Doctoral Dissertation

Modeling Time-Series with Deep Networks

Martin Längkvist
Technology

Örebro Studies in Technology 0
Örebro 2014
Modeling Time-Series with Deep Networks
Örebro Studies in Technology 0

Martin Längkvist

Modeling Time-Series with Deep Networks
Title: Modeling Time-Series with Deep Networks

Publisher: Örebro University, 2014
www.publications.oru.se

Printer: Printer Name

ISSN 1650-8580
ISBN 000-00-0000-000-0
Abstract

Deep learning is a relatively new field that has shown promise in a number of applications and is currently outperforming other algorithms on a variety of commonly used benchmark data sets. One attractive property of deep learning methods is that they take advantage of unlabeled data, which is plentiful and cheaper to obtain than labeled data, in order to construct its own features which reduces the need for heavy pre-processing of the data. However, much focus has been on applications and methods for static data and not so much on time-series data.

Learning models of complex high-dimensional time-series data introduces a number of challenges that either require modifications to the learning algorithms or special pre-processing of the data. Some of the signals in multivariate time-series data are often redundant since they come from sensors with different properties or spatial locations but measure the same real-world phenomenon. Furthermore, sensors are subject to errors in the measurements due to faults, noise, and sensor bias. Therefore, a common approach to analyzing multivariate time-series data is to heavily pre-process the data to reduce the noise and complexity with noise-reduction techniques, feature extraction and signal removal. However, many of these steps require expertise of the domain which is difficult and expensive to acquire, or could even be non-existent.

The primary contribution of this thesis is the algorithmic modifications to a deep learning algorithm that enables the algorithm to better handle multivariate time-series data. The aim is to change the amount of impact each input signal has on the feature learning. This reduces the influence that noisy or task-irrelevant inputs have on the learned features.

The secondary contribution of this thesis is the investigation of the feasibility to construct features from unlabeled raw time-series data. An advantage of using deep networks is the promise of unsupervised feature learning that removes the need to manually hand-design features. However, many of the reported successful applications that uses deep learning, and especially those applied to time-series data, still have used some form of feature extraction as a pre-processing step. This thesis investigates the importance of feature extrac-
tion for time-series data by comparing the performance from a deep network trained on raw data with models trained on feature transformed data.

A final contribution of this thesis is the application of deep learning methods to new data sets that can follow the success deep learning methods has had in computer vision applications. This thesis takes the first step by using new, challenging, and interesting multivariate time-series data sets and suggests that they can be used as benchmark data sets in order to further develop deep learning algorithms specific for multivariate time-series data.
Acknowledgements

First of all I would like to thank my supervisor Amy Loutfi for all her guidance during all aspects of my research. From formulating the topic of the thesis to the very last sentence of the thesis she has guided me in the right directions, usually in disguise of rhetorical thought-provoking questions. Thank you Amy for allowing me the opportunity to explore the topics that I’m passionate about and for supporting me. I am grateful to my opponent and the committee members that took the time to review my work. A big thanks goes to my co-supervisor Lars Karlsson for his detailed revisions of papers and the final thesis. I am especially grateful to Lars Karlsson, Federico Pecora, and Mathias Broxvall for giving me the initial opportunity to work with The Comfortable Sleeplab project, which thought me a lot about machine learning and inspired the pursuit for new state-of-the-art methods. I also thank Silvia Coradeschi for her encouragements and, together with Amy, a pleasant visit to SATRA University in India.

From SASTRA University and the Centre for Nanotechnology & Advanced Biomaterials lab (CeNTAB), that I had the privilege to visit during my research, I would like to give a special thanks to Prof. John Bosco Balaguru Rayappan for the informative and pleasant visits in Sweden and India during the Indo-Sweden project. I also thank Ganesh Kumar Mani and Prabakaran Sankar for being such good friends and making me feel I have a second home in India.

There are many people at the AASS group that I would like to thank. From the Cognitive Robotic Systems lab I thank Marjan Alirezaie, Annica Kristoffersson, Andreas Persson, and Hadi Banaee for the support and being good friends. I am also grateful to Achim Lilienthal, Marco Trincavelli, Victor Hernandez Bennett, Sepideh Pashami and Sahar Assadi from the Mobile Robotics and Olfaction lab for taking interest in my research regarding electronic nose data analysis.

From Örebro University Hospital, which I was involved with in several projects, I would like to thank Lena Leissner and Meeri Sandelin at the sleep unit of the neuroclinic for the interesting discussions about sleep data analysis. From the microbiology lab I would like to acknowledge Lena Barkman for the
bacteria sample preparation. I also thank Bo Söderquist and Per Thunberg for encouraging research collaborations between the university and the hospital.

A big gratitude goes to my family who always take an interest in whatever project I am involved in. Finally, my biggest gratitude goes to Shan for always cheering for me and giving me suggestions and ideas that provided new perspectives.
List Of Publications

This work is a compilation of journal and workshop papers. Below is a list with a description of each publication and an index that is used as reference throughout the rest of this compilation thesis:


This paper reviews recent work using deep learning techniques for time-series problems. The paper first gives an introduction to the properties of time-series data and what makes them challenging. A brief introduction to deep learning models is then provided as well as how these can be modified to make them more suitable for time-series data. Some additional tools for modeling temporal-coherence such as convolution and regularization are briefly covered. The rest of the paper looks at the recent work for time-series modeling categorized by application. The applications covered are speech recognition, music recognition, motion capture modeling, electronic nose data classification, and medical data modeling.


This paper applies a deep belief network (DBN) to polysomography recordings of model brain activity, muscle tension, and eye movements of patients with sleep apnea. The DBN is trained on both raw data and on pre-defined features. The features are chosen by a combination of features used in the literature and hand-designed features. The results are compared to a traditional shallow Gaussian mixture model (GMM) architecture. The best
classification results were achieved with the DBN using hand-picked features. A surprising result is that a DBN using raw data outperformed the GMM with features, which means that the DBN is able to construct features from raw data that are more meaningful than some of the ones used in the literature. The trained DBN is then used in an experiment of finding anomalous data collected in a different environment than the training data, namely from data collected in the home of a healthy person using self-applied electrodes. The reconstruction error is measured over time and it is shown how this can give an indication of when an error in the data has occurred.

Paper III  

This paper concerns the use of deep belief networks and stacked auto-encoders for detecting gases that are described in the literature as markers for spoiled meat. Three gas sensor based on nanostructured zinc oxide with different dopants are exposed to gases with varying concentration. The task is to classify each sample into the correct gas, concentration, and type of sensor that is used. These sensors have been custom-made to react well to a specific gas. However, the response time for them could range from 15 seconds to 15 minutes. The focus in this paper is on achieving a fast classification using as little input data as possible. The results are compared to an approach where the features are based on knowing the maximum response. The result was that a deep model using only 6 seconds of input data outperformed a feature-based approach that required the full sensor response time.

Paper IV  
Martin Längkvist and Amy Loutfi, Learning Feature Representations with a Cost-Relevant Sparse Autoencoder, International Journal of Neural Systems, Accepted for publication, 2014

This paper presents an extension of a sparse auto-encoder that gives the model a selective attention of the input data. The proposed method is evaluated on a number of static image data sets that contain task-irrelevant information. The properties of the proposed method allows the model to learn features that ignores such task-irrelevant information and instead learn more task-relevant features. Two strategies for achieving selective attention are proposed. The first strategy is to use knowledge of the labels with a feature selection algorithm on the input data in
order to rank the inputs and then focus more on reconstructing
the highly-ranked inputs. The second strategy is unsupervised
and does not require knowledge of the labels. Here the selective
attention is instead decided by the reconstruction of each input
and an input with a higher reconstruction error is focused on less
in order to save the representational capacity for other inputs.
The proposed method achieved better or comparable results on
the MNIST variations data set and Jittered-Cluttered NORB
compared to previous works that generally used a larger number
of hidden units.

Paper V  Martin Längkvist, Lars Karlsson, and Amy Loutfi, Selective At-
tention Auto-encoder for Automatic Sleep Staging, Biomedical
Signal Processing and Control, Under revision, 2014

This paper uses the same algorithmic contribution as Paper IV
and applies the idea of selective attention on a temporal auto-
coder on multivariate time-series data for the task of auto-
matic sleep staging. While Paper II uses a DBN to construct
new features from raw unlabeled data this paper uses pre-defined
features in order to instead focus on the advantage of selective
attention. The two methods of a fixed and adaptive selective
attention is further explored. The results are that both meth-
ods of selective attention outperformed a standard auto-encoder
and that the fixed method that bases the selective attention on
the knowledge of the labels was better than the unsupervised
adaptive method.

Paper VI  Martin Längkvist and Amy Loutfi, Unsupervised Feature Learn-
ing for Electronic Nose data Applied to Bacteria Identification
in Blood, NIPS workshop on Deep Learning and Unsupervised
Feature Learning, 2011

This workshop paper is the first paper that used generative deep
learning models on electronic nose data.

Paper VII  Martin Längkvist and Amy Loutfi, Not all signals are created
equal: Dynamic Objective Auto-Encoder for Multivariate Data,
NIPS workshop on Deep Learning and Unsupervised Feature
Learning, 2012

This workshop paper is the first paper that explored the idea of
selective attention for time-series data. The ideas in this paper
were further developed and resulted in Paper VI and Paper VII.

Paper VIII  Ganesh Kumar Mani, Prabakaran Sankar, Martin Längkvist,
Amy Loutfi, and John Bosco Balaguru Rayappan, Detection of
This paper uses a conditional Restricted Boltzmann Machine to detect spoiled meat. While Paper III showed that the gas and concentration of spoiled meat markers could be detected, this paper focuses on the actual application of detecting spoiled goat meat that has been exposed to Indian outdoors temperatures.
# Contents

1 Introduction  
  1.1 Contributions .................................................. 2  
  1.2 Organization .................................................... 3  

2 Representation learning  
  2.1 Greedy layer-wise pre-training ................................. 5  
  2.2 Overfitting and Regularization ................................. 6  
  2.3 Hyperparameters .................................................. 6  
  2.4 Optimization ..................................................... 7  
  2.5 Classification and regression .................................... 7  
  2.6 Modules for deep learning ..........................  
    2.6.1 Restricted Boltzmann Machine ............................. 8  
    2.6.2 Conditional Restricted Boltzmann Machine .............. 9  
    2.6.3 Auto-Encoder ............................................... 10  
    2.6.4 Temporal Auto-Encoder .................................... 12  

3 Representation learning for multivariate time-series data  
  3.1 Sleep stage classification ................................. 14  
    3.1.1 Odour identification with an electronic nose ........... 22  
  3.2 Discussion ...................................................... 27  

4 Selective attention auto-encoder  
  4.1 Proposed model ............................................... 32  
    4.1.1 Fixed weighting vector .................................. 33  
    4.1.2 Adaptive weighting vector ................................. 33  
  4.2 Selective Attention auto-encoder for Automatic Sleep Staging . 33  
  4.3 Character recognition on MNIST variations .................. 36  
  4.4 Object recognition on Jittered-Cluttered NORB .............. 37  
  4.5 Discussion ...................................................... 39
5 Conclusions
  5.1 Challenges and contributions .............................................. 41
  5.2 Critical assessment .......................................................... 42
  5.3 Societal impact ............................................................... 44
  5.4 Future work ..................................................................... 45

References ................................................................. 47
## List of Figures

2.1 (a) RBM (b) conditional RBM ........................................ 8
2.2 (a) Auto-encoder (b) Temporal Auto-encoder ..................... 11

3.1 Polysomnograph data of the five stages of sleep during 30 seconds. 15
3.2 (a) A hypnogram for one night recording of a patient with sleep apnea. (b) Construction of visible layer from sleep data. ....... 16
3.3 Data processing of the first experimental setup, raw-DBN. ...... 17
3.4 The first 20 of 200 learned features in the first layer. The learned features are of various amplitudes and frequencies. ......... 18
3.5 Estimated sleep stage (red) and true sleep stage (black) before applying HMM. .................................................. 19
3.6 Estimated sleep stage (red) and true sleep stage (black) after applying HMM. .................................................. 19
3.7 Data processing of the second experimental setup, feat-GOHMM. 20
3.8 Overall classification accuracy for one acquisition during feature selection with sequential backwards selection (SBS). ....... 21
3.9 Individual sleep stage classification accuracy during SBS. ....... 21
3.10 Data processing of the second experimental setup, feat-DBN. .. 22
3.11 Electronic nose exposed to bacteria in blood .................... 23
3.12 Sensor response for (a) undoped ZnO, (b) Mn doped ZnO, and (c) F doped ZnO. Solid line is the response towards ethanol and dotted line is the response towards trimethylamine. The color indicates ppm level where red, black, and blue represent low, medium, and high ppm level, respectively .......................... 26

4.1 (a) Values of fixed weighting vector. The method can totally ignore one feature for one sleep stage and fully reconstruct it in another sleep stage. (b) Values of adaptive weighting vector. The method focuses more on some features and sleep stages than others. The difference between values for each feature across the sleep stages is not as high as the fixed method. .............. 35
4.2 Average reconstruction error for each input unit for standard and fixed weighting vector for one sleep stage. The reconstruction error is generally increased for low values of $\alpha$ and decreased for higher values of $\alpha$ when the fixed method is used. . . . . . . . . . . 36

4.3 (a) Input data, (b) Weighting vector for fixed method, and (c) reconstructions using fixed method. . . . . . . . . . . . . . . . . 37

4.4 (a) Input data for Jittered-Cluttered NORB where each row is one category, (b) reconstruction of the input data with a CSSAE, and (c) a subset of the learned features. . . . . . . . . . . 38
### List of Tables

3.1 Transition matrix for sleep stage changes .............................. 19
3.2 Confusion matrix for raw-DBN ........................................... 20
3.3 Classification accuracy [%] (mean ± std) for five sleep stages with three different setups. ............................................. 22
3.4 Classification accuracy [%] for DBNs and cRBMs applied to 10 strains of bacteria in two different media. ............................ 25
3.5 Classification accuracy (mean ± standard deviation) [%] with five-fold cross-validation for the task of classifying material, gas, and ppm level using different set-ups. The number after DBN defines the window width (number of visible units) and the numbers after auto-encoder define the model order in the first and second layer. ......................................................... 26
4.1 Classification accuracy of sleep stage classification with and without post-processing temporal smoothing for a deep belief net (DBN), sparse auto-encoder (SAE), and selective attention auto-encoder (SA-AE) with two different methods of selecting the weighting vector. .................................................. 35
4.2 Classification errors [%] with 95% confidence intervals on MNIST variations using selective attention auto-encoder (SA-AE) with fixed weighting vector, sparse auto-encoder (SAE), supervised neural net (NNet), SVM with Gaussian kernel, 3-layered stacked auto-associator (SAA-3), 1 and 2-layered contractive auto-encoder (CAE-1, CAE-2), 3-layered stacked denoising auto-encoder (SdA-3), deep belief network (DBN-1), and a supervised point-wise gated Boltzmann machine (supervised PGBM). ......................... 38
4.3 Comparison of classification errors on Jittered-Cluttered NORB with previous works and a sparse auto-encoder with cost-sensitive learning with 2000 hidden units. ........................................ 39
Chapter 1
Introduction

Applying high-dimensional raw data directly to a machine learning algorithm often gives unsatisfactory results due to the curse of dimensionality [2]. Therefore, practitioners of machine learning algorithms first extract features from the data in order to capture relevant high-level information and reduce the dimensionality of the data and then apply the feature representations to the learning algorithm. These features are developed by domain-specific experts and the search for better feature representations is an active research area in most fields since choosing the right representation is the key for any successful application. However, developing domain-specific features for each task is expensive, time-consuming, and requires expertise of the data. For example, consider challenging tasks in artificial intelligence such as computer vision and speech recognition that contain data that are complex, noisy, and high-dimensional. Furthermore, most data from real-world problems also have a temporal component and are called sequential data or time-series data.

Time-series data has unique properties that make them challenging to analyze and model. Finding a way to represent time-series data that accurately captures the temporal information is an open question and is crucial because raw time-series data is hard to manipulate in its original structure. Prior knowledge or assumptions about data, such as noise levels, redundancies, and temporal dependencies, is often infused in the chosen model or feature representation [26].

The alternative to using hand-engineered features is to learn the feature representations automatically from the data simply by interacting with it. Unsupervised feature learning [8, 6, 23] does this by learning a layer of feature representations from unlabeled data. The advantage of unsupervised feature learning is that unlabeled data can be utilized, which is plentiful and easy to obtain, and that the features are learned from the data instead of being hand-crafted. These layers of feature representations can then be stacked to create deep networks, which are capable of modeling more complex structures in the data. The field of deep learning aims to find ways to properly construct
and train these deep networks. Unsupervised feature learning and deep learning (together lately referred to as representation learning) have shown to be successful at learning layers of feature representations for static data and have achieved state-of-the-art results on a number of benchmark data sets [6].

However, much of the focus in the deep learning community has been on developing learning algorithms specific for static data, and not so much on time-series data. Some works have applied deep learning algorithms that are suited for static data directly to time-series data, but by doing so the temporal component is lost. Other works modify the learning algorithms to capture the temporal information as well. Many of the experiments that apply deep learning algorithms to time-series data use feature extraction on the original data which means that the learned features are not built from purely raw data.

Current unsupervised feature learning algorithms treat all inputs equally. For some multivariate time-series data sets there may be signals that contain more task-relevant information than others, for example if one of the signals is redundant or contains too much noise. In that case, it can sometimes be beneficial to reduce the dimensionality simply by removing those signals. But by doing so valuable information could be lost and it can be a difficult task to manually distinguish which signals can be removed. For some data sets, it might be the case that some signals are particularly useful during some periods of time and at other times they provide less valuable information. Therefore, the ability to focus on a subset of the inputs and changing this focus over time is currently lacking in current feature learning algorithms.

1.1 Contributions

The contributions of this compilation thesis are:

1. Review the challenges with learning feature representations for structured data and recent progress of applying deep learning models to various time-series problems. (Paper I)

2. Apply deep learning models to novel multivariate time-series data sets. Some of these novel data sets could be considered for new benchmarks sets since they introduce challenges that are suited for deep learning algorithms. This contribution is investigated in Chapter 3. (Paper II, III, V)

3. Show that deep learning models are capable of building useful features from raw multivariate time-series data. Deep learning uses unsupervised feature learning to construct its own features from unlabeled data. Despite this, most experiments performed with deep learning methods on time-series data have done so by using heavily pre-processed data. The use of raw data directly to a classifier often results in poor performance,
which explains the current caution of using raw data. This contribution is explored in Chapter 3. (Paper II, III)

4. Algorithmic modifications to one representational learning algorithm that makes it more suitable for multivariate time-series data. These modifications are implemented in an auto-encoder and the feature learning capabilities are compared to standard state-of-the-art deep learning models. This contribution is presented in Chapter 4. (Paper IV, V)

1.2 Organization

Chapter 2 gives an introduction to unsupervised feature learning and deep learning. This chapter presents the construction, training, finetuning, and classification of a deep network with layers of Restricted Boltzmann Machines or auto-encoders. The process of setting the hyperparameters is discussed as well.

Chapter 3 shows how deep learning models can be applied to novel and challenging multivariate time-series problems. The classification accuracy is compared to a number of approaches that uses either raw input data or transformed input data.

Chapter 4 presents the algorithmic contribution to an existing representational learning algorithm that makes it more suitable for multivariate time-series data. These suggested modifications is implemented on an auto-encoder and compared with the performance of a standard auto-encoders on novel data sets and well-known data sets.

Chapter 5 gives the conclusions of the thesis and proposes some suggestions for future work.
Chapter 2
Representation learning

The choice of data representation has a significant impact on the performance of a machine learning algorithm and much research has been done to find good feature representations that captures the task-relevant information in the data. Designing such desirable features often requires expertise of the current domain. Representation learning algorithms [33, 7, 68, 8, 4, 6] instead attempt to create features from unlabeled data without using any prior knowledge. This has the advantage that there is no need to hand-craft the features and that cheap and plentiful unlabeled data can be utilized. Representation learning algorithms are particularly useful for large, high-dimensional, unlabeled, complex data sets where the dynamics is unknown or where traditional shallow methods fail to accurately model the data because of their limited capacity. They have shown to be successful in a number of challenging AI tasks, such as speech recognition [36, 29], object recognition [55], text classification [18], music recognition [49, 15, 35], activity recognition [46], motion capture modeling [81], emotion classification [87, 93, 37], and in physiological data [91, 54, 75, 88, 90]. The growing interest in representation learning algorithms in recent years is contributed by their application of building deep networks. Deep networks are formed by stacking layers of representation learning modules which enables the deep network to model more complex data structures than a shallow network since the data is processed through multiple non-linear transformations [4].

2.1 Greedy layer-wise pre-training

The resurgence of models with multiple layers of hidden units came with the discovery of greedy layer-wise pre-training [33, 7]. This solution overcame the problem of vanishing gradients [3], which is the fact that the error signal from the top layer diminishes as it reaches the first layer during supervised learning. Greedy layer-wise pre-training provides a more useful parameter initialization than random initialization [23] and is the process of training each layer sep-
arately in an unsupervised manner before finetuning the whole network in a supervised fashion. The training of a deep network therefore consists of (1) greedy layer-wise pre-training of each layer to initialize the model parameters, and (2) supervised finetuning of the whole network to optimize for a specific task. However, with access to a large amount of labeled data, proper initialization, and particular design choices such as using rectified linear units (ReLUs) as the non-linearity [56, 41, 73] makes it possible to skip the pre-training phase.

2.2 Overfitting and Regularization

The concept of overfitting in machine learning occurs when an large model relative to the amount of training data starts to describe the noise present in the data instead of the task-relevant patterns. Overfitting is generally prevented by adjusting the complexity of the model or using regularization. Regularization reduces the allowed parameter space of the model and guides the process of feature learning that hopefully achieves better generalization to unseen data [13]. For example, learning sparse feature representations, which are representations where only a small number of the hidden units are "activated" (outputs close to a value of 1) for each training example, is implemented by adding a sparsity constraint [48, 67] to the objective function. Another regularization technique that achieves robustness to small changes in the input data can be achieved by adding a penalty term that penalizes the Frobenius norm of the encoder’s Jacobian at training points [70] or by reconstructing clean input from corrupted input [86, 85]. Finally, a recently proposed method for regularization is dropout [78], which encourages each feature to learn meaningful representations independently of what the other features have learned by dropping half of the hidden units for each training example.

2.3 Hyperparameters

One of the challenges for training deep networks is the large number of design choices, such as connectivity, architecture, optimization method, and hyperparameters. Each regularization term comes with one or more hyperparameter. The choice of optimization also comes with a number of hyperparameters such as learning rate and momentum. A full grid search over all possible combinations of hyperparameters is impractical for deep networks due to the large amount of parameters so a recommendation is to use random grid search [5] or structured hyperparameter optimization [10, 9]. Some hyperparameters can be set according to some practical recommendations [5]. There has been some work done to reduce the amount of design choices, such as automatically set receptive fields [19], learning rates [72], number of hidden units [94], feature selection by grouping [77] or removing hyperparameters by combining regularization terms as done in sparse filtering [57], dropout [78], and contractive auto-encoder [70]. One recommendation for finding good values for hyperpa-
parameters is to look for methods that evaluates and monitor the unsupervised learned model instead of using the final prediction performance that requires the time-consuming supervised finetuning [6, 5].

2.4 Optimization

The choice of optimization algorithm also plays an important role for training deep networks. Many different optimization algorithms have been used, for example stochastic gradient descent (SGD), conjugate gradient (CG), batch methods such as limited Broyden-Fletcher-Goldfarb-Shanno (L-BFGS), and Hessian-free optimization [52, 53]. Each optimization algorithm comes with its own hyperparameters that need to be optimized together with the model hyperparameters. Some examples of hyperparameters for the optimization algorithm is the learning rate, learning rate decay, mini-batch size, and number of training iterations. The L-BFGS has the advantage of automatically setting the learning rate but it is recommended for large-scale problems to use SGD to reduce training time [14]. It has been argued that the learning rate is the most important hyperparameter to tune correctly [5].

2.5 Classification and regression

Once the layer(s) of hidden units have been pre-trained, the network is specialized for a specific task. Two common tasks are classification and regression. In a classification task the output of the model is a representation of the category that the input data belongs to. A classification system is constructed with a deep network by attaching a layer of "softmax" units on the top hidden layer. Each softmax unit represents one class and the output from softmax unit \( y_i \) is the probability that the current input data belongs to class \( k \). The weight matrix between the top layer hidden units and the softmax layer, \( W_{ij} \), is trained by minimizing the cost function:

\[
J_{\text{softmax}} = -\frac{1}{N} \sum_{n=1}^{N} \sum_{i=1}^{k} y_i^{(n)} \log(\tilde{y}_i^{(n)}) + (1 - y_i^{(n)}) \log(1 - \tilde{y}_i^{(n)}) + \frac{\lambda}{2} \sum_{i} \sum_{j} W_{ij}^2
\]

(2.1)

where \( N \) is the number of training examples in the current mini-batch, \( y_i \) is the prediction that the input data belongs to class \( i \), and \( \tilde{y}_i \) is 1 if the input data belongs to category \( i \) and 0 otherwise. The output from the softmax units is calculated as:

\[
y_i = \frac{\exp \sum_j W_{ij} x_j}{\sum_{k=1}^{k} \exp \sum_j W_{ij} x_j}
\]

(2.2)

where \( k \) is the number of classes. Similarly, the deep network can be used for a regression task if a regressor is placed on top layer hidden units.
2.6 Modules for deep learning

There are many different models that can be used for unsupervised feature learning and be stacked to construct deep networks, for example Restricted Boltzmann Machines (RBM) [33, 34, 48], auto-encoders [68, 7], sparse coding [58], deep Boltzmann machines [27], and K-means [17]. Each model has its own advantages and disadvantages and since the field is still under research, no clear winner of what model is most suitable for deep learning has yet emerged [6]. This section briefly introduces the two most commonly used modules, namely the RBM and the auto-encoder, and how they can be modified to better model sequential data.

2.6.1 Restricted Boltzmann Machine

The Restricted Boltzmann Machine (RBM) is a generative probabilistic undirected graphical model that consists of visible units $v$, hidden units $h$, and bias vector $c$ and $b$, see Figure 2.1a. The weight matrix $W$ connects the visible and hidden units. There are no visible-to-visible, or hidden-to-hidden connections. The energy function and the joint distribution for a given visible and hidden vector is defined as:

\[
E(v, h) = h^T W v + b^T h + c^T v \tag{2.3}
\]

\[
P(v, h) = \frac{1}{Z} \exp^{E(v, h)} \tag{2.4}
\]
where \( Z \) is the partition function that ensures that the distribution is normalized. For Bernoulli-Bernoulli RBM (binary visible and hidden units) the probability that hidden unit \( h_j \) is activated given visible vector \( v \) is given by:

\[
P(h_j|v) = \sigma \left( b_j + \sum_i W_{ij} v_i \right) \quad \text{(2.5)}
\]

The probability that visible unit \( v_i \) is activated given hidden vector \( h \) is given by:

\[
P(v_i|h) = \sigma \left( c_i + \sum_j W_{ij} h_j \right) \quad \text{(2.7)}
\]

where \( \sigma(\cdot) \) is the activation function. A common choice for activation function has been the sigmoid activation function \( \sigma(x) = \frac{1}{1+e^{-x}} \), but is more and more being replaced with rectified linear units [56].

The model parameters \( \theta = \{W, b, v\} \) are trained to maximize the (log)likelihood of the training data. Calculating the gradient of the likelihood function is intractable because of the partition function so instead the gradient is estimated using contrastive divergence (CD) [32], persistent contrastive divergence (PCD) [92, 82], or fast-weights persistent contrastive divergence (FPCD) [83]. For contrastive divergence the update rule is:

\[
\frac{\partial \log P(v)}{\partial W_{ij}} \approx \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{recon}} \quad \text{(2.8)}
\]

where \( \langle \cdot \rangle \) is the average value over all training samples.

Modules of RBMs can be stacked on top of each other to form a deep network. When RBMs are stacked in this manner they are called a deep belief network (DBN) [33]. The output from a lower-level RBM becomes the input to the next level RBM.

### 2.6.2 Conditional Restricted Boltzmann Machine

The conditional RBM (cRBM) [81] is an extension of the RBM. The cRBM has auto-regressive weights that model short-term temporal dependencies and hidden units that model longer-term temporal structures in multivariate time-series data. A cRBM is similar to a RBM except that the bias vectors for the
visible and hidden layers is dynamic and depends on previous visible layers, see Figure 2.1b. The dynamic bias vectors are defined as:

\[ b_j^* = b_j + \sum_{i=1}^{n} B_i v(t - i) \quad (2.9) \]
\[ c_i^* = c_j + \sum_{i=1}^{n} A_i v(t - i) \quad (2.10) \]

where \( A_i \) is the autoregressive connections between visible layers at time \( t - i \) and current visible layer, \( B_i \) is the weight matrix connecting visible layer at time \( t - i \) to the current hidden layer. The model order is defined by the constant \( n \). The probabilities for going up or down a layer is

\[ P(h_j | v) = \sigma \left( b_j + \sum_i W_{ji} v_i + \sum_k \sum_i B_{ijk} v_i(t - k) \right) \quad (2.11) \]
\[ P(v_i | h) = \sigma \left( c_i + \sum_j W_{ij} h_j + \sum_k \sum_i A_{ijk} v_i(t - k) \right) \quad (2.12) \]

The parameters \( W, b, c, A, \) and \( B \), are trained in a similar manner as the RBM using contrastive divergence.

2.6.3 Auto-Encoder

The auto-encoder \([7, 4]\) consists of an encoder and a decoder, see Figure 2.2a. The goal of the auto-encoder is to reconstruct the input data via one or more layers of hidden units. The feed-forward activations in the encoder from the visible units \( v_i \) to the hidden units \( h_j \) is expressed as:

\[ h_j = \sigma_f \left( \sum_i W_{ji} v_i + b_j \right) \quad (2.13) \]

where \( W_{ji} \) is the connection between visible unit \( i \) and hidden unit \( j \), and \( \sigma_f \) is the activation function. A common activation function is the sigmoid function which is defined as \( \sigma_f(x) = \frac{1}{1 + e^{-x}} \). In the decoder phase, the reconstructions of the input layer is calculated as:

\[ v_i = \sigma_g \left( \sum_j W_{ij} h_j + b_i \right) \quad (2.14) \]

Notice the tied weights of the encoder and decoder, that is, the weight matrix for the decoder is the transpose of the weight matrix in the encoder. The use
of tied weights acts as a regularizer and reduces the number of parameters to learn [6]. The activation function in the decoder can be the sigmoid function or the linear activation function $\sigma_g(x) = x$ if values in the input layer are not between 0 and 1.

The cost function for the auto-encoder to be minimized for a minibatch of $N$ training example is expressed as:

$$L(v, \theta) = \frac{1}{2N} \sum_{n=1}^{N} \sum_{i} (v_i^{(n)} - \hat{v}_i^{(n)})^2$$  \hspace{1cm} (2.15)
The auto-encoder can also be given a probabilistic interpretation by defining the cost function as the negative log likelihood of the probability distribution of the visible layer given the hidden layer. With the assumption that this is a Gaussian distribution with mean equal to the reconstruction of the visible layer and identity covariance we get

\[ L = -\log P(v|h) = \frac{1}{2} \sum_i (v_i^{(n)} - \hat{v}_i^{(n)})^2 \]

which is the same as Equation (2.15).

The cost function can be complemented with a number of regularization terms that restricts the allowed parameter space and helps prevent overfitting. The weights of the model in each layer can be kept small by adding a L2 weight decay term, \( \lambda \sum_i \sum_j (W_{ij})^2 \). Sparse feature representations can be achieved by adding the Kullback-Leibler (KL) divergence as a sparsity penalty term, \( \beta \sum_j KL(p||p_j) \), where

\[
KL(p||p_j) = \rho \log \frac{\rho}{p_j} + (1 - \rho) \log \frac{1 - \rho}{1 - p_j}
\]

(2.16)

and \( p_j \) is the mean activation for hidden unit \( j \) over all training examples in the current mini-batch. Each regularization term comes with one or more hyperparameters (\( \lambda, \beta, \rho \)).

Learning in an auto-encoder is the process of finding the model parameters \( \theta = \{W, b\} \) that minimizes the cost function. The model parameters can be learned with backpropagation. For auto-encoders, the decoder is only used in the pre-training phase and is not used in the supervised fine-tuning phase or during classification.

### 2.6.4 Temporal Auto-Encoder

The auto-encoder can be extended to a temporal auto-encoder in order to make it more suitable for multivariate time-series data, see Figure 2.2b. The hidden units depend on the visible units of the current timeframe as well as visible units of previous timeframes. The hidden layer at time \( t \) is calculated as:

\[
h_j = \sigma_f \left( \sum_{k=1}^{n} \sum_i B_{ji}^k v_i(t - k) + \sum_i W_{ji} v_i + b_j \right)
\]

(2.17)

where \( B^i \) is the weight matrix between the hidden layer and visible units at time frame \( t - n \). The reconstruction layer is calculated as:

\[
\hat{v}_i = \sigma_g \left( \sum_{k=1}^{n} \sum_j A_{ji}^k v_i(t - k) + \sum_j W_{ij} h_j + b_i \right)
\]

(2.18)

where \( A^i \) is the weight matrix between visible units at time frame \( t - n \) and the reconstruction of the visible layer at the current time frame \( t \). There is no reconstruction of past visible layers. The past visible layers act as an extra bias term, much similar to a conditional RBM [81].
Chapter 3
Representation learning for multivariate time-series data

This chapter presents in more detail how deep learning models can be applied to multivariate time-series problems. The analysis of time-series have been an active research area for decades [39, 20]. Modeling high-dimensional time-series data shares some of the challenges for modeling static data, such as how to represent the data, how to identify patterns, how to deal with noise, and how to introduce invariance. But it also introduces additional challenges, such as how to capture short and long-term behavior and how to find correlations between signals. The importance of temporal dependency of time-series data can be observed when two seemingly identical input frames at different times actually should be associated with different predictions depending on the preceding input frames.

Time-series data can either be sampled data from a sensor that measures a real-valued continuous process or outputs from simulations of a man-made processes. For long-term time-series data it is common to observe measures that vary periodically (seasonality) or a trend, which may be caused by natural factors or depreciation of the sensor equipment. Sensor-based data often contain noise and might produce unreliable measures. When the goal is to capture a complex real-world occurrence it is often not sufficient to use only one sensor. Therefore, multiple sensors that either have different characteristics or are spatially located are used to create multivariate time-series data. The more sensors that are used the higher the chance is to capture enough information in order to fully understand the examined process. But due to economic and practical considerations the number of sensors is limited which means that only a subset of the variables that would be required to understand the process is available. Reversely, having too many sensors means a higher redundancy and make the data more difficult to analyze.

Some popular tools for analysis of time-series data include Dynamic Time Warping (DTW) [11] for measuring similarity between two time-series and
CHAPTER 3. REPRESENTATION LEARNING FOR MULTIVARIATE TIME-SERIES DATA

motif discovery [16, 74] for finding repeating patterns. But there are also numerous techniques for modeling time-series. These can either be parametric, that is, they measure signal properties such as frequency spectra and signal correlations, or non-parametric. Non-parametric models may contain known equations of the system dynamics or be completely "black-box", which do not use any a priori knowledge of the system dynamics. Such "black-box" models can either be linear, like autoregressive models [51], Linear Dynamical Systems (LDS) [50], and Hidden Markov Model (HMM) [65], or non-linear, like neural networks, RBMs, and auto-encoders. Non-linear, non-parametric models are best suited when the system dynamics are complex or even unknown. The task for such models is to identify the parameters of the assumed model, which is a process called "training" the model. The trained model can then be used for simulation, signal analysis, prediction, fault detection, controller synthesis, or data classification.

This section presents how a RBM and an auto-encoder can be applied for learning features from unlabeled multivariate time-series data. The learned models are then used for the task of classifying sleep stages from polysomnographic (PSG) recordings and identifying odour markers from electronic nose data.

3.1 Sleep stage classification

Sleep stage classification is an important step for diagnosing chronic sleep disorders. The task involves labeling each 30-second segment of data from a polysomnograph (PSG) into pre-defined sleep stages. A PSG recording consists of various physiological signals such as brain activity from an electroencephalography (EEG), eye movements from an electrooculography (EOG), and muscle activity from an electromyography (EMG). Sleep stage labeling is a time-consuming task that is manually performed by human sleep experts. The five pre-defined sleep stages, introduced by Rechtschaffen and Kales (R&K) [69], are awake (W), stage 1 (S1), stage 2 (S2), slow wave sleep/deep sleep (SWS), and rapid-eye-movement sleep (REM). Figure 3.1 shows one epoch of data from a polysomnograph for each of the five stages. Awake stage is characterized by low EEG amplitude, more than 50% EEG alpha activity, high EMG and the presence of slow eye-movements (SEMs) and blinking artifacts in the EOG. In the first stage of sleep, S1, the amount of alpha activity in the EEG is below 50% and the EMG amplitude is decreased. There might be SEMs visible in the EOG in stage 1. In the next sleep stage, S2, there are no SEMs in the EOG and K-complexes and sleep spindles can be seen in the EEG. Slow wave sleep (SWS) is characterized by over 20% of the EEG is delta waves. The EMG amplitude is low and there are typically no eye-movements in the EOG. Finally, REM-sleep has the lowest amplitude in the EMG, clearly visible rapid eye-movements (REM) in the EOG, and EEG with low amplitude, mixed frequencies, and no K-complexes or sleep spindles. A graph that
3.1. SLEEP STAGE CLASSIFICATION

Figure 3.1: Polysomnograph data of the five stages of sleep during 30 seconds.

shows these five stages over an entire night is called a hypnogram. A typical hypnogram can be seen in Figure 3.2a.

A common approach to the time-consuming task of sleep stage classification is to try to capture the valuable information with a set of features and often a feature selection step [63]. The features are designed with the help from sleep technicians in order to mimic the R&K-system and infuse the a priori knowledge of the data. As with most learning algorithms, the choice of feature has a big impact on the classification performance. While many different feature sets and methods for creating a computerized automatic sleep stager
has been done [38, 61, 24, 95, 60, 21], the difficulty to generalize to different patients, sleep labs, or equipment has prevented an automatic sleep stager to be used in a clinical setting, and therefore it can be said no set of universally applicable set of features has yet been found [63, 71]. The ongoing search for an optimal feature set has led to the discovery of new useful features such as frequency bands [40] and fractal exponent [79], which are not directly obvious from the R&K rules.

Instead of using pre-designed features or researching for new ones, this section presents how a deep belief network (DBN) can be used to construct its own feature representation for sleep stage classification from unlabeled raw data without using any a priori expert knowledge. The advantage of this approach is that potentially new and better features, that do not necessary adhere to the R&K system, can be discovered from unlabeled data that better untangles the factors of variations in sleep data and can provide a better generalization.

The data that is used in this section has kindly been provided by St. Vincent’s University Hospital and University College Dublin, and can be downloaded from PhysioNet [28]. The dataset consists of 25 acquisitions from subjects with suspected sleep apnea (sleep-disordered breathing). Each recording consists of 2 EEG channels, 2 EOG channels, and 1 EMG channel. Only one of the EEG signals are used in order to reduce the dimensionality. Labeling of the data has been performed by one sleep technician. All signals are slightly pre-processed by notch filtering, bandpass-filtering, and downsampling.
3.1. SLEEP STAGE CLASSIFICATION

Experiment 1: raw-DBN

Figure 3.3 shows the experimental setup for the use of a deep belief network (DBN) trained on raw data. A hidden markov model (HMM) is used to smooth out the sleep stage transitions. The DBN has two layers with 200 hidden units each. The input in the first layer is the concatenation of EEG, EOG 1, EOG 2, and EMG, see Figure 3.2b. More formally, with window width, \( w \), the visible layer becomes

\[
\mathbf{v} = \begin{bmatrix}
\text{EEG}_1^{64+w} \\
\text{EEG}_1^{64+w} \\
\vdots \\
\text{EMG}_1^{64+w}
\end{bmatrix}
\]

We set \( w \) to be 1 second of data and with no overlap. With four signals and a sample rate of 64 samples per second, the number of visible units is 256. The data is scaled to have values between 0 and 1.

The first layer in the DBN is trained on the raw data and Figure 3.4 shows the weights of the 20 first learned features. The features show that the model are looking for both low and high frequency signals in the EEG and high and low amplitude signals in the EMG. This gives an indication that the model has learned different frequency ranges and amplitudes of the signals. After the features in the first layer has been trained, the same procedure is performed to train the layers in the second layer but using the output from the first layer as input to the second layer. Therefore, the second layer creates features of features and are higher-level features than the features in the first layer. The features in the second layer are problematic to display in a meaningful manner. Both layers are then fine-tuned, first in an unsupervised manner and then in a supervised manner.

A layer of softmax units are attached to the second layer that classifies the hidden activations in the second layer into 5 classes. The softmax layer output a classified sleep stage for each second of input data. This will result in lots of sleep stage switching due to misclassified segments. Therefore, a Hidden Markov Model (HMM) [66] is used to smooth out the sleep stage switching. Figure 3.5 shows the estimated and true hypnograms for patient 5 before applying HMM and Figure 3.6 show the hypnogram after applying
Figure 3.4: The first 20 of 200 learned features in the first layer. The learned features are of various amplitudes and frequencies.
3.1. SLEEP STAGE CLASSIFICATION

The overall classification accuracy before using HMM was 59.3% and after using HMM 84.5% for this particular patient.

Table 3.1 shows the transition matrix learned by the HMM. Beside the obvious fact that probabilities of staying in the same state is very high, it can be observed that going from wake-state or the near to wake-state S1 to slow wave sleep is close to 0.

A full 25 leave-one-out cross-validation is performed to obtain an average classification accuracy. In each validation, 1 of the 25 acquisitions is left out as test data while training and validation samples are randomly drawn from the other 24 acquisitions in a way that each class has equally amount of training examples. The average classification accuracy for all sleep stages is $67.4 \pm 12.9\%$. The confusion matrix for all training examples is shown in Table 3.2. The most difficult stages to classify are S1, which is the stage between being awake and falling asleep, followed by REM-sleep.

<table>
<thead>
<tr>
<th>%</th>
<th>SWS</th>
<th>S2</th>
<th>S1</th>
<th>REM</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>SWS</td>
<td>99.84</td>
<td>0.113</td>
<td>0.015</td>
<td>0.007</td>
<td>0.029</td>
</tr>
<tr>
<td>S2</td>
<td>0.065</td>
<td>99.82</td>
<td>0.043</td>
<td>0.036</td>
<td>0.039</td>
</tr>
<tr>
<td>S1</td>
<td>0.0</td>
<td>0.317</td>
<td>99.51</td>
<td>0.017</td>
<td>0.155</td>
</tr>
<tr>
<td>REM</td>
<td>0.001</td>
<td>0.041</td>
<td>0.027</td>
<td>99.90</td>
<td>0.031</td>
</tr>
<tr>
<td>W</td>
<td>0.0</td>
<td>0.021</td>
<td>0.172</td>
<td>0.002</td>
<td>99.81</td>
</tr>
</tbody>
</table>

Table 3.1: Transition matrix for sleep stage changes
Experiment 2: feat-GOHMM

A shallow feature-based approach is designed, see Figure 3.7, in order to compare the results from the DBN used on raw data. The architecture consists of a feature extraction step with 28 hand-crafted features, followed by a Gaussian observation hidden Markov model (GOHMM) and lastly a HMM for smoothing. The features that are used are a mix of feature from the literature, personal conversations with sleep experts, and newly designed features. More precisely, the used features are relative power for five frequency bands (delta (0.5–4Hz), theta (4–8Hz), alpha (8–13Hz), beta (13–20Hz), and gamma (20–64Hz)) of all signals, median of EMG, standard deviation of EOG, correlation coefficient between both EOG, entropy, kurtosis, and spectral mean of all signals, and fractal exponent [59, 64] of EEG. All features are normalized.

Feature selection is done by Sequential Backward Selection (SBS), which starts with the full set of features and greedily removes a feature after each iteration step. A Principal Component Analysis (PCA) with five principal components is used to further reduce dimensionality, followed by a Gaussian Mixture Model (GMM) with five components and a HMM with five output states.

The feature selection step gives an indication of which features are good choices. Figure 3.8 shows the overall classification accuracy for the sleep stages during the feature selection step of one acquisition and Figure 3.9 shows the individual classification accuracy. When all 28 features are used the classification accuracy is very poor. The easiest sleep stage to classify is slow wave sleep (S3) and the most difficult is S1. The accuracy stay at a plateau until the feature of kurtosis for EMG is removed where the accuracy for S1 is im-

<table>
<thead>
<tr>
<th>Expert</th>
<th>% awake</th>
<th>S1</th>
<th>S2</th>
<th>SWS</th>
<th>REM</th>
</tr>
</thead>
<tbody>
<tr>
<td>awake</td>
<td>68.4</td>
<td>13.4</td>
<td>2.5</td>
<td>0.7</td>
<td>15.1</td>
</tr>
<tr>
<td>S1</td>
<td>20.3</td>
<td>33.1</td>
<td>24.8</td>
<td>1.6</td>
<td>20.2</td>
</tr>
<tr>
<td>S2</td>
<td>1.0</td>
<td>6.3</td>
<td>76.5</td>
<td>9.1</td>
<td>7.1</td>
</tr>
<tr>
<td>SWS</td>
<td>0.1</td>
<td>0.0</td>
<td>11.1</td>
<td>88.8</td>
<td>0.0</td>
</tr>
<tr>
<td>REM</td>
<td>21.1</td>
<td>6.9</td>
<td>11.1</td>
<td>0.8</td>
<td>60.1</td>
</tr>
</tbody>
</table>

Table 3.2: Confusion matrix for raw-DBN
3.1. SLEEP STAGE CLASSIFICATION

proved (at the cost of decreasing the accuracy for REM-sleep and awake). The maximum overall accuracy is achieved after 12 features have been removed. The same validation as the DBN with raw data input is used. The average

classification accuracy for the GOHMM setup is 63.9 ± 10.8%, which is lower than the average classification accuracy for the DBN on raw data setup.
CHAPTER 3. REPRESENTATION LEARNING FOR MULTIVARIATE TIME-SERIES DATA

Figure 3.10: Data processing of the second experimental setup, feat-DBN.

<table>
<thead>
<tr>
<th></th>
<th>All stages</th>
<th>SWS</th>
<th>S2</th>
<th>S1</th>
<th>REM</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>feat-GOHMM</td>
<td>63.9±10.8</td>
<td>70.8 ± 26</td>
<td>71.6 ± 19</td>
<td>32.3 ± 24</td>
<td>57.0 ± 35</td>
<td>68.0 ± 21</td>
</tr>
<tr>
<td>raw-DBN</td>
<td>67.4±12.9</td>
<td>72.7 ± 28</td>
<td>77.6 ± 19</td>
<td>31.0 ± 25</td>
<td>66.2 ± 30</td>
<td>69.8 ± 23</td>
</tr>
<tr>
<td>feat-DBN</td>
<td>72.2±9.7</td>
<td>74.7 ± 31</td>
<td>76.3 ± 19</td>
<td>38.2 ± 27</td>
<td>70.5 ± 30</td>
<td>78.5 ± 16</td>
</tr>
</tbody>
</table>

Table 3.3: Classification accuracy [%] (mean ± std) for five sleep stages with three different setups.

Experiment 3: feat-DBN

The DBN with raw data as input outperformed a GOHMM with hand-picked features as input. The question is whether this is because the features that are learned from the raw unlabeled data are better than the hand-crafted features or because of the use of the more powerful DBN is better than using a Gaussian Mixture Model for data modeling. To examine this question, a similar 2-layered DBN with a softmax classifier is constructed but trained on the 28 hand-crafted features as input instead of raw data, see Figure 3.10. This setup achieved an average classification accuracy of 72.2±9.7%, which is better than the other two setups. This means that using the more powerful DBN gives better performance than a GOHMM regardless of the inputs are features or raw data. But since using a DBN with features give better performance than a DBN with raw data, it is still advantageous to use feature extraction when there is a good knowledge about what features to use. The use of a DBN on features also has the advantage that there is no need for a feature selection step.

Summary

To summarize, a DBN has been trained on raw sleep data in order to construct features and was compared to two approaches that uses feature extraction. The average classification accuracy and standard deviation for 25 full-night sleep recordings are presented in Table 3.3.

3.1.1 Odour identification with an electronic nose

The sense of smell is important for both humans and animals in order to localize and evaluate the quality of food sources, detect dangers, or even finding
a mate. The ability to quantify and objectively analyze odours has many applications and has been an active research area for decades. Machine olfaction is achieved by using an electronic nose (e-nose), which consists of an array of chemical gas sensors with various selectivity and a pattern recognition system [25]. The development and analyzing of e-nose data is therefore interesting for both sensor communities and pattern recognition communities. The array of chemical gas sensors consists of typically 4-32 sensors that measures the electrical signal when analyte molecules come into contact with the chemically sensitive material in the sensor. The inclusion of multiple sensors with different properties increases gas selectivity but may also introduce a high redundancy. The most common material used for gas sensors is tin-dioxide semiconductor, which is doped in order to provide selectivity. Examples of selectivity include selectivity towards volatile organic compounds (VOC), alcohols, sulphurs, carbon-based molecules e.t.c. E-noses have been used in experiments with food, beverage, and air quality monitoring, environmental monitoring, specific gas detection, and medical applications.

A typical response from a three-phase sampling system (baseline phase, sampling phase, and recovery phase) of an e-nose after removing bias and normalization is seen in Figure 3.11. The sampling has been performed in a contained instrument that automatically regulates the flow of air. The gas to be analyzed is exposed to the sensor array in the sampling phase while a reference gas is used during baseline and recovery phase in order to return the sensor values to the initial state.

![Graph showing sensor values over time](image)

**Figure 3.11:** Electronic nose exposed to bacteria in blood
There is valuable information in the static sensor values but also in the dynamic appearance in the transients of the sampling and recovery phases. The usual approach to analyzing e-nose data is to use feature extraction \[30\]. Unlike polysomnograph recordings, there are no human experts on e-nose data that can guide the design of features and infuse prior knowledge of this complex and unintuitive data. Instead, the choice of features is based on trial-and-error. Some examples of common features from the plethora of tried features are the maximum sensor response, area under the response, transient derivatives, or model parameter identification, e.g., auto regressive model parameters or dynamic time warping coefficients. As with sleep data it can be concluded that an ideal feature set has not been found. Due to the high redundancy in the signals, dimensionality reduction techniques is performed before applying a classifier such as ANN or SVM. This approach has given good classification results in closed sampling environments with a single gas source. However, the use of unsupervised feature learning and deep learning would remove the need for the guessing of hand-designing features and allows us to work directly on the raw data. It would also allow us to take advantage of unlabeled data and it could open the door for more challenging applications such as open sampling systems with multiple gas sources.

In this section, RBM-based methods and sparse auto-encoders are applied to e-nose data for the purpose of quantify and classify various odours. The classification accuracy will be compared to a feature-based approach.

Detecting bacteria in blood and agar

This experiment focuses on a medical application where the electronic nose is used to detect the presence of bacteria in two different media, namely blood and agar solution. The bacteria have been chosen among bacteria that is typically found in blood (E.coli, P. aeruginosa, S. aureus, K. oxytoca, P. mirabilis, E. faecalis, S. lugdunensis, P. multocida, S. pyogenes, and H. influenzae) and can lead to septicemia (blood poisoning).

The sampling system used for the bacteria in blood data set is the NST 3220 Emission Analyzer with 22 sensors. This data set consists of a total of 800 samples (80 samples per bacteria) and each sample is 5 minutes long. The device used for data collection for bacteria in agar is the Cyranose 320, which is a commercial generic e-nose system consisting of a sensor array of 32 conducting polymer sensors. This data set consists of a total of 740 samples and each sample is 2 minutes long. Both data sets are sampled at 2 Hz.

These data sets are applied to a deep belief network (DBN) and a conditional RBM (cRBM). Both models are trained with 1 and 2 layers with 200 hidden units each. Each class was randomly divided into training and testing sets. The data is downsampled by a factor of 2 and normalized to contain values between 0 and 1. The visible layer of the DBN is constructed by con-
catenating the first 20 samples of each sensor response in the sampling phase, similar to how the visible layer is constructed for sleep data in Figure 3.2b.

The classification accuracy for the experiments are seen in Table 3.4. A RBM with 1 layer (RBM-1) gives almost the same accuracy as a feature-based approach for the bacteria in blood data set and was improved with 2 layers. No comparable result to a feature-based approach using a RBM could be obtained with the bacteria in agar dataset. This is believed to be due to a much shorter recovery time for the agar dataset, resulting in a difficulty in properly aligning the sensor data. However, using a cRBM on the agar dataset gave much better results. In particular when a smaller model order was used. The number of levels for cRBM did not seem to have a significant effect on the classification accuracy for either of the data sets. One possible explanation for this could be that a cRBM is more suitable when the number of sensors is high (the agar data set had 32 sensor, the blood data set had 22). It is difficult for e-nose data to verify what the model has learned by visualizing the learned model parameters, as e-nose data is inherently difficult to understand.

Table 3.4: Classification accuracy [%] for DBNs and cRBMs applied to 10 strains of bacteria in two different media.

<table>
<thead>
<tr>
<th>Setup</th>
<th>Bacteria in blood</th>
<th>Bacteria in agar</th>
</tr>
</thead>
<tbody>
<tr>
<td>Features + SVM [84]</td>
<td>93.7</td>
<td>84.0</td>
</tr>
<tr>
<td>RBM-1</td>
<td>93.8</td>
<td>41.9</td>
</tr>
<tr>
<td>RBM-2</td>
<td>96.2</td>
<td>47.3</td>
</tr>
<tr>
<td>cRBM-1, model order 5</td>
<td>85.0</td>
<td>96.1</td>
</tr>
<tr>
<td>cRBM-2, model order 10-10</td>
<td>75.0</td>
<td>90.0</td>
</tr>
<tr>
<td>cRBM-2, model order 5-5</td>
<td>85.0</td>
<td>96.0</td>
</tr>
</tbody>
</table>

Detecting meat spoilage markers

The use of features and DBNs require a fix amount of data before a classification is made. For sensors with a slow response it can take several minutes before the maximum response is reached. In some applications it is necessary to perform a quick classification in just a few seconds. The experiment in this section focuses on how a fast classification system can be built using deep learning algorithms. In particular, a rapid and accurate detection system is designed for the purpose of detecting markers for spoilage in meat. We use DBNs and sparse auto-encoders that consider only the transient response from 3 sensors with different surface materials. These sensors have been custom-designed for detecting meat spoilage markers, however the response time is quite long. The gases that will be explored are ethanol and trimethylamine (TMA). Each
Figure 3.12: Sensor response for (a) undoped ZnO, (b) Mn doped ZnO, and (c) F doped ZnO. Solid line is the response towards ethanol and dotted line is the response towards trimethylamine. The color indicates ppm level where red, black, and blue represent low, medium, and high ppm level, respectively.

Sample will be classified into the type of sensor that is used, the type of gas, and the approximate concentration of the gas.

Figure 3.12a, 3.12b, 3.12c shows the total of 64 samples from 3 sensors with different surface materials (undoped ZnO, Mn-doped ZnO, and F-doped ZnO) exposed to ethanol and TMA with three ppm intervals (<20 ppm, 20–50 ppm and >50 ppm). The signal amplitude and response time vary between the sensors, e.g., the response from undoped ZnO reaches a maximum value after 6–14 min, F-doped ZnO reaches a maximum value after 1–5 min, and Mn-doped ZnO reaches a maximum value after just a few seconds.

The visible layer and signals are constructed and pre-processed in the same manner as the bacteria data sets.

Table 3.5: Classification accuracy (mean ± standard deviation) [%] with five-fold cross-validation for the task of classifying material, gas, and ppm level using different set-ups. The number after DBN defines the window width (number of visible units) and the numbers after auto-encoder define the model order in the first and second layer.
3.2. DISCUSSION

First, a support vector machine (SVM) classifier is used on seven features (the maximum response (K), the first, second, and third time constant, and the area under the response between two time constants) to achieve a classification accuracy based on knowing the maximum response. This setup achieved 89.0%, 60.1% and 42.9% for the task of classifying material, gas and ppm level, respectively, see Table 3.5. However, this method requires the knowledge of the maximum response to extract the required features, which for some sensors can take up to 14 minutes.

A 2-layered DBN with 200 hidden units in each layer is trained on the first 25 seconds of the response and achieved a classification accuracy of 86.8%, 83.7% and 49.5%, which is better at classifying gas and ppm level and slightly worse at classifying the material compared to using a SVM with seven features. The accuracy decreases when the input window decreases.

A 2-layered sparse auto-encoder with 200 hidden units in each layer achieved 93.2%, 84.3%, 61.2% with the first 25 seconds of the response, which is already better than the feature-based approach on all three categories. The model order is then decreased to use 10, 6, and 4 seconds of input data. The accuracy is higher than an the feature-based approach for 10 and 6 seconds of input data but not for 4 seconds. This means that the fastest classification at run-time, while still having comparable result with an approach where the maximum value is known, is 6 seconds of input data.

Here, the deep learning algorithms were trained on a few samples and compared to feature based approaches that required knowledge of the full sensor response. We did not compare to shallow methods on the same initial data points as the deep methods. The reason for this is that the goal is not to compare the performance between shallow and deep methods but to highlight the possibility of using raw data in the dynamic response instead of performing feature extraction, which will be crucial for the open-sampling scenario.

3.2 Discussion

In this chapter we have seen how deep learning can be applied to multivariate time-series such as sleep recordings and e-nose data. Both these data sets are new data sets for the deep learning community that not only provide a challenge for the learning algorithms but also reveal some of their shortcomings. The features were trained from unlabeled raw data without using any a priori knowledge. The classification accuracy from deep learning methods were compared to models that used domain-specific features or methods that used features that required more information from future measurements. For the sleep data, a DBN trained on raw data was capable of constructing meaningful features and achieved comparable results to a feature-based approach. For the rapid odour identification, a deep network was capable of extracting enough valuable information in just the few first seconds rather than using features that required the full sensor response.
Even though we used the term "deep learning" to describe the used method, the networks used in these experiments are not very deep. For the sleep recordings we used a maximum of two number of hidden layers. Deep learning consists of both the unsupervised feature learning aspect and the use of the learned features for building deep networks. The focus in this chapter has been on the representation learning part of deep learning and not on building very deep networks. More discussion about this is given in Section 5.2.

Emerging areas with new sensors that lack domain experts fuel the need for unsupervised feature learning approaches. In order to compare the performance between different machine learning algorithms, a benchmark data set is often used. In this thesis we propose that sleep data and e-nose data are good candidates for benchmark data sets. For a good sleep data benchmark data set there are a number of criteria that needs to be met: (1) The data must be labeled by 2 or more experts to avoid mislabeled data since expert conformance is between 80% between sleep experts [62]. If possible, the data should be labeled in segments shorter than 30 second epochs to reduce the amount of mislabeled data. (2) The data set should contain recordings from both healthy patients and patients with different sleep disorders. A future application could be to classify the correct sleep disorder. (3) Electron-placement and equipment should follow an agreeable standard.

For an e-nose data set to be suitable as a benchmark set for deep learning the data set should be large and challenging. The e-nose data sets used in this work are rather simplistic because they have been collected in a closed and controlled environment with exposure of only a single gas. A large array of varying gas sensors used in an open sampling environment with multiple gas sources at different concentrations will be far more complex and it is for this scenario that deep learning methods could be useful. Any potential sensor redundancy is a challenge that should be dealt with on the software side. We chose to use data from controlled sampling systems first since this is the first time, to our knowledge, that deep learning methods have been directly applied to raw e-nose data.

In order to move the intelligence from the feature extraction step to the learning algorithm, there is a need to look into methods that are less influenced by human engineers in order to discover relevant patterns in complex multivariate time-series data. This is especially true for the e-nose community and other domains where the knowledge about the data is limited or even absent. There are a number of emerging applications for data from an e-nose and for it to be interpreted in a consistent manner it is not tractable to have its own hand-picked features for each application. Therefore, a generalized method is required and unsupervised feature learning provides a framework towards this direction.

This does not mean that using unsupervised feature learning is always better than using pre-defined features. For the sleep data, a better accuracy was achieved when a DBN was trained on features rather than training a DBN
3.2. DISCUSSION

directly on raw data. In fact, many works that apply deep learning methods to frequency-domain rich time-series, e.g. speech recognition, use features extraction before applying the data to the learning algorithm. This indicates that there is still room for improvements in the unsupervised feature learning phase of learning to learn even better features.

The approaches presented in this chapter have focused on treating the input as static data. The problem with this approach is that the importance of time is not captured. Modeling time-series faces many of the same challenges as modeling static data, such as coping with high-dimensional observations and nonlinear relationships between variables. But by simply ignoring time and applying static models to time series one disregards much of the rich structure present in the data. When taking this approach, the context of the current input frame is lost and the only time-depencies that are captured is within the input size. In order to capture long-term dependencies, the input size has to be increased, which can be impractical for multivariate signals or if the data has very long-term dependencies.

Another problem with the current approach is that there is no signal selection that deals with the problematic signals. Sleep data is often recorded with more than 10 EEG channels in order to measure the brain activity at different locations. However, only one of the possible EEG channels was arbitrarily used in this chapter. The use of all EEG channels would provide more information about the sleep process but would also introduce redundancy. For e-nose data, the redundancy is most likely present but there is no reliable way of knowing beforehand which sensors are redundant and can be removed.

These reasons show that there is room for improvements in the learning algorithm specific for multivariate time-series. Some suggestions for improvements are explored in the next chapter.
Chapter 4
Selective attention auto-encoder

This chapter introduces some modifications to the standard sparse auto-encoder in order to learn better features and make the model more suitable for multivariate time-series data. The proposed method is particularly suitable for data sets that may contain partially noisy or task-irrelevant signals. In an auto-encoder, the representational capacity is spent on attempting to reconstruct every signal, even if that signal is problematic. This can be solved by manually removing signals that are suspected to be problematic or redundant. However, for some data sets it is not clear which signals can be removed beforehand since the dynamics between the signals are complex or unknown. Instead of removing inputs with dimensionality reduction techniques, the proposed model in this chapter will weight the reconstruction error of each input unit. This will reduce the influence that same inputs have on the feature learning and therefore give the model a kind of selective attention on the input data.

The advantages of this method is that there is no need to perform any dimensionality reduction on the input data and the model will learn more specialized features since the selective attention of the model can change over time. In many problems, not all signals are useful all the time and it is rare that a signal is completely useless and can be completely removed. The usefulness of a signals is often correlated to the category the current data belongs to. The challenge with this method is to decide which inputs should be focused on at what times. Two different strategies are proposed in this work. The first method requires labeled data and is based on performing a feature selection algorithm on the input data in order to rank the importance of each input unit. The second proposed method does not require the knowledge of the labels and is based on the the idea that inputs that are too difficult to reconstruct will consume too much of the capacity of the model and should be less focused on, which is especially true if that signal contains much noise.

A similar weighting of the input units has previously been done with the point-wise gated Boltzmann machine (PGBM) [77], cost-sensitive learning [22, 31, 42], early-saturation prevention [89], sparse feature learning [67].
The proposed method is different in that it focuses on the unsupervised phase of learning and does not require a pre-trained model. An importance sampling strategy has also been done for a large-scale natural language processing task by reconstructing only the non-zeros and a random subset of the zeros [27].

The method is evaluated on a multivariate time-series data set for the task of automatic sleep staging and on two computer vision tasks, namely character recognition with the MNIST variations data set and object recognition with the Jittered-Chuttered NORB data set. These data sets have in common that they may contain category-dependent task-irrelevant inputs.

4.1 Proposed model

The regular auto-encoder attempts to reconstruct all input units equally, as can be seen from Equation 2.15. Context-dependent selective attention is implemented in an auto-encoder by introducing a weighting vector, \( \alpha_i^k \), which indicates the probability that unit \( i \) should be reconstructed if the input belongs to class \( k \). In the cost function, the weighting vector is multiplied with the reconstruction error. The model generalizes to a regular sparse auto-encoder if \( \alpha_i^k = 1 \) for \( \forall i, k \). If \( \alpha_i^k \) is small than the weighted reconstruction error for the \( i \)th unit for inputs that belong to class \( k \) is decreased. This means that reconstruction error for this unit has less of an impact on the weights and biases in the previous layers during back-propagation. The cost function of the proposed model is defined as:

\[
L(v, \theta, \alpha^k) = \frac{1}{2N} \sum_{n=1}^{N} \sum_i (v_i^{(n)} - \hat{v}_i^{(n)})^2 \cdot \alpha_i^k + f(\alpha_i^k) \quad (4.1)
\]

where the weighting vector penalty term \( f(\alpha_i^k) \) adds a penalty for changing the values of the weighting vector from 1. Without this term every value in the the weighting vector would go to 0 and the model would not try to reconstruct anything.

The cost function can also be derived from the probabilistic interpretation of the auto-encoder. In a regular auto-encoder all inputs are treated equally because of the assumption that each input unit given the hidden units has a Gaussian distribution with unit variance. If we instead set the precision (inverse variance) as a learnable parameter, i.e., \( P(v|h) = N(\hat{v}, \alpha^{-1}) \) we get the following cost function:

\[
L = -\log P(v|h) = \frac{1}{2N} \sum_{n=1}^{N} \sum_i (v_i^{(n)} - \hat{v}_i^{(n)})^2 \alpha_i - \sum_i \log \alpha_i \quad (4.2)
\]

which gives us the weighting vector penalty term \( f(\alpha_i^k) = -\sum_i \log \alpha_i \). Another possibility is to use the Kullback-Leibler divergence from Equation (2.16) with
4.2. SELECTIVE ATTENTION AUTO-ENCODER FOR AUTOMATIC SLEEP STAGING

Both methods have been tried during our experiments and they have shown to give equal results.

A crucial aspect of the proposed method is how to set the weighting vector $\alpha^k_i$. Two methods for doing this is explored in this thesis.

4.1.1 Fixed weighting vector

The first method is when the values are fixed and are set with a feature selection algorithm. The weighting vector penalty term $f(\alpha^k_i)$ is removed since the values of the weighting vector are constant. In this work we use the score from the t-test algorithm on the input data to set the values of the weighting vector and is calculated as:

$$t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}}}$$

where $\mu_i$, $\sigma_i$, and $n_i$ is the mean, standard deviation and number of examples of a variable that belongs to class $i$. For more than two classes the t-test algorithm is calculated for each class in a one-vs-all decomposition.

The scores are then normalized so that each value is between 0 and 1 and so that the sum of each weighting vector is equal to the number of input units. This reduces the difference between the reconstruction error and the other regularization terms so that the hyperparameters do not need to be re-adjusted.

4.1.2 Adaptive weighting vector

When using an adaptive weighting vector, the values of the weighting vector are learned together with the model parameters. The weighting vector is parametrized according to $\alpha = \log(1 + \exp(\alpha))$, where $\alpha$ is the learnable parameter. This parametrization is done in order to avoid numerical problems, i.e., keeping $\alpha > 0$. If $\alpha_i$ decreases from 1, the square error term will decrease and the weighting vector penalty term will increase. However, for values of $\alpha_i$ above 1 the reconstruction error will increase but the weighting vector penalty term is negative. A hyperparameter $\gamma$ is introduced to the weighting vector penalty term in Equation (4.2) in order to control the trade-off between reconstruction error and the weighting vector values deviating from 1 and prevent the values to blow up. During learning, the algorithm determines a weighting vector that minimizes the total cost function.

4.2 Selective Attention auto-encoder for Automatic Sleep Staging

PSG recordings contain signals that are redundant or less informative than others depending on the current sleep stage. For example, the most important
factor for distinguishing between REM and stage 1 is the amplitude of the EMG, while the EEG channels during these stages show a similar appearance. But for discriminating between stage 1 and 2 the amplitude of the EMG is no longer as crucial as the appearance of sleep events in the EEG [76]. A selective attention auto-encoder is used in this section to learn more specialized features that only reconstructs the part of the input data that is suspected to be relevant for the current sleep stage. This ensures that the representational capacity is not used on inputs that are task-irrelevant for the current sleep stage.

The same data that was used in Section 3.1 is used to train both a standard and a selective attention auto-encoder. The temporal auto-encoder presented in Section 2.6.4 is used with a time order of 10. The size of the model is set to 500 hidden units and is trained on the training set for 200 epochs. Over-fitting is prevented by using the trained model that minimized the cost function on the validation set. The hyperparameters are set with random grid search [5]. Training is first performed with unsupervised pre-training with stochastic gradient descent (SGD) with momentum and decaying learning rate and then supervised finetuned with a layer of softmax units attached on the hidden layer. Finally, the trained model is used to perform feed-forward classification on the test set.

The values of the weighting vector \( \alpha^k_i \) are set in two different ways: (fixed) the values are set using the absolute and normalized raw scores from the t-test feature selection algorithm and are not updated, and (adaptive) all values are initially set to 1 and then updated together with the model parameters during learning. The standard auto-encoder will be referred to (standard) and is the equivalent of setting all values of \( \alpha^k_i \) to 1 for all \( i \) and \( k \). The weighting vector is not used during the supervised finetuning phase.

Table 4.1 shows the classification results when using the three types of standard, fixed, and adaptive weighting vector and with and without temporal smoothing with a Hidden Markov Model (HMM). The fixed and adaptive achieves a higher classification accuracy on the test set than a standard auto-encoder. The fixed weighting vector gave better classification accuracies than the adaptive weighting vector. Temporal smoothing increased the accuracies for all three methods.

The weighting vector values for the fixed method is shown in Figure 4.1a. It can be seen that the attention for each feature vary between high and low depending on their importance for the current sleep stage. For example, the attention to EEG frequency features is higher for stage 1, which is defined for having a decreasing amount of alpha-waves, than in REM-sleep, which has a broader range of frequency range. Another example is the amplitude of the EMG which is variable for all stages except for REM-stage where it is at its lowest and acts as one of the defining features for REM-stage sleep. The final weighting vector values for the adaptive method has a different look and can be seen in Figure 4.1b. This method does not use the knowledge of the
### 4.2. SELECTIVE ATTENTION AUTO-ENCODER FOR AUTOMATIC SLEEP STAGING

#### Method

<table>
<thead>
<tr>
<th>Method</th>
<th>Without temporal smoothing</th>
<th>With temporal smoothing</th>
</tr>
</thead>
<tbody>
<tr>
<td>DBN [43]</td>
<td>-</td>
<td>72.2 ± 9.7</td>
</tr>
<tr>
<td>SAE</td>
<td>66.9 ± 4.94</td>
<td>71.9 ± 5.2</td>
</tr>
<tr>
<td>SA-AE (Fixed $\alpha$)</td>
<td>71.0 ± 5.93</td>
<td>77.7 ± 6.9</td>
</tr>
<tr>
<td>SA-AE (Adaptive $\alpha$)</td>
<td>70.3 ± 6.55</td>
<td>76.5 ± 6.5</td>
</tr>
</tbody>
</table>

Table 4.1: Classification accuracy of sleep stage classification with and without post-processing temporal smoothing for a deep belief net (DBN), sparse auto-encoder (SAE), and selective attention auto-encoder (SA-AE) with two different methods of selecting the weighting vector.

Figure 4.1: (a) Values of fixed weighting vector. The method can totally ignore one feature for one sleep stage and fully reconstruct it in another sleep stage. (b) Values of adaptive weighting vector. The method focuses more on some features and sleep stages than others. The difference between values for each feature across the sleep stages is not as high as the fixed method.

labels so the values are based on the capability of the model to reconstruct each feature at each sleep stage. There are visible vertical and horizontal lines which indicate that in general some features (and sleep stages) are easier to reconstruct than others.

The average reconstruction error for standard and fixed weighting vector can be seen in Figure 4.2. For a standard auto-encoder each input is treated equal so the average reconstruction error for each input unit is around the same. But with a fixed weighting vector the reconstruction error is lower for the inputs that have a higher value of $\alpha$. This shows that the reconstruction
error can be lowered for some inputs if the attention of the model is shifted to allow for a higher reconstruction error for other inputs.

### 4.3 Character recognition on MNIST variations

This section presents the use of the selective attention auto-encoder for the task of recognizing handwritten digits from a static image data set. The data set that is used is the MNIST variations data, which contains the standard MNIST handwritten numbers [44] with added background images (mnist-back-images), added background noise (mnist-back-rand), rotated numbers (mnist-rot), and rotated numbers with background images (mnist-rot-back-images). Each data set consists of 10000 training examples, 2000 validation examples, and 50000 test examples. The input size is $28 \times 28$. The input data can be seen in Fig. 4.3a. A 1-layered selective attention auto-encoder with 500 hidden units and fixed weighting vector is trained on the input data. The weighting vector is set using the t-test feature selection algorithm on the training data and the values can be seen in Figure 4.3b, where each row is the weighting vector for each of the four data sets and each column is the category-specific weighting vector for each of the ten digits. The reconstructions from the trained model
can be seen in Figure 4.3c, where it can be seen that the background of the reconstructions are blurred because the model is focused to only reconstruct the inputs that contain the task-relevant digit.

Table 4.2 shows the test error rate for the selective attention auto-encoder (SA-AE) compared to other methods. It should be noted that our model uses one layer of 500 hidden units while some of the other works use model sizes between 500 and up to 6000 per layer. The proposed method has a lower test error rate for all data sets compared to a SAE, NNet, and SVM (except for mnist-rot). For mnist-back-images the SA-AE achieved a lower classification error than a 3-layered sparse auto-associator. For mnist-back-rand comparable results to SdA-3 and CAE-2 were achieved and a lower error than SAA-3, and CAE-1. For mnist-rot the SA-AE only performed better than DBN-1. One reason for this is that mnist-rot does not contain any added noise so there is little gain in paying less attention to a subset of the input. For mnist-rot-back-images a lower test error rate is achieved compared to SAA-3, DBN-1 and CAE-1.

The two methods that achieved a lower test error rate on all 4 data sets than the proposed method are the supervised PGBM [77] and CAE-2 [70]. The difference between our method and the CAE-2 is that the CAE-2 uses two layers of hidden units with 1000 units in each layer while our method only uses one layer with 500 hidden units. The lower test error for the supervised PGBM could be explained by its use of generative feature selection both at the high-level of learned features from a pre-trained RBM and at the low-level at the visible units by blocking task-irrelevant units while our method only uses feature selection on the low-level input units before training.

4.4 Object recognition on Jittered-Cluttered NORB

The Jittered-Cluttered NORB data set consists of stereo-scopic images of size $108 \times 108 \times 2$ of five small toy figures (animal, human figure, airplane, truck, and car) taken at 6 lighting conditions, 9 elevations, and 18 azimuths where
The central object has been jittered, and disruptive background clutter and a distractor object has been added. A sixth category is created by only adding background clutter and a distractor object but no toy figure. Figure 4.4a shows some examples of the input data. The training set is split into a training and validation set resulting in 234446 training images, 57154 validation images, and 58320 test images. Each image is downsampled to $32 \times 32 \times 2$ and normalized by subtracting the mean of the image and divided by the average standard deviation of all training pixels [56]. A selective attention auto-encoder (SA-AE) of 500 and 2000 hidden units with fixed weighting vector is trained on the training images. The values of the weighting vector are updated with the t-test algorithm for each training mini-batch due to the large size of the training set. Figure 4.4b shows the reconstructions of the trained model and Figure 4.4c shows a subset of the learned features.

Table 4.2: Classification errors [%] with 95% confidence intervals on MNIST variations using selective attention auto-encoder (SA-AE) with fixed weighting vector, sparse auto-encoder (SAE), supervised neural net (NNet), SVM with Gaussian kernel, 3-layered stacked auto-associator (SAA-3), 1 and 2-layered contractive auto-encoder (CAE-1, CAE-2), 3-layered stacked denoising auto-encoder (SdA-3), deep belief network (DBN-1), and a supervised point-wise gated Boltzmann machine (supervised PGBM).

<table>
<thead>
<tr>
<th>Model</th>
<th>mnist-back-images</th>
<th>mnist-back-rand</th>
<th>mnist-rot</th>
<th>mnist-rot-back-images</th>
</tr>
</thead>
<tbody>
<tr>
<td>SA-AE</td>
<td>18.39 ± 0.35</td>
<td>10.98 ± 0.28</td>
<td>12.86 ± 0.30</td>
<td>47.68 ± 0.45</td>
</tr>
<tr>
<td>SAE</td>
<td>19.97 ± 0.36</td>
<td>13.60 ± 0.31</td>
<td>14.23 ± 0.38</td>
<td>53.56 ± 0.45</td>
</tr>
<tr>
<td>NNet [45]</td>
<td>27.41 ± 0.39</td>
<td>20.04 ± 0.35</td>
<td>18.11 ± 0.34</td>
<td>62.16 ± 0.43</td>
</tr>
<tr>
<td>SVM$_{rbf}$ [45]</td>
<td>22.61 ± 0.37</td>
<td>14.58 ± 0.31</td>
<td>11.11 ± 0.28</td>
<td>55.18 ± 0.44</td>
</tr>
<tr>
<td>SAA-3 [45]</td>
<td>23.00 ± 0.37</td>
<td>11.28 ± 0.28</td>
<td>10.30 ± 0.27</td>
<td>51.93 ± 0.44</td>
</tr>
<tr>
<td>CAE-1 [70]</td>
<td>16.70 ± 0.33</td>
<td>13.57 ± 0.30</td>
<td>11.39 ± 0.28</td>
<td>48.10 ± 0.44</td>
</tr>
<tr>
<td>CAE-2 [70]</td>
<td>15.50 ± 0.32</td>
<td>10.90 ± 0.27</td>
<td>9.66 ± 0.36</td>
<td>45.23 ± 0.44</td>
</tr>
<tr>
<td>SdA-3 [85]</td>
<td>16.68 ± 0.33</td>
<td>10.38 ± 0.27</td>
<td>10.29 ± 0.27</td>
<td>44.49 ± 0.44</td>
</tr>
<tr>
<td>DBN-1 [45]</td>
<td>16.15 ± 0.32</td>
<td>9.80 ± 0.26</td>
<td>14.69 ± 0.31</td>
<td>52.21 ± 0.44</td>
</tr>
<tr>
<td>supervised PGBM [77]</td>
<td>12.85 ± 0.30</td>
<td>6.87 ± 0.23</td>
<td>—</td>
<td>44.67 ± 0.44</td>
</tr>
</tbody>
</table>

Figure 4.4: (a) Input data for Jittered-Cluttered NORB where each rows is one category, (b) reconstruction of the input data with a CSSAE, and (c) a subset of the learned features.
Table 4.3: Comparison of classification errors on Jittered-Cluttered NORB with previous works and a sparse auto-encoder with cost-sensitive learning with 2000 hidden units.

<table>
<thead>
<tr>
<th>Model (Hidden units)</th>
<th>Test error rate [%]</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAE (500)</td>
<td>26.42</td>
</tr>
<tr>
<td>SA-AE (500)</td>
<td>20.04</td>
</tr>
<tr>
<td>SA-AE (2000)</td>
<td>14.48</td>
</tr>
<tr>
<td>RBM (binary) [80] (4000)</td>
<td>18.7</td>
</tr>
<tr>
<td>RBM (NReLU) [80] (4000)</td>
<td>16.5</td>
</tr>
<tr>
<td>Conv Net [47] (8-24-100)</td>
<td>16.7</td>
</tr>
</tbody>
</table>

Table 4.3 shows the test error rate for our models and previous works. The SA-AE with 500 hidden units achieves a lower test error rate compared to a sparse auto-encoder (SAE) but not lower than a RBM or a multi-layered convolutional net. But when the number of hidden units is increased to 2000, the SA-AE achieved a lower test error rate than a RBM with twice the number of hidden units with rectified linear units and a 6-layered convolutional net.

4.5 Discussion

In this chapter we have seen how an auto-encoder can be modified to have class-dependent selective attention. The proposed approach was tested on multivariate time-series data for the task of automatic sleep stage classification and on static data for the task of handwriting digit recognition and object recognition. The selective attention was implemented by weighting the reconstruction error for each input unit and was set with two different methods. The first method used a supervised feature selection algorithm on the input data to obtain a fixed weighting vector, and the second method used and learned an adaptive weighting vector during the unsupervised phase of learning.

The fixed approach of using supervised feature selection slightly outperformed the adaptive approach in terms of classification accuracy in our experiments. A possible reason for this is that the fixed approach uses more information about the data such as the labels and therefore gets a more task-relevant selective attention. Both the fixed and the adaptive approach of weighting the inputs achieved better classification than a standard auto-encoder that treats all input equally.

The suitability of using a selective attention depends on the current problem and the properties of the data. For data sets that do not contain much task-irrelevant information there is no or little gain by using a model with selective attention. Whereas for data sets that contain much task-irrelevant information the gain of using selective attention is more noticeable. The property of context-dependency of the proposed model also means that better per-
formance is more likely to be achieved on high-dimensional and multi-class problems.

The proposed method offers a number of advantages. The first is the increased efficiency of the model’s capacity. When each hidden unit is focused on only capturing task-relevant patterns there are fewer units that are wasted on irrelevant patterns which means that the size of the model can be kept to a minimum. A simpler model has the benefits of being faster to train and also prevents overfitting. The second advantage is that there is less need to perform any feature selection. The proposed method can be seen as a form of feature selection for multi-class problems, but unlike traditional feature selection methods, the proposed model allows each input to have an impact of the learned features. This means that for multi-class problems inputs that are particularly good for discriminating between two classes, but might act as noise during other classes, do not need to be discarded to avoid the curse of dimensionality. With context-dependent selective attention the learned features of the model only captures the information from those inputs that are currently providing useful information.

The modification discussed in this chapter makes the auto-encoder particularly useful for unintuitive, complex, high-dimensional data sets that contain a lot of task-irrelevant information. The purpose of this idea is to change the model to fit all kinds of data instead of requiring altering the data to fit a certain model.
Chapter 5
Conclusions

In this chapter the challenges and contributions are summarized. Then a critical assessment of our approach is discussed followed by a presentation of the possible societal impact this work might have. Finally, some directions for future work are discussed.

5.1 Challenges and contributions

This thesis has focused on the problem of learning feature representations for multivariate time-series data using deep learning methods. The challenges of learning feature representations for time-series data have been reviewed and can be summarized in four points:

1. Many deep learning methods are developed for static data and are not capable of modeling the temporal structure in time-series data.

2. There is a lack of large benchmark time-series data sets that are suitable for evaluating and further developing deep learning algorithms for structured data.

3. Multivariate time-series data is often high-dimensional, complex, noisy and unintuitive and requires a feature extraction phase to reduce the dimensionality and separate the valuable information from the noise.

4. Multivariate time-series data may have signals that are redundant or task-irrelevant meaning that there is a need for feature selection.

In this thesis we have addressed these problems by training and providing algorithmic contributions to current deep learning algorithms for the purpose of building feature representations from raw unlabeled multivariate time-series data. More specifically, the contributions of this thesis are:
• We have investigated the challenges of modeling temporal data and reviewed recent development of deep learning methods for structured data. We have also identified the problems of applying static deep learning models to time-series data.

• We have introduced new challenging multivariate data sets to the deep learning community and outlined the prerequisites for making these data sets new benchmark sets that can further encourage development of deep network methods for temporal data, where progress have been slower then for deep learning methods for static data.

• We have shown that deep learning algorithms are capable of building useful feature representations from raw multivariate time-series data. We hope that this contribution inspires further experiments and development of algorithms that are capable of dealing with raw complex high-dimensional inputs. The performance was measured in terms of classification accuracy and to some extent by visual examination of the learned features. The motivation for using feature learning methods on raw data is that it removes the step of feature extraction and reduces the need for a priori knowledge about the data since the features are automatically constructed. This is especially beneficial for much of the unintuitive data that has been used in this work.

• We have made algorithmic contributions to one deep learning algorithm, namely the auto-encoder, making it more suitable for multivariate time-series. These modifications consisted of two parts: (1) the structure of the auto-encoder was changed to include connections from past hidden layers in order to capture the temporal information, and (2) context-dependent selective attention was introduced by weighting the reconstruction error for each input unit which made the learned features more focused on the task-relevant information in the data and reduced the need for feature selection.

5.2 Critical assessment

The overall objective of this thesis is to advance the field of deep learning specifically for time-series data. In this section we discuss some points about our approach towards this objective.

The first point concerns the word "deep" that is used throughout this work. The networks used in the experiments in this thesis are in fact not very deep. We used at most two layers of hidden layers. Such networks might still be able to be trained by regular backpropagation without using deep learning techniques such as greedy layer-wise pre-training. However, the field of deep learning covers more than requiring the data to be transformed through several non-linear transformations. In the early stages of deep learning it was believed
5.2. CRITICAL ASSESSMENT

that the use of several hidden layers would be the reason for the increased performance. However, deep learning uses unsupervised feature learning to initialize the network parameters in each layer and it turns out that this process is one of the key components for a successful application. Therefore, the focus in deep learning have shifted focus from building deep networks to the representation learning. Besides, in most applications there are diminishing returns on the actual gain in performance by adding more layers and the increase of training time have encouraged researchers to use no more than two or three hidden layers. Even though the name "deep learning" has stuck, there is a shift to use the more appropriate name "representation learning" which occurs more frequently in deep learning papers (such as [6]). One of the new conference for deep learning that was founded in 2014 is actually called "International Conference of Learning Representations" (ICLR).

The second point concerns the evaluation of our proposed model. We used the classification accuracy on a number of data sets and compared it to other methods as the performance measure. Such comparisons can be misleading for a number of reasons: (1) the training of the algorithm that was compared to was not as thoroughly optimized, (2) the data set that was used is more suitable for one of the methods, and (3) the human-reachable performance of the used data set is not mentioned or unknown.

The first issue is best countered by comparing the results on the same data set from other independent research groups. It has been a challenge to find such results due to the previously mentioned lack of benchmark time-series data sets for deep learning research. In this thesis the only comparison with other research groups was on the static data sets of handwritten digit recognition and object recognition where our method was able to improve the classification accuracy by a few percentage points compared to other methods. The issue of the second point concerns the choice of data set for the performance comparison. The proposed method of selective attention works best on high-dimensional data sets that contain much task-irrelevant information and preferable with high inter-variable multi-class problems. The focus in this work has not been to perform any data set selection to find a data set that gives the best comparative performance. Finally, the last point concerns the human-reachable optimal performance of the data set that is used. For example in sleep staging there is an inter-rater reliability of 80% [62], which means that if one algorithm is close the reachable optimal performance, a better algorithm will only slightly improve the accuracy. Without knowing this upper bound on the accuracy for the current problem it can be misleading when comparing the performance of two algorithms.

A challenge in representation learning for both static and temporal data is the issue of robustness and invariance. The need for learning invariant features depends on the nature of the data set. For static data such as images it can be assumed that translation and scale invariance (and to some degree rotation invariance) is very important. This means that the weighting vector for static
data should also be invariant. Convolutional neural networks are capable of learning invariant features which explains the recent success of such models applied to computer vision tasks. The development of an invariant weighting vector for convolution neural nets and static data is outside the scope of this thesis. The experiments of applying the idea of a weighting vector on static data was to illustrate the usefulness for other domains as well.

For time-series data, the need for more invariant features is task-dependent. For multivariate time-series there is the added challenge that some signals are slightly shifted in time and it depends on the task if the learned features should be invariant to such translations. Just like deep learning algorithms for static data, more research is needed for learning robust and invariant feature representations for time-series data. This research should begin with uni-variate time-series. However, this has not been the focus of this thesis since this work focuses on the problem of selective attention for multivariate time-series.

5.3 Societal impact

We see the contributions in this thesis might have a number of impacts on society. This section discusses how the findings in this work could affect both research in academia, business in the industry, and economics in society.

The goal of artificial intelligence (AI) is to building intelligent machines. While traditional AI methods have resulted in have shown some impressive applications, the models that these programs reason about have to be encoded manually. The field of machine learning emerged from AI, statistics, and pattern recognition with the same goal but with the approach that the models are trained from data instead of being programmed. However, a large part of machine learning research is still focused on how a priori knowledge can be infused into the learning algorithm by using cleverly designed and hand-picked features. This approach has solved many tasks that traditional AI techniques was unable to solve. We hope that our work can inspire a move towards a focus on developing more generalized methods that are less dependent on human engineers and are capable of handling complex, unintuitive, and high-dimensional time-series data.

This is especially important for the industry which sees a number of emerging applications with new sensor technologies. For example, the development of electronic noses are have lead to new medical applications and better and faster food quality measurements. These amount of new applications have made it intractable to provide hand-made application-specific features. New sensor technologies are often designed with a particular task in mind and need to be evaluated in terms of selectivity and discriminative power. The challenge is that there is a lack of domain expert knowledge whenever a new sensor is developed. We therefore see how more research on representation learning methods for time-series data can facilitate the evaluation and development of new sensor technologies.
The use of selective attention also allows for applications that require more information and have a long-term time dependency. In economics, stock-market prediction is one such application where the stock price data can be combined with relevant additional information to achieve better stock price predictions [1]. Previously such attempts have not been successful because of the sheer amount of information. With selective attention there could be a breakthrough in how stock-market prediction is performed because one need not to worry about feeding the algorithm too much information and suffer from the curse of dimensionality.

5.4 Future work

Deep learning is a relatively new and very active research area that is constantly evolving. Since research on deep learning methods for structured data has had a slower progress than for static data and there is a growing interest to apply deep learning to time-series problems there is a need to further develop methods for time-series data. This thesis has focused on taking steps towards a general solution for dealing with multivariate time-series. There are possible extension to this work that could be explored but there are also other possibilities that can be explored.

An interesting future direction for the idea of selective attention is to merge it with other representation learning algorithms, e.g., convolution neural nets, RBMs, and denoising auto-encoders. It would also be interesting to see how selective attention would work together with other popular new techniques such as dropout and rectified linear units. New ways of setting the weighting vector should be explored. In this work we have used two methods for setting the weighting vector: fixed and adaptive. For the fixed method a feature selection method that uses multi-class variable ranking criteria such as Fisher’s criterion [12] could be considered and for the adaptive method other choices of weighting penalty terms that depend on more factors than just the weighing vector should be explored. An interesting direction for future work would be to develop a method that combines the two methods into a general single method. For static data a possible future direction is to use convolution neural networks and develop an invariant weighting vector.

For automatic sleep staging, a future direction is to continue the research on developing invariant feature representations from raw input data. The problem could be divided by using one model that learns invariant features for each signal and then combine the output into a higher-level model that captures the correlations between the signals.

For e-nose data, a future direction is to find methods to collect a large amount of meaningful data in an open sampling environment and then apply deep learning methods to it in order to automatically find interesting patterns. Here, the focus need to be on noise reduction and learning features that are invariant to long-term sensor bias.
References


[95] Lukáš Zoubek, Sylvie Charbonnier, Suzanne Lesecq, Alain Buguet, and Florian Chapotot. A two-steps sleep/wake stages classifier taking into account artefacts in the polysomnographic signals. In Proc. of the 17th World Congress of The International Federation of Automatic Control (IFAC), 2008. (Cited on page 16.)