a α^{L+1} sum total over the mean α^{L+1} value over all signals get $\eta_i = \eta_i + 1$ and those signals (i.e. all other signals) with a lower sum total get $\eta_i = \eta_i - 1$. This means that half the signals, which are considered relatively easy-to-reconstruct signals, gets an increased model order, while relatively hard-to-reconstruct signals get a decreased model order.

The model is first trained for five epochs with only θ being updated in order to let the model settle in before updating α. The reason for this is that is difficult for signals with assigned low α value to do a come-back. Early stopping based on the validation error is implemented in order to prevent over-fitting and the hyper-parameters are set by random sampling [2].

Algorithm 1 Train one layer of Dynamic objective sparse auto-encoder

```
initialize η, θ
repeat
  for all layers, l = 1 : L do
    initialize α^l
    if epoch < 5 then
      θ ← θ, α^l
    else
      θ ← θ
      α^l ← θ
    end if
  end for
  initialize α^{L+1}
  θ, α^{L+1} ← unsupervised finetuning of all layers
  θ ← supervised finetuning of all layers
  η ← η, α^{L+1}
until convergence
```

4 Results and Comparisons

4.1 Motion capture Data

A dataset of total length of 164 seconds with 4 locomotive styles of motion (jog, jump, run, walk) is obtained from the CMU Graphics Lab Motion Capture Database. Sample rate is 120 frames per second. Pre-processing is done by downsampling by 4. The ground-plane forward velocity is calculated from (and replacing) the X and Z position coordinates. The following signals can be removed: toes and hand (noisy), fingers and thumb (not recorded) as well as clavicle (noisy). However, for the purpose of demonstrating the strength of our algorithm, they will not be removed.

4.2 Influence of α

Figure 5 shows the contribution from each term in the objective function over time. At epoch 10, α is starting to update which increases the error for the α-penalty term but lowers the cost for reconstruction error. This illustrates that α acts like a boost to the system by ignoring signals that are difficult to reconstruct and the total cost is reduced. The sparsity penalty term is slightly reduced but the weight decay term is mostly unaffected with the introduction of α.

4.3 Influence of γ

The difference between γ as a constant and a vector can be seen in Figure 6. A darker color indicates a lower α value and thus time frames that have a reduced reconstruction error cost. With the introduced time-dependant mechanic of decaying γ, the appearance of α is more structured over time and

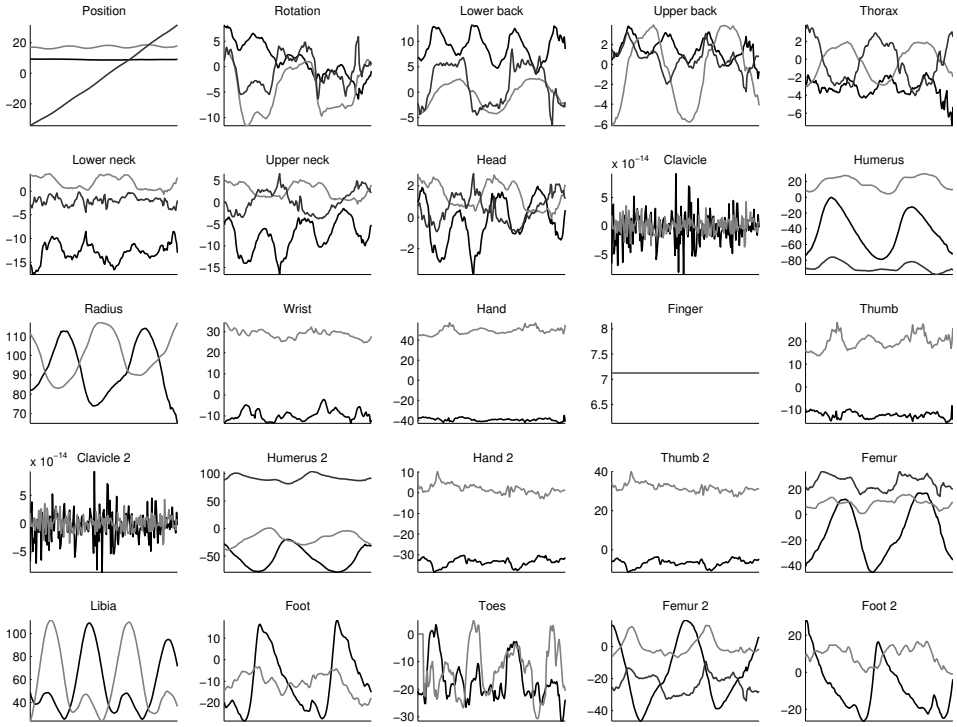


Figure 4: Unnormalized motion capture data for 5.8 seconds of jogging.

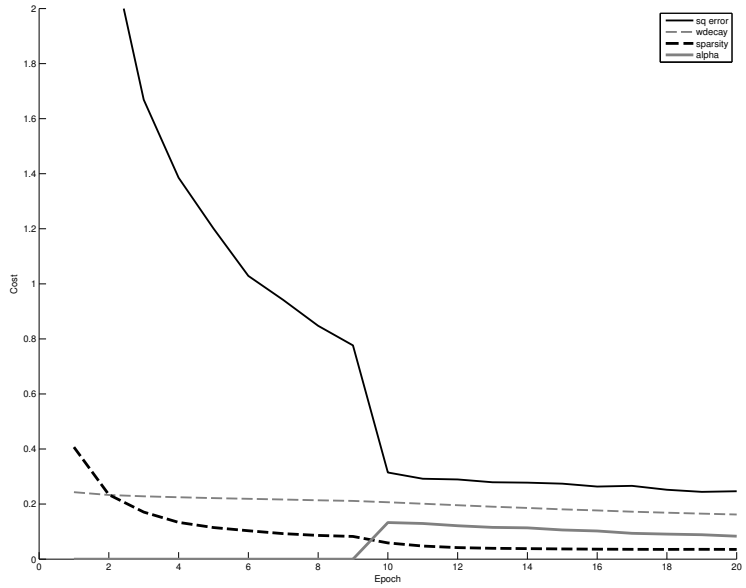
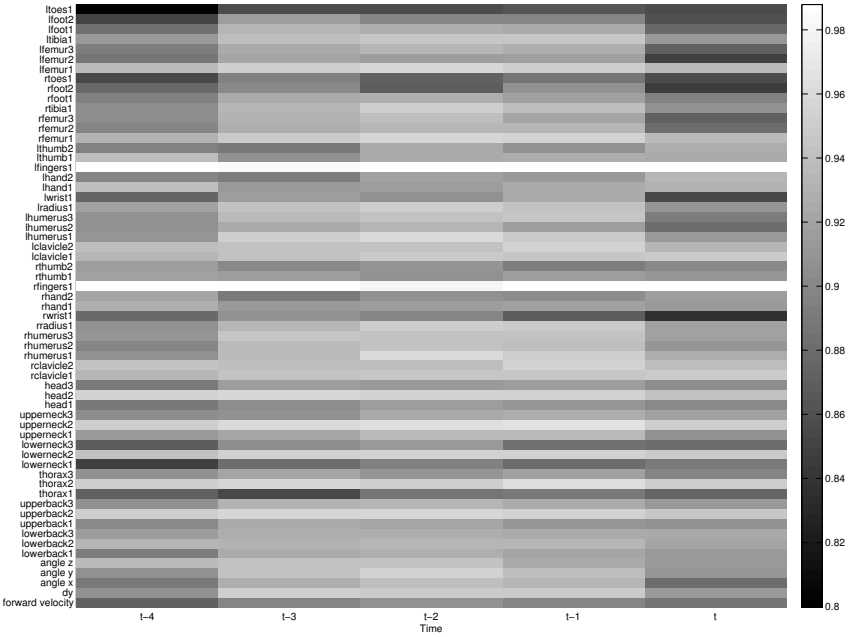
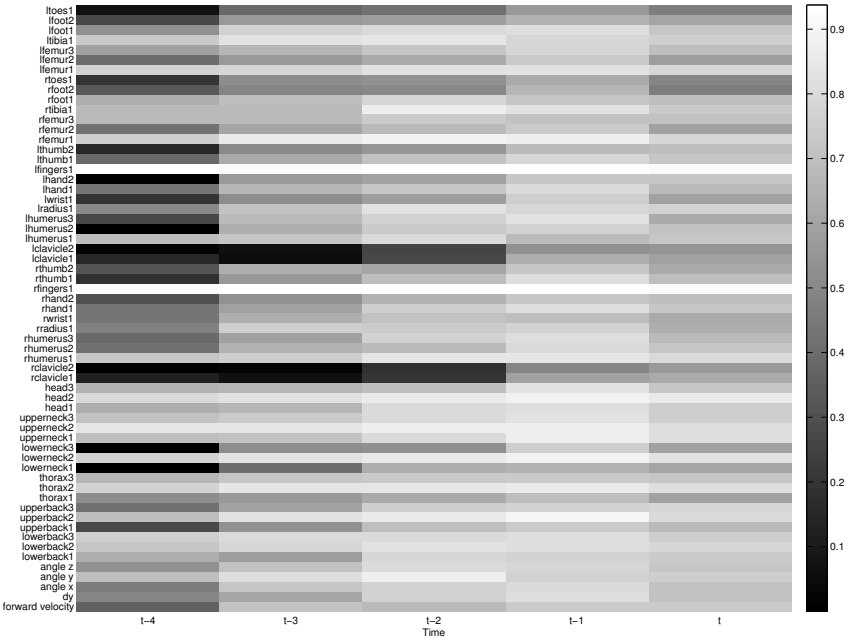


Figure 5: Plot of each term in the cost function for a dynamic objective auto-encoder (doAE) where α is starting to update at epoch 10.

problematic signals, such as left and right clavicle, is better captured. Both methods assign a high α value for the signals for finger movements (which are constant, see Figure 4), which illustrates that our method does not reject such trivial signals at the unsupervised phase of the learning. However, such signals should be manually removed before training anyway.



(a)



(b)

Figure 6: α from training first layer with γ as (a) a constant (b) a vector with reduced γ for units further away from current time frame.

4.4 Classification results

Table 1 shows the classification accuracy for the classification of four different styles of motion with four different architectures. The architectures used are our dynamic objective Auto-Encoder (doAE), a regular sparse auto-encoder (sAE), a denoising Auto-Encoder (dAE), and a conditional Restricted Boltzmann Machine (cRBM). All models use two layer of hidden units and the number of hidden units in each layer is set to 100. Model order is set to 5 in both layers for cRBM and doAE.

For sAE and dAE the input window is 5 samples. Experiments are performed with 5-fold repeated random sub-sampling validation. We see that the classification accuracy is very similar between all

Table 1: Classification accuracy (mean $\pm$ standard deviation) [%] from four different architectures for the task of classifying four styles of motions.

	Accuracy
sAE	89.1 ± 5.4
cRBM	88.7 ± 5.3
dAE	89.7 ± 4.8
doAE	90.2 ± 3.5

four models. However, our doAE achieved the best result and had the smallest standard deviation. This result was achieved after three iteration updates of η, see Table 2. A fourth iteration did not improve the result further.

Table 2: Classification accuracy (mean $\pm$ standard deviation) [%] using dynamic objective Auto-Encoder (doAE) for iterations of η.

	Accuracy
doAE, 1st iteration η	88.6 ± 5.5
doAE, 2nd iteration η	89.6 ± 4.9
doAE, 3rd iteration η	90.2 ± 3.5

5 Discussion and Future work

This paper shows that an automatic method that finds an optimal individually signal model order improves the learning of feature representations. We applied the model to a motion classification task and archived an improvement in classification accuracy compared to other state-of-the-art architectures. A few comments about the proposed method can be made. Firstly, there is no implementation of auto-regressive connections as in the cRBM. Instead, we introduced the hyper parameter γ to scale the cost for units further back in time. Secondly, the current implementation resets θ after each change of η, which could be handled differently by changing the parameters in θ directly instead by adding/deleting rows in the weight and biases vectors. Finally is the question on what to do with trivial signals, such as the finger signals in the motion capture data. One solution is to evaluate such signals in the supervised phase, which will be the next step for this algorithm.

Acknowledgements

The motion capture data used in this project was obtained from `mocap.cs.cmu.edu` and was created with funding from NSF EIA-0196217.

References

[1] Yoshua Bengio. Learning deep architectures for AI. Technical Report 1312, Dept. IRO, Universite de Montreal, 2007.

[2] Yoshua Bengio. Practical recommendations for gradient-based training of deep architectures. arXiv:1206.5533, 2012.

[3] Yoshua Bengio, Aaron Courville, and Pascal Vincent. Unsupervised feature learning and deep learning: A review and new perspectives. arXiv:1206.5538, 2012.

[4] Yoshua Bengio, Pascal Lamblin, Dan Popovici, and Hugo Larochelle. Greedy layer-wise training of deep networks. *Advances in Neural Information Processing Systems 19 (NIPS 2006)*, pages pp. 153–160, 2006.

[5] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13:281–305, February 2012.

[6] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In *Proceedings of the Twenty-nine International Conference on Machine Learning (ICML)*, 2012.

[7] G. E. Hinton, Osindero S., and Teh Y. A fast learning algorithm for deep belief nets. *Neural Computation 18*, pages 1527–1554, 2006.

[8] Martin Längkvist, Lars Karlsson, and Amy Loutfi. Sleep stage classification using unsupervised feature learning. *Advances in Artificial Neural Systems*, 2012, 2012. doi:10.1155/2012/107046.

[9] Martin Längkvist and Amy Loutfi. Unsupervised feature learning for electronic nose data applied to bacteria identification in blood. In *NIPS workshop on Deep Learning and Unsupervised Feature Learning*, 2011.

[10] Piotr Mirowski, Deepak Madhavan, and Yann LeCun. Time-delay neural networks and independent component analysis for eeg-based prediction of epileptic seizures propagation. In *Association for the Advancement of Artificial Intelligence Conference*, 2007.

[11] Jiquan Ngiam, Aditya Khosla, Mingyu Kim, Juhan Nam, Honglak Lee, and Andrew Y. Ng. Multimodal deep learning. In *In Proceedings of the Twenty-Eigth International Conference on Machine Learning*, 2011.

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

[13] Marc'Aurelio Ranzato, Christopher Poultney, Sumit Chopra, and Yann LeCun. Efficient learning of sparse representations with an energy-based model. In J. Platt et al., editor, *Advances in Neural Information Processing Systems (NIPS 2006)*. MIT Press, 2006.

[14] Tom Schaul, Sixin Zhang, and Yann LeCun. No more pesky learning rates. arXiv:1206.1106, 2012.

[15] Graham Taylor, G. E. Hinton, and Sam Roweis. Modeling human motion using binary latent variables. In *Advances in Neural Information Processing Systems*, 2007.

[16] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. *Electroencephalogr. Clin. Neurophysiol.*, pages 119–124, 1982.

[17] Guanyu Zhou, Kihyuk Sohn, and Honglak Lee. Online incremental feature learning with denoising autoencoders. In *In Proceedings of the 15th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2012. JMLR W&CP 22.