Deep generative models for semi-supervised motion classification

Jonathan Richard Schwarz

School of Informatics
The University of Edinburgh

This dissertation is submitted for the degree of
Master of Science

August 2016

Declaration

I declare that this tehsis was composed by myself, that the work contained herein is my own excpt where explicitly state otherwise in the test, and that this work has not been submitted for any other degree or prefessional except as specified.

Jonathan Richard Schwarz
August 2016

Acknowledgements

During the course of this thesis, I have been supported by numerous people. I would like to express my sincerest gratiude to:

- Taku Komura, for his instructive supervision. His helpful comments and advice have helped me to more clearly formulate my research and understand which ideas are worth pursuing.
- Daniel Holden, for helping out with any questions and practical issues encountered. His expertise provided important insight into the problem.
- Iain Murray, Amos Storkey and Charles Sutton for helpful advice which helped me to identify and apply the correct methods to the task. Special thanks for all the answers to the numerous Machine Learning questions.
- My family, for supporting me throughout my life and always being there. This can never be valued highly enough.
- All my friends who made my studies a fun and intellectually challenging experience. Knowledge is the only good which increases when shared.
- My labmates: Abie, Joe, Georgi and others. For many great moments, mutual support and interesting conversations.

Abstract

The automated analysis of human motion as recorded by motion capture (MOCAP) systems is an important tasks in areas as diverse as medicine, robotics and computer animation. However, the acquisition of high-quality, labelled motion data involves considerable cost and human labour. This is particularly true in medicine, where subjects often need to meet many conditions and the labelling has to performed by expert staff.

Both the lack of large supervised data sets as well as the complexity of the data have long limited the success of machine learning systems in motion analysis. As motion data is typically represented as a time-series of joint angles and positions, it is difficult to put forward feature sets of the data which are well suited for subsequent classification. While advances in deep neural networks have significantly reduced the need for feature engineering, many of the previously unmatched results in fields such as speech and vision have only been possible through the availability of large labelled datasets.

Instead, in the work at hand, we investigate the use of novel methods proposed to train deep neural networks in a semi-supervised/unsupervised fashion. This will allow us to take advantage of the representational power of such models, without requiring the creation of large labelled data sets. We apply, evaluate and extend previously proposed techniques put forward for the task of pattern-recognition in motion sequences.

We show through experimental results, that the systems discussed in this thesis are capable of accurately and reliably producing correct labels for sequences. As distinct from traditional approaches, we show how unlabelled motion data can be used to increase generalisation on unseen motion sequences. As part of this thesis, we put forward a novel learning method which automatically learns expressive representations by utilising the statistical strength of deep generative models. Our algorithm outperforms the previous state of the art, achieving a test-set classification accuracy of 88%, an improvement of 2.36%.

Table of contents

List of figures

List of tables

Nomenclature

Roman Symbols

$\mathbf{A}, \mathbf{B}$	Matrices
$\mathbf{a}, \mathbf{b}$	Vectors
$\mathbf{b}$	Bias
$\mathbf{h}$	Hidden variables (in deterministic networks)
$\mathbf{I}$	The identity matrix
$\mathbf{W}$	Weights
$\mathbf{x}$	Input
$\hat{\mathbf{x}}$	Reconstructions
$\tilde{\mathbf{x}}$	Corrupted version of $\mathbf{x}$
$\mathbf{y}$	Output/Targets (in supervised learning)
$\mathbf{z}$	Latent variables (in generative networks)

Superscripts

$x^{(i)}$	Data point i

Other Symbols

$\mathbb{E}_p[X]$	Expectation of random variable X under distribution p
$\mathcal{D}_{KL}(p\|\|q)$	Kullback–Leibler divergence between distributions p and q
$\log(x)$	The natural logarithm

$\mathcal{L}$	Loss
$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$	The Gaussian (normal) distribution with mean $\boldsymbol{\mu}$ and covariance $\boldsymbol{\Sigma}$
`ReLu`	Rectifier activation function: $f(x) = max(0, x)$
θ, ϕ	Parameters of a model

Acronyms / Abbreviations

CNN	Convolutional neural network
LSTM	Long short-term memory
MLP	Multi-Layer Perceptron
MOCAP	Motion capture
SGD	Stochastic gradient descent
TSVM	Transductive support vector machine
VAE	Variational Autoencoder

Chapter 1

Introduction

Data captured by motion capture (MOCAP) systems has become a standard format for digital processing of human motion. Due to the importance of such data in fields as diverse as animation, robotics, sports, medicine and others, the automated recognition of patterns in MOCAP data has been an active area of research in recent years (e.g. Harvey and Pal [26], Chen and Koskela [12], Müller and Röder [51]). This has proven to be a difficult task, primarily due to:

(i) The lack of publicly available large-scale labelled data sets. This makes it difficult to apply models with sufficient statistical strength capable of disentangling causal factors of the observations at hand.

(ii) Feature extraction: As it has been proven difficult to apply existing pattern recognition algorithms to raw motion data, the definition of hand-crafted features has been a popular choice. Such approaches however, require extensive expertise in the problem domain and are expensive and difficult to acquire.

In this thesis, we will re-visit the problem of human motion classification by proposing methods to address the aforementioned problems. We will do so by explicitly proposing to learn useful representations of motion data by utilising existing unlabelled databases. We show how a representation learning approach enables the application of powerful and flexible learning algorithms.

1.1 Motion classification

The problem of motion classification corresponds to the automated recognition of particular patterns in motion data and their assignment to a pre-defined set of classes.

In most practical application, motion classification systems are required to recognise both subtle differences between similar classes while simultaneously distinguishing between logically unrelated actions. This could, for instance, be an actor rotating both arms either forward or backward. Clearly, compared to an actor performing a cartwheel or simply walking, the difference between these classes is much harder to recognise. A suitable classification algorithm must hence be capable of processing coarse as well as fine-grained features. This makes the manual definition of features particularly difficult, as the subtle differences between motions are usually difficult to encode. Classification algorithms thus tend to miss-classify such fine-grained classes.

Furthermore, the lack of important properties in motion data makes it difficult to apply pattern recognition algorithms to raw data. An example is the missing property of *smoothness*, the assumption that numerical proximity corresponds to semantic similarity, i.e. to similar motions or actions in a sequence. Consider, for example, two actors walking in the same direction at different pace. Even though the motion is logically equivalent, the difference in velocity will result in large numerical distance. This makes it difficult to apply machine learning and pattern recognition without preceding feature extraction. More formally, the underlying non-linear function $f : X \rightarrow Y$ which maps from the input or feature space X to labels or classes Y will assume that data points $(\mathbf{x}^{(i)}, y = k), (\mathbf{x}^{(j)}, y = k)$ of class k are close in feature space. Hence, the function must not be steep at any point. As smoothness is a fundamental assumption of the vast majority of pattern-recognition/machine learning algorithms (e.g. k-means (MacQueen et al. [50]) or k-nearest neighbours (Altman [1])), the extraction of robust features that satisfy this property poses a a main challenge in motion classification.

A range of previous approaches to motion classification have attempted to base subsequent classification on hand crafted features. Examples of proposed representations include both binary and real-valued features derived from geometric relations between joints (e.g. Müller et al. [52]). Other feature extraction methods are based on representing motions as sequences of cluster centroids (Liu et al. [47]). A common problem of such methods is the difficulty of defining appropriate features that allow for accurate and robust recognition across tasks. The quality of hand-engineered features is highly dependent on the particular classification task, as well as the nature of the data. Consequently, it is often non-trivial to apply these methods to new data sets considering the vast variety of human motion. It is for instance, unrealistic to claim

that features shown to work well on motion data recorded for film making purposes will be adequate for gait classification in health.

Instead, we advocate the application of models capable of learning complex data representations tailored for the task at hand. Not only has this been shown to improve generalisation in many domains, it also significantly eases the application to new sources of data. Moreover, many of the proposed feature extraction algorithms are computationally cumbersome during both training and test time. Once trained, on the other hand, the methods proposed as part of this thesis are fast and efficient.

Finally, the task of motion recognition is inherently a time-series problem, an important consideration regarding the choice of classification algorithm. Hence, the learning algorithm chosen must be capable of explicitly incorporating the temporal aspect of the data at hand. We will motivate our choice of models regarding this property.

1.2 Contributions

This thesis formulates and critically evaluates several hypotheses. We introduce various new architectures and learning techniques in the context of motion classification. This thesis constitutes a contribution in the following respects:

- The introduction of a novel learning framework for motion classification. We benchmark our models and achieve an improvement of the current state-of-the art on the challenging HDM05 (Müller et al. [53]) data set. Our proposed framework is simple to implement and can be trained efficiently.[1]
- We investigate the applications of deep generative models to motion classification. To the best of our knowledge, this is the first time such models have been applied to the task.
- The application of convolution neural networks (CNNs) to motion analysis. While previous approaches have mainly favoured recurrent architectures, we show how CNNs are well suited for the task.
- The investigation and evaluation of multiple models suited for semi-supervised learning. We extend previously introduced architectures and show how their performance can be improved by adding unlabelled data to the training process.

[1] We achieve state-of-the-art results in less than 3 minutes on a Nvidia Geforce Titan GPU.

1.3 Overview of this thesis

We begin by providing the reader with the relevant background in Chapter 2. While we do assume familiarity with machine learning concepts and the basics of neural networks, we review several common architectures. Furthermore, we give an introduction into semi-supervised and representation learning, the main paradigms of this thesis. We will motivate these methods and elaborate their relevance to the detection of patterns in human motion. In Chapter 3, we review and critically discuss recent research in motion classification. This will allow us to formulate a baseline for the experiments carried out. Chapter 4 will motivate the application of latent variable models to the problem. We will briefly review latent variable models and thereafter introduce the variational autoencoder, a generative model which allows us to combine the ideas of neural networks with latent variable models. Such architectures are known as deep generative models and will form the core component of the final model proposed in this thesis. We will give its formulation and show how it can be applied to human motion by proposing a novel learning architecture. In Chapter 5, we critically evaluate the hypotheses stated. We benchmark several methods discussed on a publicly available data set and give a separate empirical evaluation for each of the hypotheses stated. Finally, in Chapter 6, we discuss both the advantages and shortcomings of the methods proposed. We draw conclusions and give concrete suggestions for future research.

Chapter 2

Background

In this chapter, we will provide the reader with the relevant background necessary to understand the work at hand. We will introduce the fundamental concept of semi-supervised representation learning and motivate its applications to motion classification.

In addition, we introduce and motivate a selection of existing algorithms which will be evaluated later in this thesis. Note that many of the reviewed architectures form a fundamental building block for more sophisticated methods. It is thus crucial to understand their definition and motivation.

In general, we assume familiarity with basic concepts in Machine Learning, which can be reviewed in (Bishop [7]). Furthermore, the scope of this thesis only allows a brief discussion of some of the methods applied to motion classification. The interested reader is advised to conduct (Bengio and Courville [4]) for a more comprehensive discussion.

2.1 Motion capture data

Motion capture data is typically stored as a time-series of joint position and angles in 3D-space according to a *kinematic chain*. A kinematic chain is a simplified model of the human skeleton which defines individual joints of interest as well as the connection among particular joints. During the recording of an action sequence, markers are attached at the respective position on the actor's limbs. This allows the tracking of each marker through infra-red cameras positioned at different angles to the actor. Figure 2.1 shows an actor in a motion capture recording. The white markers are visible on a suit (black), which an actor is to wear during a recording. Commercial

MOCAP systems typically come with specialised software allowing the reconstruction and post-processing of the kinematic chain from the sensor data.

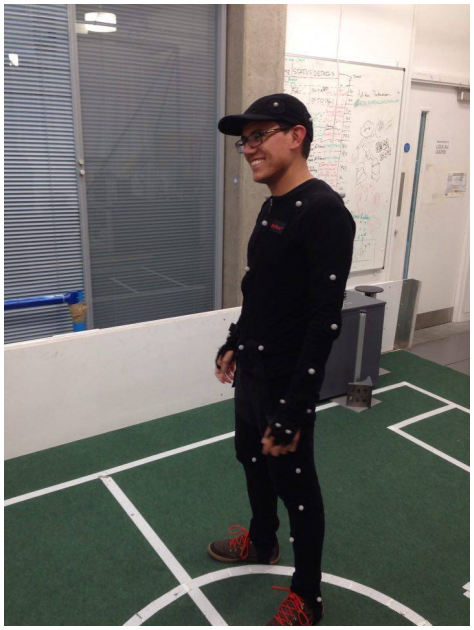


Fig. 2.1 An actor in a motion capture studio. Makers (white) on the black suit allow the tracking of joints with infra-red cameras. Courtesy of Nestor Lobato Garcia.

As an alternative source of human motion data, RGB cameras with depth sensors (RGB-D) have become increasingly popular due to their low cost in comparison to MOCAP systems (Cho and Chen [14]). The high portability of RGB-D systems, such as the Microsoft Kinect, make this a popular choice in computer vision and robotics. The additional depth information in comparison to mere RGB video data make the extraction of the human skeleton considerably easier (Shotton et al. [61], Xia et al. [68]) and allow the acquisition of motion data in real-time. In order to ease the comparison to previous approaches and due to the lack of publicly available RGB-D data sets, we will benchmark models proposed as part of this thesis on MOCAP data only. We stress however, that the models and methods proposed are suitable for a vast range of different pattern recognition problems. The application to motion data as captured by RGB-D sensors should thus be conceptually simple given sufficient data.

2.2 Semi-supervised learning

We will explicitly approach the motion classification task as a problem in which we aim to utilise the availability of unlabelled data. This problem is usually referred to as

semi-supervised learning in the machine learning literature. Semi-supervised learning describes a scenario where in addition to labelled data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$, the learning algorithm is to harness the availability of unlabelled data $\{\mathbf{x}^{(j)}\}_{j=1}^{N}$ in order to improve the performance on a given task. Typically, the amount of unlabelled data points is significantly larger (that is, $N >> M$). Thus, we can evaluate the success of a semi-supervised approach by comparing it to the performance of a purely supervised learner on the same task.

Popular approaches to semi-supervised learning aim at optimising the decision boundaries between classes using unlabelled data. The transductive support-vector machine (TSVM, Joachims [37]) combines the maximum-margin objective of a regular SVM with the goal of placing as few unlabelled data points near the margin as possible. Other examples aim at constructing a similarity graph between observations. An example of such a graph-based approach is the label propagation algorithm (Raghavan et al. [58]). We show its application to a synthetic data set in Figure 2.2. Shown on top is the original, synthesised data set consisting of positive (red) and negative observations (blue). The resulting decision boundary is shown on the top-right. Colour strength corresponds to the confidence of the classifier. On the bottom, we show the same figures after adding additional unlabelled data (in grey). It can be clearly seen how the decision boundary is modified in favour of the observations in blue.

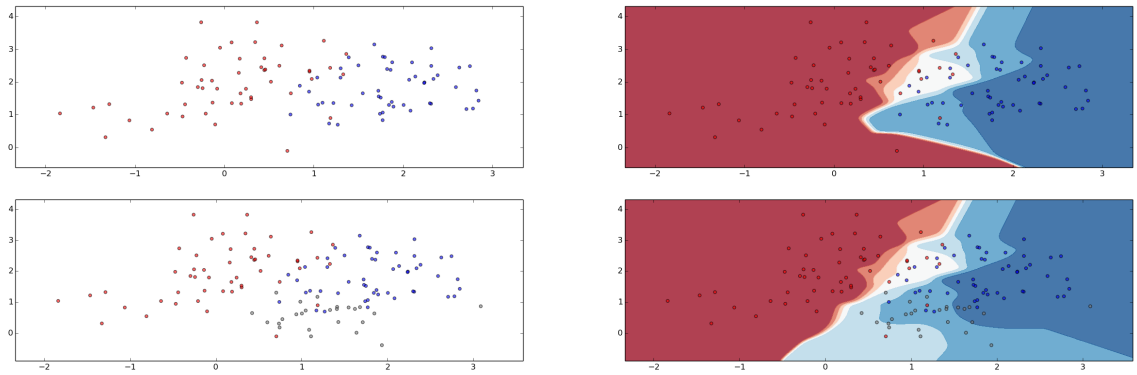


Fig. 2.2 An application of label propagation to a semi-supervised classification task. Top: Original (synthesised) data and the resulting decision boundary of a supervised classifier. Bottom: The boundary changes after unlabelled data is added. Figures produced using code published as part of the scikit-learn library [56].

However, these methods suffer from an important drawback: Their incapability of learning representations from raw data. In cases where the assumption of smoothness

does not hold, their application requires a preceding feature extraction algorithm. This feature extraction algorithm must be able to learn a transformation optimally suitable for the subsequent semi-supervised classifier. Thus, we argue that the main challenge in semi-supervised motion analysis is learning transformation which lead to such representations, rather than the optimisation of decision boundaries.

Therefore, we will explore the applicability of neural network architectures trained in a semi-supervised fashion. The main advantage of this paradigm is the ability of neural networks to automatically learn a representation suitable for the classification algorithm.

To avoid confusion, for the rest of this thesis, we will refer to semi-supervised learning as the aforementioned scenario in which we have access to labelled and unlabelled observations. This in contrast to other uses of the term which describe the simultaneous optimisation of supervised and unsupervised objectives. We refer to the second case as Multi-task learning, which we will review in section (2.6).

2.3 Representation learning

Recent technological and theoretical breakthroughs in the training of deep neural networks have led to remarkable results on complicated, high dimensional data. Among many other domains, their application has been particularly successful in object recognition (e.g. on the ImageNet data set (Deng et al. [16])) or phoneme labelling in speech processing (Graves and Schmidhuber [25]). Besides technological advances and renewed interest in the field, this has been possible largely due to the existence of large-scale labelled data sets.

In many domains of interest however, the considerable cost of creating such data sets has prohibited the application of high-capacity models trained in a supervised fashion. This is also the case for motion classification, where in contrast to natural images for instance, the creation of merely small data sets requires special hardware, trained staff and significantly more time.

It is important to understand that a neural network trained for classification merely implements a range of several non-linear transformations which serve as the input of a linear classifier. Thus, these non-linear transformations must map the raw input data in a space where instances of particular classes are linearly separable (typically by a

softmax-regression classifier). We can think of training as the process of both learning such transformations as well as fine-tuning the linear classifier which is applied on top of previous transformations. The process of learning transformation in such an alternative space is referred to as *representation learning*.

In domains where labelled data is only available is small amounts, the task of representation learning is inherently linked to the problem of unsupervised learning. This raises the questions, how models capable of learning useful representations in the absence of supervised learning targets can be put forward. While this problem has been in the focus of Deep Learning research for many years (e.g. (Hinton and Salakhutdinov [29], Bengio [5], Radford et al. [57])), it is still unclear which method (if at all) is applicable to the task at hand.

Due to the availability of both labelled and unlabelled motion data from several small-medium sized data sets, we will adopt semi-supervised representation learning as the main paradigm in the context of this thesis. Due to their statistical strength, we will consider this in the context of deep neural networks. We show that these techniques provide a means of tackling the previously discussed challenges in motion classification.

While measuring the qualitative difference of one representation over another is an open problem in general, we can usually argue in terms of the performance metric for a subsequent task, if such a task is present. As we are fundamentally concerned with the recognition of human motion, this allows us to adopt classification metrics to evaluate the success of representation learning in the context of this thesis.

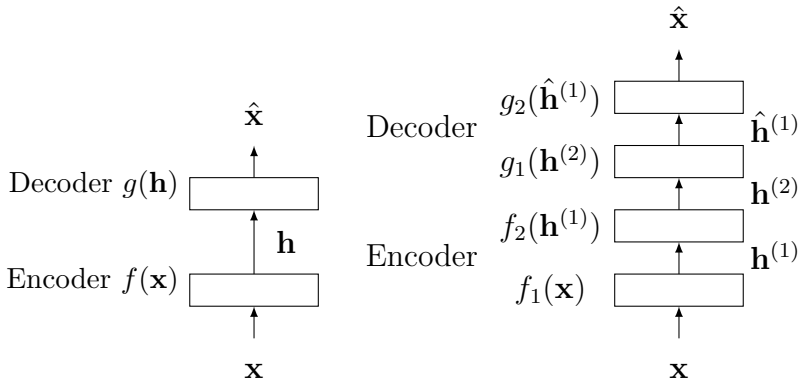
2.4 Autoencoders

As a first means to representation learning, we will consider the autoencoder, a neural network optimised to reconstruct its input. As this simple training objective does not require any other input apart from the data itself, it forms the basis of a simple and flexible *unsupervised* learning algorithm. By constraining the way in which autoencoders are to achieve this objective, they can for instance be used as a means to dimensionality reduction and feature extraction (e.g. Hinton and Zemel [30]). In the context of semi-supervised learning, features learnt during the unsupervised training of autoencoders can thereafter be optimised to help solve the subsequent supervised

problem. Thus, unlabelled examples directly contribute to the training process by enabling the model to learn more robust and expressive features.

While autoencoders have been known as a standard Deep Learning technique for decades (Le Cun [45], Bourlard and Kamp [9]), recent breakthroughs in their training (Vincent et al. [67]) and their combination with generative models (Kingma and Welling [40], Kingma et al. [39]) make them a powerful choice for unsupervised/semi-supervised learning. Thus, they constitute a main paradigm for the remainder of this thesis.

When specifying the architecture of an autoencoder, we distinguish between encoding and decoding functions $\mathbf{h} = f(\mathbf{x})$ and $\hat{\mathbf{x}} = g(\mathbf{h})$ respectively. Here, $\mathbf{x}$ is the input, $\hat{\mathbf{x}}$ the reconstruction of $\mathbf{x}$ and $\mathbf{h}$ are hidden features. After training, we are typically only interested in the parameters of the encoder, since the learnt projection from $\mathbf{x}$ to $\mathbf{h}$ can help to disclose interesting properties of the observations. The autoencoder architecture is illustrated in Figure 2.3. Note also the generalisation to a deep autoencoder, which simply denotes the application of several encoding/decoding function $f_1, \ldots, f_n, g_1, \ldots, g_n$. This can lead to more abstract features at each layer.



(a) Autoencoder (b) Deep Autoencoder

Fig. 2.3 An illustration of an autoencoder (a) and its extension to arbitrarily deep models (b). f, g are encoding and decoding functions, respectively. Through the subsequent applications of these functions, the autoencoder is to reconstruct its input. This leads to several hidden representations $\mathbf{h}$ which may disclose interesting properties of the data.

Encoding and decoding functions f, g must not follow any particular constraints, which allows the combination of the idea behind autoencoders with many other deep learning techniques. Common choices for f, g are affine transformations $\mathbf{W}\mathbf{x} + \mathbf{b}$ and convolution operations. Here, we denote the weights of a transformation with $\mathbf{W}$,

while $\mathbf{b}$ are bias parameters. In practice, these transformations are usually combined with a non-linearity such as the sigmoid or tanh function. Training of an autoencoder can as usual be achieved by optimising a custom cost function through application of the Backpropagation algorithm (Rumelhart et al. [60]). A simple default choice of an optimisation objective $\mathcal{L}(\theta)$ with respect to parameters $\theta = \{\mathbf{W}, \mathbf{b}\}$ is the mean-squared error between inputs and reconstructions:

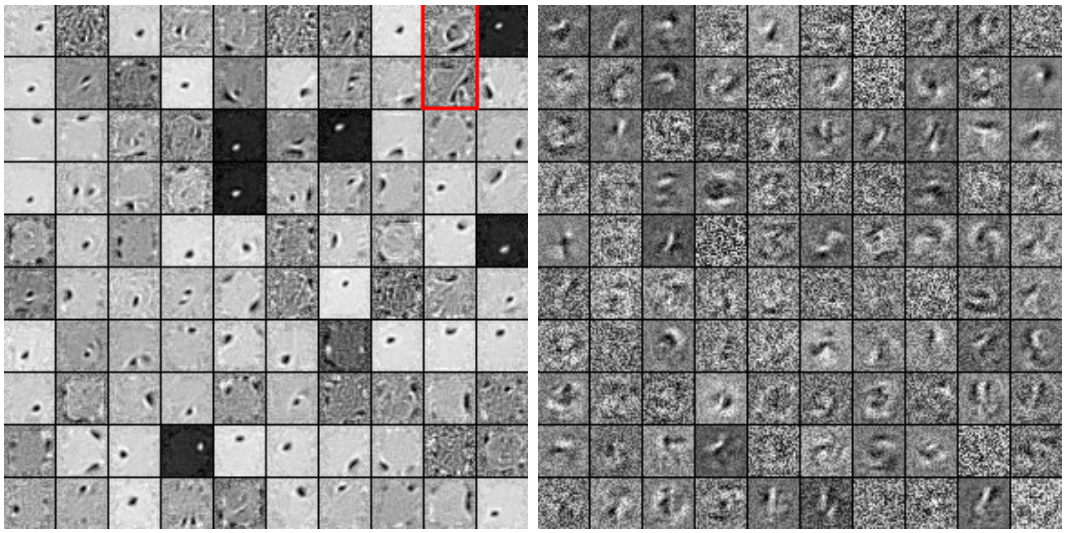
$$\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} ||\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}||_2^2 = \frac{1}{m} \sum_{i=1}^{m} ||\mathbf{x}^{(i)} - g(f(\mathbf{x}^{(i)}))||_2^2 \tag{2.1}$$

Thus, the autoencoder constitutes a neural network trainable in a fully unsupervised fashion.

In the case of a simple affine transformation, training of an autoencoders is in fact equivalent to Principal Component analysis (PCA) (Pearson [55]), i.e. the model learns a projection into a linear subspace. The addition of non-linearities in the encoding and decoding process thus serves an important purpose: They empower autoencoders to learn more expressive features in a non-linear subspace.

In practice, it is important to impose constraints on either the architecture or the learning process. This prevents the autoencoder from simply learning the identity function, i.e. copying its input. A simple means of doing is a constraint on the dimensionality of the hidden space: $dim(\mathbf{h}) < dim(\mathbf{x})$. Clearly, by limiting the capacity of the hidden space, the network is no longer able to merely copy the input, but must instead learn a *lower-dimensional manifold* of the data.

While the motivation for such so-called *undercomplete* architectures is self-explanatory for dimensionality-reduction, we argue that it is equally important in feature learning. In general, we assume that a significantly smaller number of causal factors underlies a particular observation. Such underlying factors in motion data may, for instance, correspond to the pace or style in which a particular action is carried out. Undercomplete architectures enforce the learning of only a handful of features which must hold enough information for the data to be reconstructed. If this form of unsupervised feature learning can be applied successful, the hidden features ought to correspond to the most salient aspects of the data.



(a) Denoising autoencoder (b) Supervised MLP

Fig. 2.4 Feature detectors learnt by unsupervised and supervised approaches on the MNIST data set. The supervised method achieved a good test set accuracy of 98.1%. Figures produced using code that was released as part of the Deep Learning tutorials.[a]

[a]http://deeplearning.net/tutorial/

We compare the features learnt by a supervised Multi-Layer Perceptron (MLP) with those resulting from unsupervised autoencoder training on the popular MNIST data set in Figure 2.4. Note that although the supervised learner had access to class information, the features are difficult to interpret. While this is also true for most features of the unsupervised learner, some of the detectors appear to be sensitive to particular digits. Features on the top right (highlighted in red), for instance, appear similar to digits '6' and '7'.

The learning of features in an unsupervised fashion makes autoencoders one of the simplest neural network approaches to semi-supervised learning. This can be achieved by initialising the weights of a neural network designed for supervised learning with the parameters learnt during unsupervised learning. If the unsupervised training of the autoencoder (the *pre-training step*) succeeds in learning features relevant for the subsequent supervised task, the MLP can *fine-tune* these feature detectors to maximise their usefulness for the supervised task. This can also be interpreted as initialising the supervised model in a more promising area of the high-dimensional parameter space. Thus, such methods can, in theory, lead to both a faster learning process as well as an improvement in generalisation.

2.4.1 Sparse autoencoders

As an alternative to undercomplete models, we can add add regularising objectives to the loss functions. This leads to the notion of sparse autoencoders, who are optimised by adding ℓ^1 (lasso) regularisation (Tibshirani [66]) to the loss function in equation 2.1. In addition to preventing learning of the identity, the ℓ^1 penalty allows us to explicitly encourage sparse representations (hence the name) as a training objective. Thus, only few features must capture sufficient information for the network to allow for reconstruction. Sparse representations are desirable, as they help to disentangle causal factors (Bengio et al. [6]), one of the main goals in representation learning. This makes sparse autoencoders typically more suitable for subsequent classification. The modified loss function looks as follows:

$$\mathcal{L}(\theta) = \frac{1}{m} \sum_{i=1}^{m} ||\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}||_2^2 + \lambda ||\mathbf{W}||_1 \tag{2.2}$$

where λ a constant used to trade-off between the cost functions. It is typically set on a held-out validation set.

In theory, sparse autoencoders allow for latent spaces of higher dimensionality, since learning of the identity will not allow the reduction of the regularisation cost. However, following our previous argumentation, if hidden features are truly expressive, only a relatively small dimensional space ought to suffice in explaining the data.

2.4.2 Stacked denoising autoencoders

The formulation of the learning process as a denoising problem is yet another method which may be applied to prevent the model from learning the identity function. This is done as follows: Obtain a corrupted version $\widetilde{\mathbf{x}}$ of the input by adding noise to a data point $\mathbf{x}$. Subsequently, the network is to optimise the mean-squared error between the reconstructed version and the original (non-corrupted) input. Not only does this prevent the network from learning the identity function, but can also be argued to lead to the learning of more robust representations (due to the denoising objective).

It is important to note that the type of noise added to a training example can be crucial for good performance. In general, we aim to add noise that is as close as possible to variations found in real-world data. In motion analysis and computer vision for instance, Gaussian noise does often not lead to the desired results. In such

cases it can be useful to draw a random variable from a binomial distribution for each feature in the observation. This corresponds to randomly discarding some of the input information, i.e. setting 3D positions of random joints in a time sequence of MOCAP data to zero. This has for instance been done by (Holden et al. [33]) and can also be helpful to deal with noise encountered in real-world data. Nevertheless, for the rest of the section we will assume Gaussian noise to ease notation.

Denoising autoencoders have proven particularly useful when combined with the idea of stacked autoencoders (Vincent et al. [67]). Stacked autoencoders correspond to the individual training of single layers in a deep network as denoising autoencoders. Stacked Autoencoders are designed for learning of subsequently more abstract features of latent variables in an unsupervised fashion. In comparison to deep autoencoder, this has been empirically shown to often lead to improved generalisation in classification tasks (Erhan et al. [21], Larochelle et al. [43]).

Subsequently more abstract feature learning can be achieved by separately training a layer, and using its representation as the input to training of a subsequent layer. This is known as *greedy layerwise unsupervised pre-training* and has has been one of the first methods allowing the training of deep neural networks. It is also used in other unsupervised training schemes. An example is the forming of deep belief nets (Hinton and Salakhutdinov [29]) by individually training Restricted Boltzmann Machines (Smolensky [62]).

Figure 2.5 shows how a stacked denoising autoencoder is obtained by re-using learnt representations and training a single layer at each step during training. In steps (a)-(c) both unlabelled and labelled data points are being used during training. (a) Initially, the network merely learns to denoise the original training examples. The reconstructed input will constitute the first layer of latent features. (b)-(c) Subsequently, each layer l is trained to reconstruct its own input, i.e. the output of layer $l-1$. Step by step, layers are stacked on top of those already trained. (d) Finally, the final layer is trained as a linear classifier. This is known as the fine-tuning phase. Hence, only labelled examples are used during this step.

We give an algorithmic formulation of the training process for a semi-supervised classification task in Algorithm 2.1. The conditional class probability is estimated through the application of a softmax activation. As a supervised cost, we assume the

categorical cross-entropy. While we show the training process for an MLP, the affine transformation can simply be replaced with other common transformations without loss of generality.

In terms of semi-supervised learning, such models can be utilised in the same way pre-training/fine-tuning scheme as normal autoencoders.

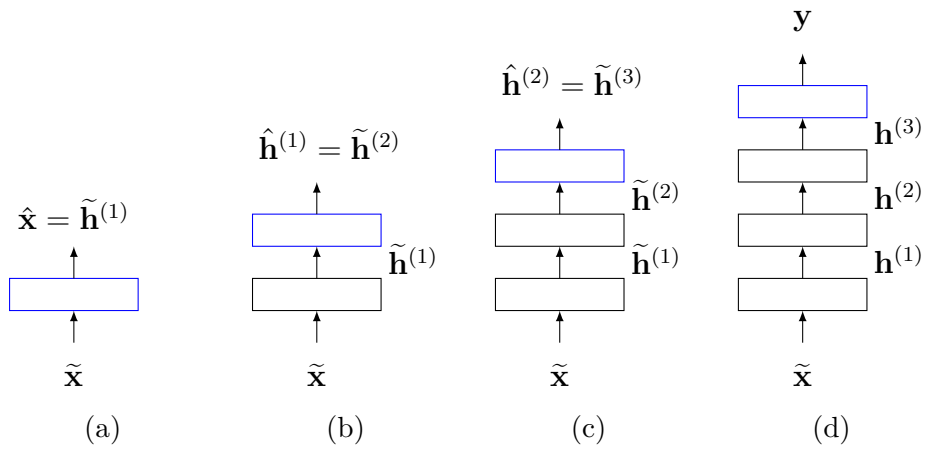


Fig. 2.5 The training procedure of a stacked denoising autoencoder. At each step during training, only the parameters of the layers indicated by blue colour are updated.

2.5 Convolutional neural networks

2.5.1 The convolution operation

Convolutional neural networks (CNNs, Fukushima [22], LeCun et al. [46]) implement the mathematical convolutional operation and can be perceived as a means to explicitly account for the time-series property of many sequential tasks. CNNs have been in the forefront of deep learning research leading to remarkable results in a vast variety of pattern recognition problems (e.g. Krizhevsky et al. [41]). This makes CNNs an interesting technique for recognising patterns in human motion.

Their applicability to motion data has been previously studied in (Holden et al. [33]), where CNNs used as part of a denoising autoencoder framework, trained to learn a manifold of human motion. This can for instance be useful when a corrupted MOCAP sequence (in case of unreliable sensors) is to be denoised.

Algorithm 2.1: Training of stacked denoising autoencoders for a semi-supervised classification task. Here SGD = Stochastic gradient descent.

Input : Labelled data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$, Unlabelled data $\{\mathbf{x}^{(j)}\}_{j=1}^{N}$, Parameter $\theta = \{\mathbf{W}_1, \ldots, \mathbf{W}_k, \mathbf{b}_1, \ldots, \mathbf{b}_k\}$, Noise covariance Σ

Output : Trained Parameters θ

begin

  `// Unsupervised pre-training`

  for $\mathbf{x} \in (\{\mathbf{x}^{(i)}\}_{i=1}^{M} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N})$ **do**

    `// Add noise to the input`

    $\widetilde{\mathbf{x}} \sim \mathcal{N}(\mathbf{x}, \Sigma)$

    $\widetilde{\mathbf{h}}^{(1)} \leftarrow \widetilde{\mathbf{x}}$

    for l in $[2, \ldots, k]$ **do**

      $\hat{\mathbf{h}}^{(l-1)} \leftarrow \mathbf{W}_{l-1}\widetilde{\mathbf{h}}^{(l-1)} + \mathbf{b}_{l-1}$

      $\mathcal{L}(\{\mathbf{W}_{l-1}, \mathbf{b}_{l-1}\}) = ||\widetilde{\mathbf{h}}^{(l-1)} - \hat{\mathbf{h}}^{(l-1)}||_2^2$

      Optimise $\{\mathbf{W}_{l-1}, \mathbf{b}_{l-1}\}$ by minimising $\mathcal{L}(\{\mathbf{W}_{l-1}, \mathbf{b}_{l-1}\})$ via SGD.

      $\widetilde{\mathbf{h}}^{(l)} \leftarrow$`activation`$(\mathbf{W}_{l-1}\widetilde{\mathbf{h}}^{(l-1)} + \mathbf{b}_{l-1})$

  `// Supervised fine-tuning`

  for $\mathbf{x}, \mathbf{y} \in \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{M}$ **do**

    $\mathbf{h}^{(1)} \leftarrow \mathbf{x}$

    for l in $[2, \ldots, k]$ **do**

      $\mathbf{h}^{(l)} \leftarrow$ `activation`$(\mathbf{W}_{l-1}\mathbf{h}^{(l-1)} + \mathbf{b}_{l-1})$

    `prediction` $\leftarrow$ `softmax`$(\mathbf{W}_k\mathbf{h}^{(k-1)} + \mathbf{b}_k)$

    `// Categorical cross-entropy`

    $\mathcal{L}(\{\mathbf{W}_k, \mathbf{b}_k\}) = -\log p($`prediction` $= \mathbf{y}|\mathbf{x})$

    Optimise $\{\mathbf{W}_k, \mathbf{b}_k\}$ by minimising $\mathcal{L}(\{\mathbf{W}_k, \mathbf{b}_k\})$ via SGD.

Note that we will discuss the convolution explicitly as a 1D-operation from the point of view relevant to the goals of this thesis. We follow the notation used in (Bengio and Courville [4]), which provides a more general overview with examples of application to problems in compute vision and audio processing.

The one-dimensional convolutional operation is defined over two real-valued functions $x(t), w(t)$ known as the input and kernel. The discrete operation is defined as follows:

$$f(t) = \sum_{\tau=-\infty}^{\infty} x(\tau)w(t-\tau) \tag{2.3}$$

and is often abbreviated $x(t) * w(t)$. We refer to the output $f(t)$ as a feature map. In the more general continuous case, the sum is to replaced by an integral. Furthermore, in the case of time-series data, we usually apply the convolution for a fixed window of size T, up to the most recent input:

$$f(t) = x(t) * w(t) = \sum_{\tau=-T}^{0} x(\tau)w(t-\tau) \tag{2.4}$$

Intuitively, the operation may be thought of as a weighted average over a series of data, where the kernel function $w(t)$ defines the weight given to each data point. Thus, it may be used as a simple means of smoothing a noisy signal, e.g. the electromagnetic response of a neuron to particular stimuli or position information of an autonomous robot.

In the context of neural networks, this linear operation is applied analogous to the affine transformation in a standard MLP. Followed by a non-linearity, the convolution acts as a feature extractor taking as input a fixed window of elements in a sequence. We typically apply the convolutional operation several times (here n times) to the same input, using different kernels $w_n(t)$ thus resulting in multiple feature maps $f_n(t)$. This is almost always desirable, as it allows to extract different types of features from the input. Choosing the rectifying linear function (Glorot et al. [23]): $\texttt{ReLu}(x) = max(0, x)$ as the non-linearity, we can write the 1D-convolution as a feature extractor:

$$f_n(t) = \texttt{ReLu}(x(t) * w_n(t)) = \texttt{ReLu}(\sum_{\tau=-T}^{0} x(\tau)w_n(t-\tau)) \tag{2.5}$$

Learning thus corresponds to optimising the kernel function $w(t)$ so as to extract meaningful information from the input. The convolution operation has several attractive

properties a normal affine transformation does not fulfil. Among the most important for motion sequences are:

- Weight sharing: Note that the same kernel is applied regardless of the length of the sequence along the temporal dimension. A convolutional layer has thus significantly fewer parameters than an affine transformation applied to the same sequence. This reduction in model complexity makes over-fitting a much less likely occurrence.
- Temporal invariance: As the same feature detector is subsequently applied along the temporal dimension, the exact position of an indicative movement is irrelevant. This is in strong contrast to the affine transformation.

Figure 2.6 shows a full network which implements a 1D-convolution over time. We also show how convolutional layers can be combined with an affine transformation.

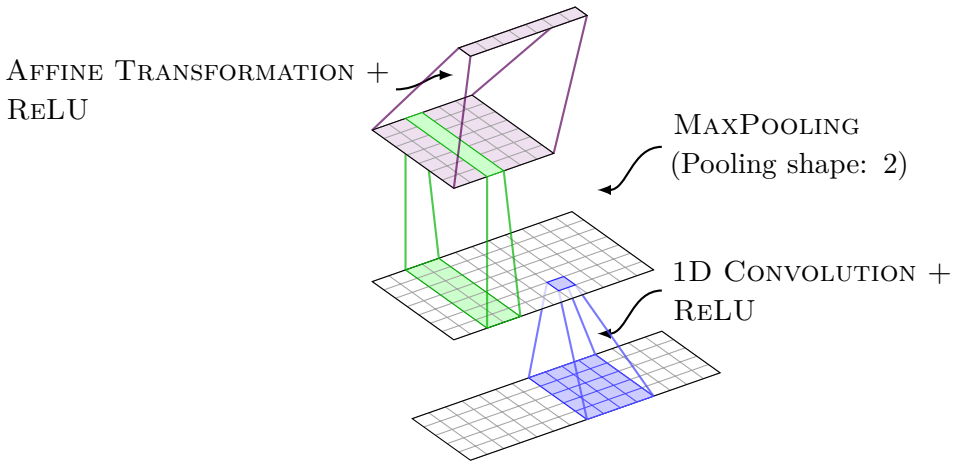
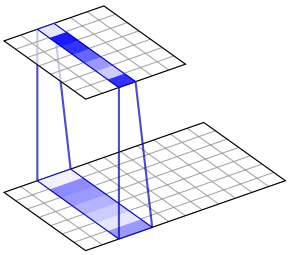


Fig. 2.6 An illustration of a convolutional neural network. Shown is a 1D-convolution over time as well as temproal MaxPooling

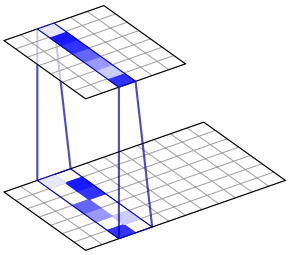
When describing the convolution operation as part of a larger neural network architecture the subsequent application of different kernel functions to the same input is usually referred to as a convolutional layer. We will adopt this for the remainder of this thesis.

2.5.2 Pooling

In nearly all CNN architectures, an application of a convolutional layer is directly followed by the pooling (also known as sub-sampling) operation. Pooling is a means to



(a) Upsampling by distributing the activation evenly



(b) Upsampling by randomly placing the activation

Fig. 2.7 A comparison of two upsampling techniques. Such layers are can be perceived as the approximate inverse of max-pooling.

reduce the size of the respective feature maps. Furthermore, it improves the positional invariance of the network by summarising subsequent values of each feature map. A simple and popular pooling operation is MaxPooling (Zhou and Chellappa [69]), the maximum value of m subsequent values of a feature map: $\max(f(t), ..., f(t+m))$. Positional invariance arises as these values were computed with the same kernel and an overlapping input.

When convolutional architectures are used as part of an autoencoder framework, it is important to note that the inverse of the pooling operation can not be computed. Instead, approximate methods, so-called "upsampling" operations are used. In previous applications of convolutional autoencoders to motion data (Holden et al. [32, 33]) the authors propose two approximate techniques shown in in Figure 2.7. Figure 2.7b Shows how upsampling can be performed by assigning the activation value to a random index in the inverse pooling window. As an alternative, the activation value can be easily distributed across the inverse pooling window (shown in Figure 2.7a).

2.6 Multi-Task learning

For complex real-world problems, it is common to apply separate learning algorithms for each individual property of interest, even when the input data is identical. This allows each predictor to extract the features most useful for its particular task. However, the amount of labelled data for each of the sub-tasks may be small, e.g. when a property is only recorded for a subset of the data. Thus, this leads to the familiar semi-supervised scenario in which one would wish to learn relevant statistics using both labelled and unlabelled data (w.r.t. each task).

In such cases, it can be useful to share general properties about the data while simultaneously allowing each model to extract additional information. This sharing of information is known as inductive transfer and can be a powerful tool for the improvement of generalisation across multiple learning problems.

In (Caruana [11]), the joint learning of low-level features is introduced as a simple and intuitive way of achieving inductive transfer, called *Multi-task learning*. These low-level features are shared and serve as the input to more complex, task-specific feature extractors. This allows each predictor to learn a hierarchy of higher-level abstractions while sharing statical strength with other modules. In the context of deep learning, Multi-task learning can be easily implemented by sharing lower-level non-linear transformation layer. This is depicted in Figure 2.8.

During optimisation, the task dependent layers are trained as usual, while the parameters of the shared feature extractors are updated by adding the gradients of all task-dependent errors. Hence, backpropagation may be applied as usual, making the joint optimisation in Multi-task learning simple and efficient. This joint training leads to a number of advantages, namely:

(i) An improvement of generalisation: Multiple learning tasks typically expose different statistics of the observations. Sharing these statistics through inductive transfer may give each predictor access to information that would otherwise not be available.

(ii) Regularisation: As lower-level information is shared between tasks, it becomes unlikely that these features merely capture particularities of the respective training set. Such information would typically not allow for robust performance on a variety of tasks. This has also been shown in empirical observations (e.g. Harvey and Pal [27]).

(iii) Learning efficiency: As parameters are shared across tasks, none of the individual predictors must optimise the low-level feature detectors single-handedly. This reduces the cumulated learning time.

More importantly, Multi-task learning provides a simple and intuitive way of combining supervised with unsupervised learning. This is possible as the generality of the method imposes no restrictions on the type of objective functions available for each sub-task. Thus, one can for instance jointly optimise parameters to solve a supervised classification problem while simultaneously learning a manifold of the data through reconstruction. Notice that this slight detail sets Multi-task learning in strong contrast to the previously studied pre-training and fine-tuning phase of autoencoder training. Such methods suffer from a serious drawback: Unsupervised learning cannot by definition not know which features will be relevant for subsequent classification (Rasmus et al. [59]). Multi-task learning on the contrary, may guide unsupervised learning and encourage the exposure of additional information relevant to the supervised task. This has been exploited in several recent applications where learning was performed in the low-data regime , including applications to motion data (Harvey and Pal [27], Cho and Chen [14]).

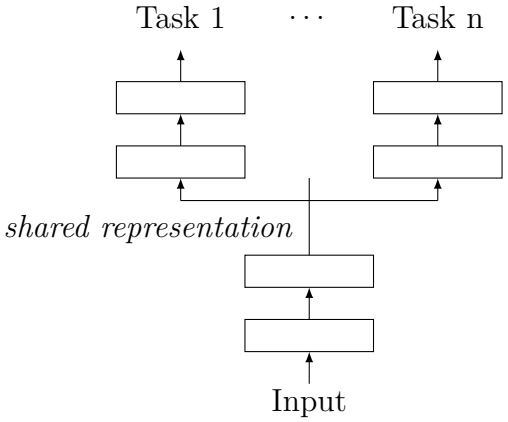


Fig. 2.8 An example of neural network trained in a Multi-task framework.

Chapter 3

Related work

We now review recent approaches to motion classification on the HDM05 data set. Note that since most recent results where achieved by different deep learning based methods, this review will be focused around such techniques. (Cho and Chen [14]) however, provides a comparison to other non-neural network based methods.

3.1 Providing meaningful evaluation

Before we give an overview of previous approaches to motion classification, we first discuss evaluation criteria used on the HDM05 data set. In recent years, several academic publications reported accuracies on the test set significantly over 90%, suggesting that motion classification on HDM05 is a solved problem. Most of these approaches fail to provide a correct split of the data to allow for meaningful evaluation.

Since the actions recorded as part of the HDM05 data set were performed by merely five actors, actors perform the same motion multiple times. Therefore, a random split into training and test sets is highly likely to result in takes of the same action performed by a particular actor in both sets. This makes classification of such data points in the test set significantly easier.

This problem has been first recognised by (Harvey and Pal [26]) who propose to split training and evaluation sets by actors. In order to obtain a more realistic estimate of the performance of other methods, the authors re-implement previously published methods and re-estimate their performance to obtain a more realistic estimate of future generalisation error.

However, as this was only done for a small subset of methods, we will highlight this by distinguishing between results obtained on HDM05 (random split) and HDM05 (split by actors). Results reported performing a random split must be considered over-confident. In (Harvey and Pal [26]) the authors found that an accuracy of 94.13% using the approach of (Cho and Chen [14]) on a random split corresponds to merely 81.64% on the data split by actors. However, as many of these approaches make an important contribution in terms of methodology, we will discuss these publications nevertheless.

3.2 Related approaches to motion classification

In (Chen and Koskela [13]) a specific type of single hidden-later neural network, known as an extreme learning machine (ELM, Huang et al. [34]), is proposed as a classifier on top of manually extracted features. The approach handles sequential motion data by classifying each individual frame to a particular action. By using the label of a motion sequence as the target class for each frame in the motion, this allows to increase the size of the labelled training data by a multiple.

While this helps with the general lack of supervised data in motion sequences, the learning algorithm is likely to be presented data points very similar to each other with different class labels. This could be the case for poses that are part of many actions (e.g. standing). This assumption is problematic as the property of smoothness. Final classification of a sequence is achieved by a majority vote of the per-frame action classifications.

In the ELM, the weights and biases of the hidden layer are initialised at random and held constant during training. This in strong contrast to the vast majority of other neural network architectures. Instead of a non-linear feature extractor, the first layer thus acts as a basis function expansion. The parameters between hidden and output layer can then be learnt in a single step by calculating the Moore–Penrose pseudoinverse. This allows to train the ELM much faster than traditional networks optimised by backpropagation. The system is illustrated in 3.1. An important part of the approach is the definition of manually extracted features. These features include (normalised) 3D marker positions and several distances to centroids and key joints.

Furthermore, the computation of the temporal difference of feature vectors is proposed to handle actions that are kinematically inverse of one another. An example of such a movement is the HDM05 action "*StandUpKnee*", where an actor stands

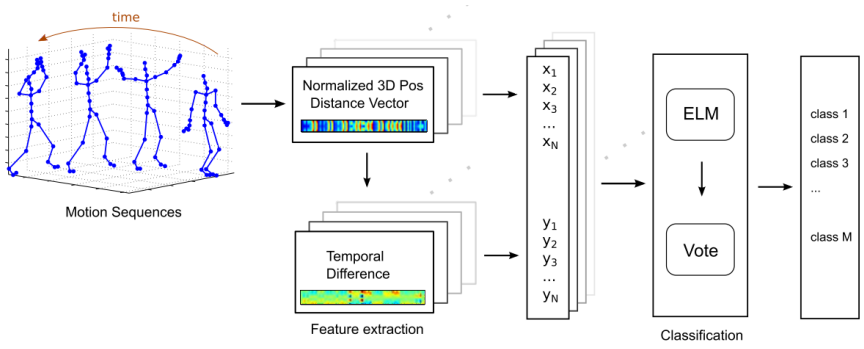


Fig. 3.1 An overview of the architecture proposed in (Chen and Koskela [13]). After hand-crafted feature extraction, a motion sequence is classified by a majority voting of an extreme learning machine.

up from a kneeling position, in contrast to doing so reverse ("*SitDownKnee*"). It is argued that, without the temporal difference, such actions could be easily misclassified as the respective kinematic inverse. This is a direct consequence of the frame-wise classification. The relevance of temporal difference computation is mainly justified by the existence of inverse actions in the HDM05 data set. The temporal inverse property of a convolution, on the other hand, does not require such additional feature definitions.

A combination of unsupervised and supervised training objectives has been first proposed in (Cho and Chen [14]), who propose to train an MLP in a Multi-task setting. The architecture proposed consists of a deep auto-encoder which shared its hidden layer with a classifier. Hence, the network is encouraged to learn an expressive hidden representation useful for both tasks. As the reconstruction error provides a natural unsupervised learning objective, the method can also be trained in a semi-supervised setting. This can be done by only evaluating the supervised cost for labelled examples. Unfortunately, the authors do not carry out such experiments.

The sequential property of MOCAP data suggests the use of architectures designed to handle such data. (Harvey and Pal [27]) explicitly account for this property by proposing a recurrent encoder-decoder as a means of semi-supervised motion recognition. Similar to (Cho and Chen [14]), a combination of supervised and unsupervised learning objectives is proposed. It is argued that this form of Multi-task learning enables the network to learn more invariant and expressive features. As before, backpropagation of the reconstruction error allows semi-supervised learning. In contrast to the Multi-task MLP, the explicitly incorporate unlabelled CMU data in the training process, making

it the first semi-supervised method for motion classification.

learning. The recurrent encoder-decoder network consists of several individual modules, namely a frame auto-encoder, frame classifier, recurrent encoder, recurrent decoder and sequence classifier. In addition to simply classifying and reconstructing the entire sequence as in (Cho and Chen [14]), the network also attempts to reconstruct and classify each individual frame. The recurrent encoder and decoder are designed as Long short-term memory (LSTM) cells (Hochreiter and Schmidhuber [31]), a popular choice type of recurrent neural network. In choice is motivated by recent advances with LSTMs in sequence-to-sequence modelling (Sutskever et al. [65]) and phrase representation (Cho et al. [15]).

A main advantage of LSTM networks is their ability to convert an arbitrarily long sequence into a fixed length representation at each time step. This is in contrast to a convolution operation over time. While an LSTM can learn arbitrarily long sequences, we argue that this is most likely not of crucial importance in motion data. This is because we expect an indicative movement for a class to be carried out in only a short window (1-2 seconds). Infinite context however, could lead to the recognition of coincidental correlations between movements far away in time.

Overall, the recurrent encoder-decoder approach provides state-of-the-art performance on the HDM05 dataset. As the full architecture consists of four different modules, the authors experiment with several combinations of architectures. Given the novel definition of the validation and test set, a combination of frame reconstruction, sequence reconstruction and sequence classification outperforms all previously reported results. The use of unlabeled data in form on the CMU database is shown to significantly improve the accuracy of the highest overall best performing model. On closer inspection of the results, however, it can be seen that the use of the CMU database actually weakens performance for a range of different combinations of modules. Unfortunately, the authors do not explain why this is the case. Since the architecture is proposed as a new means to semi-supervised learning, one would expect both additional supervised and unsupervised data to improve performance. It is also unclear if all parts of the architecture are needed given that the best model does not use the frame classifier. Nevertheless, the approach in (Harvey and Pal [27]) provides strong evidence that reliable motion labelling can be performed with semi-supervised deep learning models.

To summarise, we show the classification accuracies reported in recent publications in Table 3.1. As we will conduct our experiments using training and test sets split by actors, the baseline for our experiments will be 85.64%.

	HDM05 (random split) Test set Accuracy	HDM05 (split by actors) Test set Accuracy
Extreme Learning Machine [34]	91.9[a] (±1.20)	-
Deep LSTM networks [70]	97.25 (±0.43)	-
Hierarchical RNN [20][b]	92.98 (±1.05)	70.63 (±3.33)
Multitask MLP [14][b]	95.61 (±0.67)	81.64 (±2.41)
Recurrent Encoder-Decoder [26]	-	85.64 (±1.98)

Table 3.1 A comparison of recent classification results reported on the HDM05 data set. Shown are both results on a random split of the data as well as a split by actors, as proposed in (Harvey and Pal [26]). We show 95% confidence intervals of future test error.

[a] using only 40/65 motion classes
[b] as reported in [26]

Chapter 4

Deep generative models

We have so far introduced several means to representation learning as methods to expose the underlying structure of data. We hypothesised that, if such structure can be learnt through the formulation of unsupervised objectives, the availability of unlabelled data can be utilised. Thus, in comparison to traditional semi-supervised methods, representation learning allows us a to take on an alternative perspective.

The inference of causal relationships that led to observations at hand is however, a much broader problem and has been at the very core of machine learning research for decades. In particular, the learning of representations has close links to the study of *generative models*. Such models describe how data was generated given access to abstract properties, so-called *latent variables*. Such properties might correspond to the intentions of an actor before executing a particular motion, for instance the type of motion, the direction of movement or the pace in which it is carried out. This is in contrast to low-level information such as the rotational velocity at a particular point in time. As another example, consider a collection of photographs. Latent variables may correspond to particular objects shown, an angle or the intention of the photographer (e.g. to make a particular scene look dramatic). Clearly, knowledge of such high level concepts could be helpful for classification tasks.

This raises the question about how we can infer such latent variables, given access to only the observations themselves. If we can propose powerful generative models in conjunction with sound inference processes, latent variables may give access to useful information that may otherwise not be detected by other unsupervised methods.

In this chapter, we will investigate models proposed to explain generative processes in the presence of latent variables, so-called latent variable models. This will allow us to bridge the gap between this type of models, traditionally studied as part of the probabilistic modelling community, and deep learning research. We will introduce the variational autoencoder (VAE, Kingma and Welling [40]), which has quickly established itself as a powerful unsupervised learning method. This will form the basis of the final model proposed for motion classification.

Finally, we present a novel learning algorithm by combining the ideas of VAEs, convolutional neural networks and multi-task learning.

4.1 Latent variable models

In generative modelling, we are concerned about maximising the likelihood of observations in our data set while simultaneously minimising the density of unlikely data points. As mentioned before, we consider the observations $\mathbf{x}$ to be dependent on same latent variables $\mathbf{z}$. Thus, the data density can be obtained by considering all possible latent configurations according to a prior $p(\mathbf{z})$ and marginalising over the joint distribution $p(\mathbf{x}, \mathbf{z})$:

$$p(\mathbf{x}) = \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z} \tag{4.1}$$

Inference of latent variables in this framework can simply be formulated by applying Bayes' rule:

$$p(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z}) p(\mathbf{z})}{\int p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}} \tag{4.2}$$

However, the integral over all possible latent variables poses a serious problem to the generative framework. Even when we ensure that $dim(\mathbf{z}) << dim(\mathbf{x})$, the integral over the latent space will can easily become intractable. This can happen when we assume continuous latent variables, which may lead to an integral that cannot be solved analytically. In such cases one can make simplifying assumptions, e.g. that of binary or categorical latent variables. This allows the explicit summation over all configurations of the latent space. Notice that, even for discrete $\mathbf{z}_i$ the number of possible configuration grows exponentially with the dimensionality of the latent space. Let M denote the number of of possible values each $\mathbf{z}_i$ can take on. Then, the inference operation poses exponential complexity $\mathcal{O}(N^{dim(\mathbf{z})})$. This is an example of the *curse*

of dimensionality (Bellman [3]). Alternatively, one can either resort to approximate methods (such as variational inference, see Barber [2]) or sampling (Murray [54]).

4.2 The variational Autoencoder

The ideas of generative latent variable models have recently been combined with the representational power of neural networks. The resulting model, the variational autoencoder (VAE, Kingma and Welling [40]) constitutes a powerful and efficient *unsupervised* feature learning method. Since their proposal, VAEs have been in the forefront of research on deep generative models (e.g. (Burda et al. [10], Kingma et al. [39])). As they are powerful devices for unsupervised representation learning, they provide a simple means to semi-supervised learning, analogous to the discussion of previous models.

As highlighted in the previous section, generative latent variables must both give a definition the distribution over latent variables as well as an answer to how the intractable integral over the latent space can be performed. We will give answers to both of these questions in the following subsections. In order to explain the variational autoencoder, we closely follow the notation and explanation given in Doersch [18].

4.2.1 Defining the generative process

As distinct from models which require the specification of latent features, we will now introduce a model capable of learning latent features. Compare this to the definition of the Gaussian mixture model: Here we explicitly specified that latent features correspond to the mixture component of the model. We argue that this is makes the model easier applicable to data of complex nature. Indeed, research on state-of-the-art generative models has shown that learnt latent features correspond to humanly interpretable properties of the data. In (Larsen et al. [44]) the authors conduct experiments on a data set of human faces and found that some dimensions of the latent space correspond to the presence or absence of glasses.

This section will introduce a formulation of the prior and likelihood terms $p(\mathbf{z}), p(\mathbf{x}|\mathbf{z})$ in a VAE framework. While the general VAE framework allows the definition of a wide range of distributions, we will consider the case where both prior and likelihood are chosen as Gaussian distributions. A list of alternative distributions can be found in

(Kingma and Welling [40]).

We begin with defining the prior as an isotropic Gaussian distribution:

$$p(\mathbf{z}) = \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{4.3}$$

In order to understand how such a surprising simple distribution over the latent variables is possible, it is important to understand that any arbitrary distribution can be generated through a transformation of Gaussian distributed random variables. This is schematically shown in Figure 4.1.

Also note that explicitly define independence between the latent variables by choosing to set the covariance matrix equal to the identity matrix. While this first appears as mere mathematical convenience, it has an important interpretation. By forcing independent between the latent features, the generative model is to learn disentangled causal factors, a main goal in representation learning (Bengio et al. [6]). This objective is much more directly expressed with an isotropic Gaussian prior, rather than in other methods (e.g. the ℓ_1 penalty in sparse autoencoders) who have the same objective.

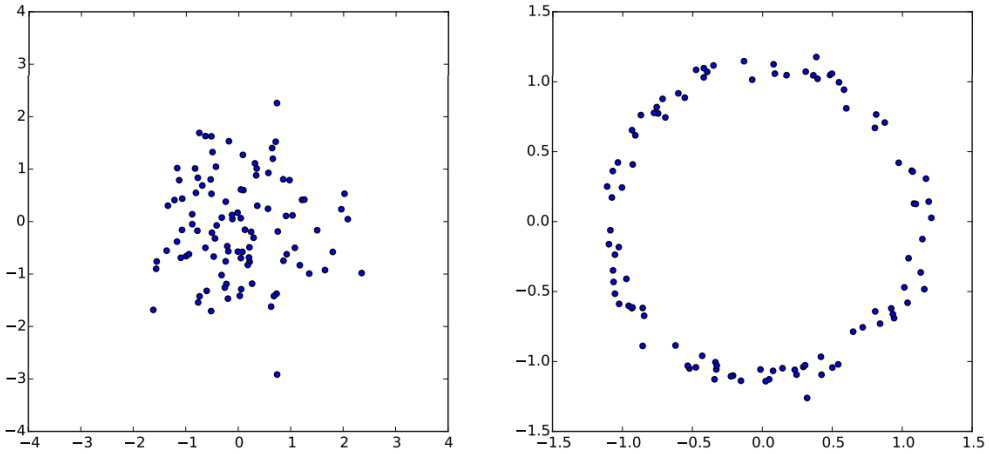


Fig. 4.1 The transformation of normally distributed random variables. The transformation chosen is $g(\mathbf{z}) = \mathbf{z}/10 + \mathbf{z}/||\mathbf{z}||$. Figure taken from (Doersch [18])

Secondly, we define the likelihood to also be Gaussian, with mean equal to some transformation $f(\mathbf{z})$ of the latent variables:

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(f(\mathbf{z}), \mathbf{I}) \tag{4.4}$$

$f(\mathbf{z})$ can be understood as the function that transforms the normally distributed variables to the actual observations at hand. It corresponds to the transformation in Figure 4.1. In variational autoencoders, we choose $f(\mathbf{z})$ to be the non-linear transformation of a neural network. We will denote the parameters of the network as θ, allowing us to write $f(\mathbf{z};\theta)$ to denote a forward pass through this neural network. During training, we will be able to learn this transformation by encouraging the network to reconstruct its input. We define the define the reconstruction loss $\mathcal{L}_{us}$ as:

$$\mathcal{L}_{us} = ||\hat{\mathbf{x}} - \mathbf{x}||_2^2 \tag{4.5}$$

In addition, a hyperparamter σ^2 is usually defined to allow scaling of the covariance:

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(f(\mathbf{z};\theta), \sigma^2 \cdot \mathbf{I}) \tag{4.6}$$

The graphical model of the generative process in VAEs in shown in Figure 4.2. $\mathbf{z}^{(n)}$ are the latent and $\mathbf{x}^{(n)}$ the observed variables for data point n. Th arrow from $\mathbf{z}^{(n)}$ to $\mathbf{x}^{(n)}$ can be thought of as the transformation through $f(\mathbf{z};\theta)$, i.e. the forward pass of a neural network. Thus, we include the parameters which are fixed once the model is trained. Note that since we defined $p(\mathbf{z})$, latent variables can be easily sampled. Passing these through $f(\mathbf{z};\theta)$, new data points can be generated.

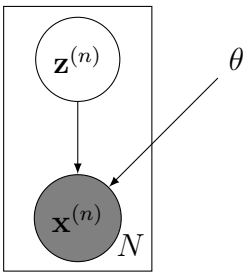


Fig. 4.2 The generative process in a variational autoencoder. Given latent variables $\mathbf{z}^{(n)}$, $\mathbf{x}^{(n)}$ are created through non-linear function $f(\mathbf{z},\theta)$ parametrised by θ.

4.2.2 The variational lower bound

The previous section introduced the generative perspective of a variational autoencoder. We will now show how such a model can be trained to optimise the likelihood of the observations. Furthermore, it is not yet clear how inference in this model is performed, i.e. how to compute $p(\mathbf{z}|\mathbf{x})$. Recall that this term involves an intractable integral over the latent space (see equation 4.2). Rather than attempting to approximate this

posterior via sampling, we will instead utilise variational methods (explaining the name of the VAE). That is, we approximate $p(\mathbf{z}|\mathbf{x})$ through some distribution $q_\phi(\mathbf{z}|\mathbf{x})$ with parameters ϕ. This eludes the intractable integral which would otherwise need to be solved. If the parameters ϕ can be estimated through a model with sufficient statistical strength (e.g. a deep neural network), inference is fast and efficient.

In order to relate $p(\mathbf{z}|\mathbf{x})$ to $q_\phi(\mathbf{z}|\mathbf{x})$ we first define the Kullback-Leibler divergence between two distributions $q(x)$ and $p(x)$:

$$\mathcal{D}_{KL}(q(x)||p(x)) := \int q(x) \log \frac{q(x)}{p(x)} dx \tag{4.7}$$

which can be thought of a measure of difference between two distributions. Note that the KL-divergence is not a distant measure, since $\mathcal{D}_{KL}(q(x)||p(x)) \neq \mathcal{D}_{KL}(p(x)||q(x))$. We can see that that

$$q(x) = p(x) \rightarrow \mathcal{D}_{KL}(q(x)||p(x)) = 0 \tag{4.8}$$

In addition, it is always true that

$$\mathcal{D}_{KL}(q(x)||p(x)) >= 0 \tag{4.9}$$

The KL-divergence allows us to relate the true posterior $p(\mathbf{z}|\mathbf{x})$ and the approximation $q_\phi(\mathbf{z}|\mathbf{x})$. By rewriting the integral within the KL-divergence as an expectation, we obtain:

$$\mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{q_\phi}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{z}|\mathbf{x})] \tag{4.10}$$

We can bring the log marginal likelihood $\log p(\mathbf{x})$ into the expression by applying Bayes' rule to $\log p(\mathbf{z}|\mathbf{x})$:

$$\begin{aligned} \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) &= \mathbb{E}_{q_\phi}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{x})}{p(\mathbf{z})}] \\ &= \mathbb{E}_{q_\phi}[\log q_\phi(\mathbf{z}|\mathbf{x}) - \log p(\mathbf{x}|\mathbf{z}) - \log p(\mathbf{z})] + \log p(\mathbf{x}) \end{aligned} \tag{4.11}$$

where we moved the log marginal likelihood out of the expectation as it does not depend on $\mathbf{z}$. $\log p(\mathbf{x})$ is explicitly written out as is it is the quantity maximised during training. By doing so, we ensure that observed data is assigned high density while implausible configurations are less likely under the generative model. Also, If $\log p(\mathbf{x})$

is maximised, it must be possible to generate data from the latent variables.

Finally, we can rearrange and express the right-hand-side in terms of another Kullback-Leibler divergence:

$$\log p(\mathbf{x}) - \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{q_\phi}[\log p(\mathbf{x}|\mathbf{z})] - \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{4.12}$$

This forms the mathematical basis of the variational autoencoder. Each of the terms on either site of the equation has an important interpretation: Starting from the left, we have $\log p(\mathbf{x})$, the quantity we want to maximise during training. This can be done by performing gradient ascent on the right-hand-side of the equation, a we cannot easily evaluate $\log p(\mathbf{x})$ directly. As we will, this optimisation procedure can be done efficiently be performing stochastic gradient ascent (Bottou and Bousquet [8]) (or any other optimisation algorithm capable of learning in mini-batches). Hence, we can optimise $\log p(\mathbf{x})$ up to the approximation error measured by the KL-divergence between $q_\phi(\mathbf{z}|\mathbf{x})$ and the true intractable posterior $p(\mathbf{z}|\mathbf{x})$. As the KL-divergence is guaranteed to be non-negative, the right hand side provides a *lower bound* on the log marginal likelihood. Thus, this term is called the **variational lower bound**.

4.2.3 Learning of latent variables

Learning in variational autoencoders corresponds to the optimisation of equation 4.12. We will now show how this quantity can be optimised with stochastic gradient ascent. Consider first the KL-divergence between the true prior $p(\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ on the right-hand-side of equation 4.12. During training, we will evaluate $q_\phi(\mathbf{z}|\mathbf{x})$ through a forward-pass of a neural network chosen to estimate the parameters ϕ. Thus, we do not sample the latent variables $\mathbf{z}$ from $p(\mathbf{z})$, as we would do during generation (Figure 4.2). This introduces a variational approximation error $\mathcal{L}_{KL}$ which can be measured by another KL-divergence:

$$\mathcal{L}_{KL} := \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{4.13}$$

If, in addition to $p(\mathbf{z})$, we also choose $q(\mathbf{z}|\mathbf{x})$ to be Gaussian, the KL-divergence can be computed analytically, as shown in (Kingma and Welling [40]). Rather than estimating a full covariance $\mathbf{\Sigma}_z$, the neural network will only estimate the terms on the diagonal. This enforces the disentangling of causal factors. In order to make this

explicit, we will write $\boldsymbol{\sigma}_2^2$ for the vector of values on the diagonal of the covariance matrix. Denoting the estimated mean of $q_\phi(\mathbf{z}|\mathbf{x})$ as $\boldsymbol{\mu}_z$, the exact solution of $\mathcal{L}_{KL}$ is:

$$\begin{aligned}\mathcal{D}_{KL}(\mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)||\mathcal{N}(\mathbf{0}, \mathbf{I})) &= \int q_\phi(\mathbf{z}|\mathbf{x}) \log p(\mathbf{z}) d\mathbf{z} - \int q_\phi(\mathbf{z}|\mathbf{x}) \log q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} \\ &= \int \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2) \log \mathcal{N}(\mathbf{0}, \mathbf{I}) d\mathbf{z} \\ &- \int \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2) \log \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2) d\mathbf{z} \\ &= \frac{1}{2} \sum_{j=1}^{J} (1 + \log \boldsymbol{\sigma}_{z,j}^2 - \boldsymbol{\mu}_{z,j} - \boldsymbol{\sigma}_{z,j}^2)\end{aligned} \tag{4.14}$$

The first term of the left hand side of 4.12 poses a problem, however. It involves an expectation the latent variables. We can however, approximate this quantity with the Monte Carlo estimate:

$$\mathbb{E}_{q_\phi}[\log p(\mathbf{x}|\mathbf{z})] \approx \frac{1}{S} \sum_{s=1}^{S} p(\mathbf{x}|\mathbf{z}^{(s)}); \mathbf{z}^{(s)} \sim q_\phi(\mathbf{z}|\mathbf{x}) \tag{4.15}$$

where S is the sample size. In fact, we can set $S = 1$ and directly use the result of the forward pass, i.e. $q_\phi(\mathbf{z}|\mathbf{x})$ in the computation of the gradient. This approximation is possible as each $\mathbf{z}^{(s)}$ will be a likely sample of the VAE (as explained in (Doersch [18])).

Thus, this leads us to the following equation:

$$\log p(\mathbf{x}) - \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z}|\mathbf{x})) = \log p(\mathbf{x}|\mathbf{z}) - \mathcal{D}_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) \tag{4.16}$$

Finally, in order to compute the gradients w.r.t. to each parameter of the neural networks chosen to approximate the parameters of q_ϕ, $f(\mathbf{z}; \theta)$ we need to specify back-propagation for the network. While we can simply apply the chain rule of derivatives for most of the layers, the latent variables pose a problem. Recall that $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$, i.e. the latent variables are sampled during training. Backpropagation however, is unable to handle stochastic layers. To surpass this problem, the *reparameterisation trick* is proposed in ([40]).

Recall that for any multivariate Gaussian:

$$\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \boldsymbol{\mu} + \boldsymbol{\Sigma} \cdot \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{4.17}$$

Thus, in case of a Gaussian q_ϕ, we can simply obtain samples of $\mathbf{z}$ by evaluating:

$$q_\phi(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2) = \boldsymbol{\mu}_z + \boldsymbol{\sigma}_z^2 \cdot \boldsymbol{\epsilon}; \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{4.18}$$

That is, samples of $\boldsymbol{\epsilon}$ are presented to the network as the input. This works as we can compute the gradients of the parameters if we formulate $\boldsymbol{\epsilon}$ as stochastic input. This is trivial, as data points $\mathbf{x}^{(i)}$ can also be considered stochastic during the training of any neural network. This allows us to compute the gradients of any of the parameters ϕ in the encoder network through standard Backpropagation.

We now show the schematic illustration of a variational autoencoder in Figure 4.3. For each data point $\mathbf{x}^{(i)}$, we obtain a sample $\boldsymbol{\epsilon}^{(i)}$ from a multivariate normal distribution. $\mathbf{x}^{(i)}$ is passed through several non-linear transformations in the encoder, resulting in variables $\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2$. Using $\boldsymbol{\epsilon}^{(i)}$ we obtain $\mathbf{z}^{(i)}$ through a linear combination. By passing $\mathbf{z}^{(i)}$ through the decoder, we obtain a reconstruction $\hat{\mathbf{x}}$ of the input, allowing us to evaluate $\mathcal{L}_{us}$. Evaluating expression 4.14 yields $\mathcal{L}_{us}$ allowing us to perform a step of stochastic gradient ascent by by evaluating the gradients $\frac{\partial(\mathcal{L}_{kl}+\mathcal{L}_{us})}{\phi_i}$, $\frac{\partial(\mathcal{L}_{kl}+\mathcal{L}_{us})}{\theta_i}$.

Note that the reconstruction is a particularly important term. By training the model from reproducing a data point from only the latent features, these latent features must capture enough information about the observations. Instead of specifying the interpretation of the latent space beforehand however, it is learnt automatically.

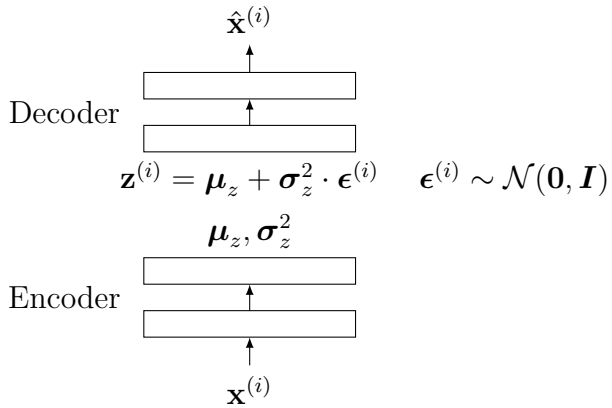


Fig. 4.3 Sketch of a variational autoencoder. Figure inspired by [18]

The semi-supervised training of VAEs is conceptually simple. Latent features are learnt in a fully unsupervised fashion on both the labelled as well the unlabelled data.

Subsequently, a classifier is applied after transformation of the data in the latent space. This has been proposed as the M1 model in (Kingma et al. [39]).

4.3 Convolutional Multi-task VAEs

We now introduce a novel learning architecture for motion classification, the convolutional Multi-task VAE (akin to the M2 model in Kingma et al. [39]). As the name suggests, this method combines the ideas of variational autoencoders, convolutional neural networks and Multi-task learning.

These design choices have been made for the following reasons:

- Convolution neural networks: A convolution over time allow the model to account for the inherent time series property of motion data. Weight-sharing furthermore allows the model to be efficiently trained.
- Variational Autoencoders: We aim to make use of the representational power of deep generative models to allow learning of expressive latent features.
- Multi-Task Learning: We add a supervised objective to help guide the model to learn latent features relevant for the task at hand. As this supervised task is optional, the model can be trained in a semi-supervised fashion by only evaluating the performance on the supervised task for labelled examples.

As each of these ideas are general, their combination is straight-forward given access to modular implementations of each component. We show the forward pass of the method during test-time in Figure 4.4. Notice that in order not to overload the figure, we do not show the decoder. It can simply be thought of as the inverse of the encoder network. As before, the sampled latent variables are the input of the decoder.

Training of this model is analogous to the training of a standard VAE with an additional supervised cost $\mathcal{L}_s$. We show the training procedure in algorithm 4.1. In addition, we introduce weights $\lambda_1, \lambda_2, \lambda_3$ which can be used to express a trade-off between the reconstruction, classification and KL-divergence losses $\mathcal{L}_{us}, \mathcal{L}_s, \mathcal{L}_{KL}$,

Algorithm 4.1: The learning procedure for convolutional Multi-Task VAEs. We use the functions $decode(\mathbf{x}^{(i)}), encode(\mathbf{z}^{(i)})$ as acronyms for the forward-pass of an encoder or decoder neural network respectively.

Input : Labelled data $\{(\mathbf{x}^{(i)}, y^{(i)})\}_{i=1}^{M}$, Unlabelled data $\{\mathbf{x}^{(j)}\}_{j=1}^{N}$, Parameters θ, Loss weights $\lambda_1, \lambda_2, \lambda_3$

Output : Trained parameters θ

begin
  for $\mathbf{x} \in (\{\mathbf{x}^{(i)}\}_{i=1}^{M} \cup \{\mathbf{x}^{(j)}\}_{j=1}^{N})$ **do**
    `// Encoder`
    $\boldsymbol{\sigma}_z^2, \boldsymbol{\mu}_z \leftarrow encode(\mathbf{x}^{(i)})$
    $\mathbf{z}^{(i)} \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)$
    $\mathcal{L}_{KL} = \mathcal{D}_{KL}(\mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)||\mathcal{N}(\mathbf{0}, \mathbf{I}))$
    if $y^{(i)}$ **then**
      `// Classifier`
      $\mathcal{L}_s = -\log P(y^{(i)}|\mathbf{x}^{(i)}, \mathbf{z}^{(i)})$
    `// Decoder`
    $\hat{\mathbf{x}}^{(i)} \leftarrow decode(\mathbf{z}^{(i)})$
    $\mathcal{L}_{us} = ||\mathbf{x}^{(i)} - \hat{\mathbf{x}}^{(i)}||_2^2$
    if $y^{(i)}$ **then**
      $\mathcal{L} = \lambda_1 \mathcal{L}_{KL} + \lambda_2 \mathcal{L}_{us} + \lambda_3 \mathcal{L}_s$
    else
      $\mathcal{L} = \lambda_1 \mathcal{L}_{KL} + \lambda_2 \mathcal{L}_{us}$
    Optimise θ by minimising $\mathcal{L}$ via stochastic gradient descent.

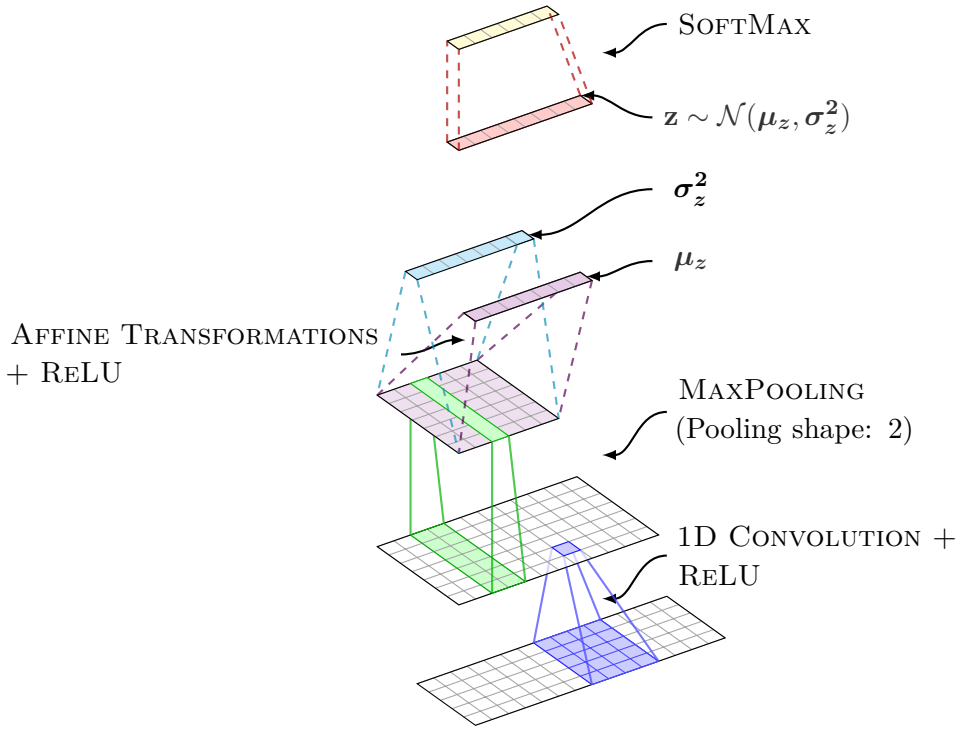


Fig. 4.4 A Convolutional Multi-Task VAE. For the sake of clarity, we do not show the decoder, which is simply the inverse of the encoder. Thus, this Figure illustrates the use during test time for a classification task. The layer highlighted in red is the result of sampling from a multivariate Gaussian distribution with the parameters previously estimated by the encoder.

Chapter 5

Experiments

Throughout this thesis, we have stated several hypotheses regarding the design of an architecture capable of the robust analysis of human motion. In order to justify these claims, will now empirically evaluate each of the hypotheses stated. Finally, we evaluate the learning architecture proposed in section 4.3. We present state-of-the-art results on a challenging motion recognition problem outperforming a large range of previously proposed approaches. In addition, we provide insight into both the learning process, as well as the representational strengths of the proposed approaches.

To summarise, the will evaluate the following hypotheses:

(i) Learning representations in a semi-supervised setting: As a main paradigm of this thesis, we put forward algorithms capable of learning from both labelled and unlabelled data. We will conduct experiments evaluating the capability of the proposed models to harness the availability of unlabelled data. Since this is a fundamental concept of this thesis, we will evaluate this property as part of each of the hypotheses rather than in a separate section.

(ii) Analysing sequences with CNNs: We argued that the sequential nature of data is a fundamental property which must be considered in the design and choice of learning algorithms. We proposed a convolution over time as implemented by a convolutional neural network as a means to classifying motion sequences.

(iii) Improving generalisation by enabling inductive transfer: Multi-task learning was introduced as a powerful, yet simple method to a) enable inductive transfer between learning tasks and b) guide unsupervised learning through a supervised learning signal.

(iv) Latent feature learning with deep generative models: Deep generative models have been presented as powerful latent variable models. We motivated their formulation through the fundamental concept of representation learning.

5.1 MOCAP data sets

5.1.1 The HDM05 data base

The HDM05 data base[1] is one of the few examples of high-quality collections of MOCAP recordings with consistent annotations. It has therefore been a popular benchmark data set for MOCAP classification algorithms. It comprises a large range of motion, ranging from simple walking over a variety of different sports (e.g. badminton or rope skipping) to miscellaneous activities, e.g. the tying of shoes. In addition, the data base incorporates fine-grained classes (e.g. grabbing actions at different heights). This allows us to give an estimate of the performance in applications in which the detection of subtle differences is crucial (e.g. medical gait analysis). We give an example of a punching action in HDM05 in Figure 5.1.

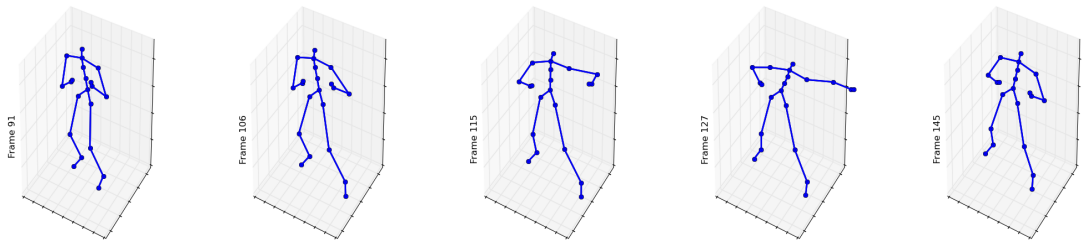


Fig. 5.1 An example of the 'PunchLSide' action defined in the HDM05 data base. We made a full animation available.[a] (This also shows the effect of padding as part of the pre-processing)

[a] http://www.jonathanschwarz.de/wp-content/uploads/2015/03/PunchLSide.gif

While the original data base distinguishes between over 100 classes, we follow (Cho and Chen [14]) and summarise some of these classes. To give an example, we summarise all classes that distinguish between different number of repetitions of the same motion. This leaves us with 65 classes, a full list of which is given in Table 4.

[1] http://resources.mpi-inf.mpg.de/HDM05/

All actions in the data base have been performed by merely five actors, which makes the split into training and test sets an important consideration (see section 3.1). The number of actions performed by each actor is as follows: ['bd' : 638, 'bk' : 748, 'dg' : 631, 'mm' : 538, 'tr' : 595]. The proposed split into training and test data proposed in (Harvey and Pal [26]) is shown in Table 5.1. One of the actors in the training set may be used as held-out set for validation purposes. In our experiments, we chose the actor with the initials ['mm'] as a validation set. While we experienced with 3-fold cross-validation, we found a simple validation set to be sufficient as an estimate.

Training		Test	
Actors	Fraction	Actors	Fraction
['bd', 'bk', 'mm']	61.1%	['dg', 'tr']	38.9%

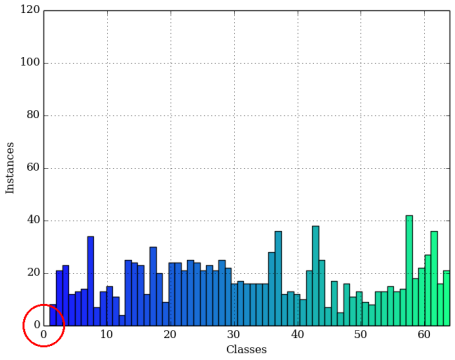
Table 5.1 Assignment of motions performed by actors in the HDM05 data set into training and test. This split has been proposed by (Harvey and Pal [26]).

The distribution of training examples across classes is shown in Figure 5.2. Notice that using the split in (Harvey and Pal [26]), there is no instance of class 0 (cartwheel) in either training or validation set (highlighted with a red circle). Thus, none of the models about to be evaluated will be able to correctly classify instances of class 1. While we considered yet another re-definition of the training and validation sets, we chose keep the sets as defined in order to ease the comparison with previous results. We argue that the introduced error is small, as the fraction of actions of class 0 is only 1.5% of the test set and 0.5% of the overall data set. This decision is reflected in all confusion matrices about to be shown.

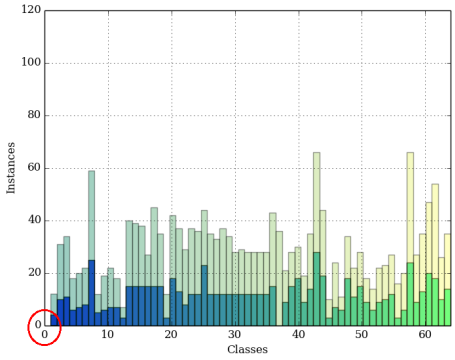
5.1.2 The CMU data base

As a data set for unsupervised learning, we will consider the CMU data base[2] which comprises over 10 hours of MOCAP recordings with 144 different actors. In contrast to HDM05, the data base follows no consistent annotation scheme, making it unsuitable for supervised learning. Instead, the motions are given a coarse description of the actions performed. Overall the CMU data base includes an extremely vast variety of human motions, including several instances of actors riding a motorcycle and long sequences with little movements (e.g. actors talking).

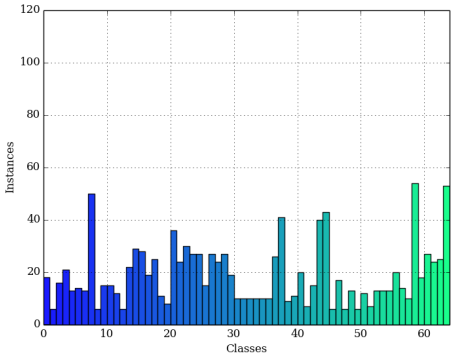
[2] http://mocap.cs.cmu.edu/



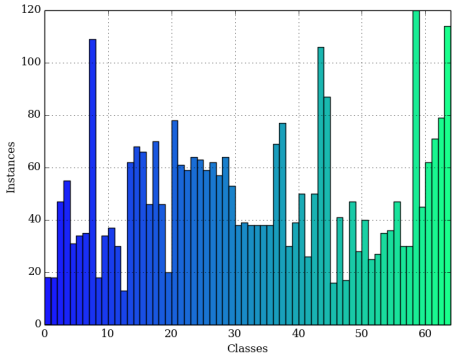
(a) Training set (Actors 'bd' & 'bk')



(b) The validation set (Actor 'mm').



(c) Test set (Actors 'dg' & 'tr')



(d) The overall distribution of instances.

Fig. 5.2 The distribution of instances for each class in the HDM05 data base. Shown in b) is also the union of training and validation sets (shaded).

Unfortunately, The vast variety of these descriptions does not correspond to any particular class in HDM05. We argue that this makes the application of classical semi-supervised learning algorithms (e.g. a TSVM or Label Propagation - see section 2.2) more difficult. If the unlabelled data is fundamentally dissimilar, approaches that merely optimise decision boundaries may change these boundaries in an unfavourable fashion. This can happen if many unlabelled data point are projected in the margin between two classes in feature space. Representation learning, on the other, aims at learning underlying properties of the data (e.g. a manifold of human motion) and is thus less likely to suffer from such problems. In order to highlight the different types of motions in the data bases, we show a recording taken from the CMU data base in Figure 5.3. In this MOCAP clip, the action appears to be swinging while simultaneously holding onto a bar. We found no similar motion in the HDM05 data base.

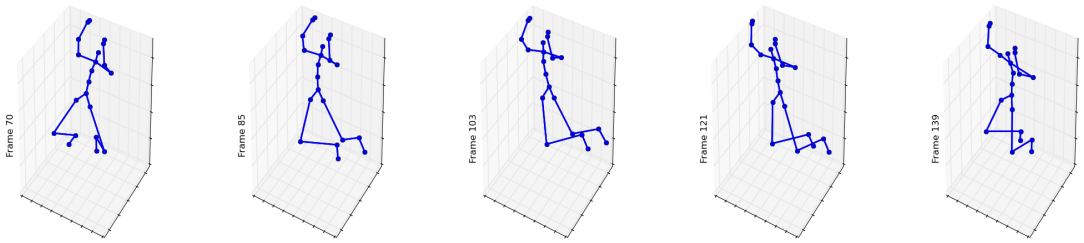


Fig. 5.3 An examples of a motion sequence randomly picked from the CMU data base. We made a full animations available.[a]

[a] http://www.jonathanschwarz.de/wp-content/uploads/2015/03/Hanging.gif

5.1.3 Data preprocessing

In terms of preprocessing, we apply the preprocessing steps in (Holden et al. [32]) to both the CMU and HDM05 databases. In particular, we represent each pose in a motion in terms of the 3D-joint positions of 22 joints with respect to the body's local coordinate system, originated at the ground. We also normalise the data by subtracting the mean pose and dividing by the standard deviation to normalise the data. This is done independently for each data set and helps in making the data more similar. Details can be found in (Holden et al. [32]).

As actions of both data bases are recorded at varying lengths, we separate the actions into fixed length windows of 240 frames. As many of the CMU recordings are significantly longer, we separate such actions into multiple short ones. This is done with an overlap of 120 frames in order to maximise the amount of unlabelled training data. For motion sequences shorter than this, we place the recorded sequences in the middle of the 240 frames and padd the sequence with the first/last frames to either side.

5.2 Initial experiments

Before evaluating each of the aforementioned hypotheses, we now discuss initial choices considering the learning process. In general, as this thesis is fundamentally about evaluating models, we avoided the excessive tuning of hyper-parameters and reported the best result for each model after only few experiments. This choice was also made to reduce the risk of incorrectly interpreting the performance of the model (e.g. when claiming a model to be superior while the causal factor of the improvement was a different weight initialisations). Thus, for all models, we kept the seed of the random number generator fixed and made only minimal changes to the architecture. We also avoided the use of sophisticated weight initialisations and regularisation techniques.

We used the Adam optimisation routine (Kingma and Ba [38]) with Nestorov momentum (Sutskever et al. [64]) for all experiments conducted. This choice was made after initial experiments with several popular optimisation methods. After observing the result on several random initialisations, we found Nadam to perform best in the majority of cases.

For all of the experiments, we found it crucial to report the final test set accuracy *after training on the union of training and validation set*. This was done after evaluating different parameters through early stopping on the validation set. We did this to maximise the amount of training data. Rather than stopping after the same number of training epochs that led to the best result on the validation set, we stopped this final training once a similar loss was reached. We found that doing so resulted in significantly better performance.

The specification of all models evaluated in this chapter can be found in Table 1 (for discriminative models) and Table 2 (for generative models).

5.3 Analysing sequences with CNNs

In Chapter 2, we introduced a convolution over time as a means to handle the sequential property of motion data. This is in contrast to recent approaches that have favoured recurrent architectures. We argue that such networks as they reduce the chance of detecting coincidental correlations and due to their simpler architecture.

In order to evaluate this hypothesis, we trained both a CNN as well as an MLP and compare it to previously reported results obtained with recurrent networks. The MLP will serve as an example of a model incapable of explicitly handling sequences. Note that all of these methods are trained in a purely supervised fashion and will thus form a supervised baseline for further experiments. The results of the experiments are shown in Table 5.2. Both sequential methods show an improvement over the MLP, which nevertheless achieves a respectable performance of 80.08%, considering its simplicity.

	HDM05
LSTM [26]	81.97 (±2.37)
MLP	80.08 (±2.56)
CNN	**82.05** (±2.36)

Table 5.2 A comparison of convolutional, recurrent and MLP neural networks architectures. We compare the performance of purely supervised training on HDM05.

A visualisation of the weights of its first layer offers an interesting insight into the behaviour of the MLP. Figure 5.4 shows such connections for two representative units. Some of the weights indicate short temporal correlations for particular joints, indicating that a particular unit is sensitive to the position of one or more joints. However, as the unit is connected to all features at any point in time, a strong (positive or negative) activation only occurs if an actor is moving in a very particular style throughout the motion. We argue that in most cases, the indicative movement in a particular action may happen at any point in time and ought to be easily detectable, regardless of the occurrence along the temporal axis. The MLP, on the contrary, can only achieve the detection of such occurrences by combining the activations of many units that have been trained to observe complete sequences. This also makes it easier for the MLP to over-fit the training data, as units can learn to react to individual data points in the training set.

A convolution over time, on the contrary, is designed to scan the entire sequence for such patterns in the motion. The position along the temporal dimension of the

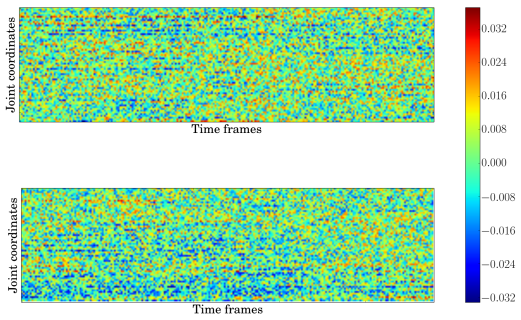


Fig. 5.4 Visualisations of the weight matrix of an MLP trained on HDM05. Shown are the connections of two randomly picked units in the first layer. Green colour corresponds to a weight of 0, red colour to positive, blue colour to negative weights.

indicative movement of a action is thus irrelevant and can be easily recognised. This also makes it easy to handle motions which are the kinematic inverse of one another (as present in the HDM05 data set). Other feature-based approaches require explicit feature definitions to handle such actions (see Chapter 3).

As we do not observe a clear difference in performance between the recurrent and convolutional networks, we suggest to make the choice of architecture dependent on other considerations. A reason to prefer a recurrent network, for instance, could be their property of easily handling sequences of arbitrary length (which requires padding with CNNs).

The best overall performance so far and thus the supervised baseline is achieved by a CNN at 82.05%.

5.4 Improving generalisation by enabling inductive transfer

The second hypothesis stated advocated the simultaneous optimisation of several objectives as implemented in Multi-task learning. We now conduct experiments comparing such methods to pre-training/fine-tuning involving autoencoder architectures.

This will show whether a supervised objective can indeed help guide unsupervised learning to learn features relevant for the supervised task. Since all methods in this sec-

tion are applicable in both supervised and semi-supervised settings, we will investigate the results of their application in both scenarios. As before, we compare convolutional to recurrent and MLP networks, providing further empirical evidence of hypothesis (i).

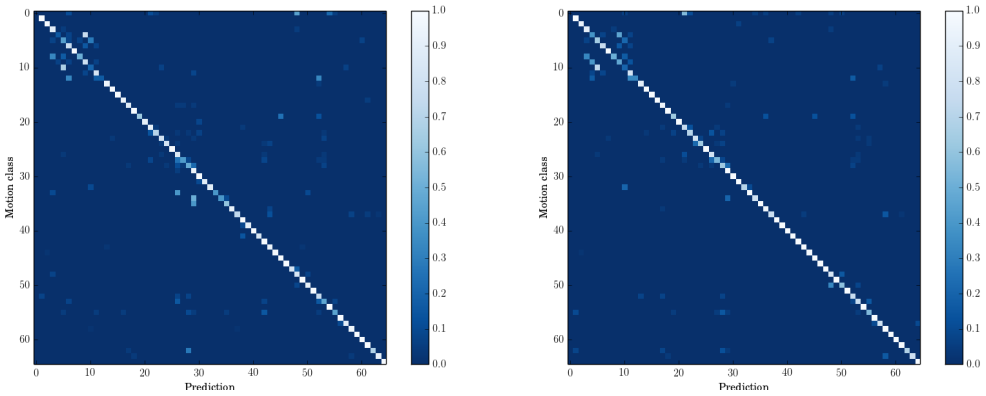
The baselines for the experiments in this section are (a) The recurrent encoder-decoder framework in (Harvey and Pal [26]) (which constitutes the current state-of-the-art results on HDM05) for sequential models. (b) The Multi-task MLP in (Cho and Chen [14]) which will help us compare non sequential methods. As we are not aware of any publications comparing the pre-training of auto-encoders, we also evaluate a stacked denoising autoencoder (SDAE) and a sparse denoising autoencoder (DAE).

The results of these experiments are shown in Table 5.3. Three main observations stand out: (1) The superior performance of networks trained in a Multi-task setting. (2) Improvement of all models in semi-supervised learning. (3) The poor performance of models which have been initialised with weights of a denoising autoencoder.

We show practical results of observations (1) and (2) in Figure 5.5. It shows the (normalised) confusion matrices of both the purely supervised CNN (82.05% accuracy) and a CNN trained as a Multi-task network on HDM05 and CMU. These confusion matrices show the high misclassification rates of both network for classes 4-14. Consulting Table 4 we notice that these are examples of fine-grained classes with only small differences (e.g. the height). Also, we found that Multi-task learning led to effects similar to regularisation This has also been observed in (Harvey and Pal [26]).

Again, we observe an constant improvement in performance through the application of sequential models. This provides further justifications of hypothesis (i).

Regarding (3), the visualisations of a representative selection of convolutional filters in the first layer provides an interesting insight. Figure 5.6 shows these filters for both the supervised CNN and the sparse DAE. Note that we show the weights after unsupervised pre-training of the DAE in order to see if the features obtained after unsupervised pre-training look useful. After observing clear temporal and inter-joint correlations in Figure 5.5b, we were first enthusiastic and expected an improvement considering the seemingly meaningless features learnt by the fully supervised model in Figure 5.5a. Considering the poor results of after DAE pre-training, however, this must be considered as a misleading intuition. While the filters of the CNN defy an intuitive interpretation, the clearly stronger better performance of the supervised method shows



(a) Supervised CNN (b) Semi-supervised Muli-Task CNN

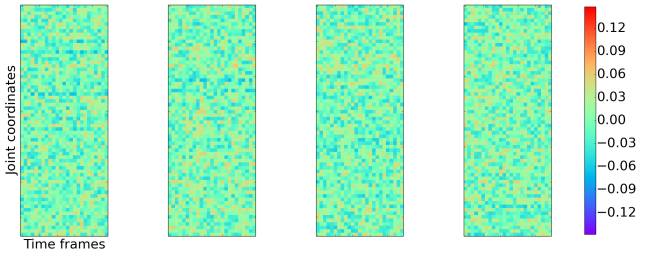
Fig. 5.5 (Normalised) confusion matrices of a CNN in normal and Multi-task training on HDM05.

	HDM05 Accuracy	HDM05 + CMU Accuracy
CNN	82.05 (±2.36)	-
MLP	80.08 (±2.56)	-
Recurrent Encoder-Decoder [26]	80.24 (±2.55)	**85.64** (±1.98)
Multi-task MLP [14][a]	81.64 (±2.41)	82.54 (±2.32)
Multi-task CNN	82.46 (±2.32)	85.07 (±2.04)
SDAE Pre-training (CNN)	75.25 (±2.99)	76.28 (±2.91)
DAE Pre-training (CNN)	71.04 (±3.31)	77.24 (±2.82)

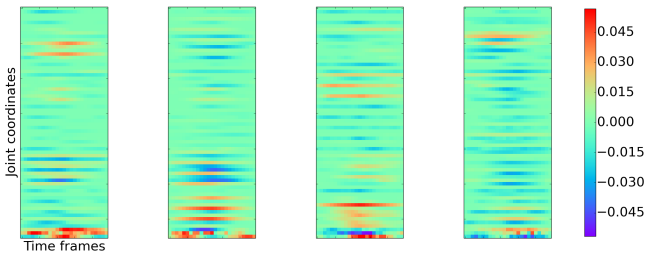
Table 5.3 Results of models trained in a semi-supervised fashion. We show results of approaches dependent on unsupervised pre-training and the performance of models trained in a Multi-task fashion.

[a] as reported in [26]

that apparently useful features learnt during unsupervised training do not appear useful for classification. We also observed significantly slower learning process after DAE pre-training. We argue that hence, the weight initialisation might correspond to a unfavourable position in weight-space. We also observed similar uninterpretable filters as those in Figure 5.5a for models with greater performance (e.g. the Multi-task CNN) or after significantly longer training time. Therefore, our best guess to this point is that features optimal for classification might not be humanly interpretable. Recall the supervised features shown in Figure 2.4 where we made a similar observation despite good performance.



(a) Supervised training



(b) Unsupervised pre-training

Fig. 5.6 Trained filters of the first convolutional layer. (a) Shows these filters after purely supervised training (b) Filters after unsupervised pre-training with a denoising autoencoder.

5.5 Motion analysis with deep generative models

We now evaluate the performance of deep generative models on the task. As explained in Chapter 4, VAEs can be utilised for semi-supervised learning by either using the latent features learnt during unsupervised learning or by implicitly incorporating labels in the learning task. Analogous to the argument for Multi-task CNNs over pre-training & fine-tuning, we motivated the second option due to the advantages of Multi-task learning. Again, we train a MLP-VAE (which is the original VAE definition) and compare it to the convolutional counterpart.

Table 5.4 shows both the classification accuracy as well as the unsupervised cost ($\mathcal{L}_{KL}$: Variational cost, $\mathcal{L}_{us}$: Reconstruction cost). The unsupervised cost for the MLP-VAE & CNN-VAE models is reported after pre-training.

	HDM05		HDM05 + CMU	
	$\mathcal{L}_{KL} + \mathcal{L}_{us}$	Accuracy	$\mathcal{L}_{KL} + \mathcal{L}_{us}$	Accuracy
MLP-VAE	0.043	55.08 (±3.97)	0.032	70.27 (±3.36)
CNN-VAE	0.012	77.79 (±2.78)	0.019	79.1 (±2.66)
Conv. Multi-task VAE	0.128	**87.3** (±1.78)	0.06	86.39 (±1.89)

Table 5.4 Performance of deep generative models on the HDM05 data set.

The results show several interesting properties: (1) Multi-task learning leads to a significant improvement, despite a high unsupervised cost. (2) While producing good results in the supervised case, the Conv. Multi-task VAE does not improve in a semi-supervised setting (3) The convolutional operation appears particularity useful in this generative framework (confirming hypothesis (i)). Also note that, even though the CNN-VAE fails to show competitive performance, it performs better than both of the pre-training/fine-tuning methods discussed before.

Regarding (1), we argue that this is due to two main factors:

- The supremacy of Multi-task learning over unsupervised pre-training: The previous section showed that both autoencoder variants were unable to learn useful features in an unsupervised fashion. The VAE experimetns show similar behaviour. It appears as though all models depend on a supervised training signal to optimise unsupervised feature learning. We also found that, similar to observations before, Multi-Task learning helped to distinguish between fine-grained classes (see Figure 5.10a).

- The prior on the latent space: By explicitly forcing the latent dimensions to be independent, the model successfully learnt to disentangle causal factors. None of the other Multi-task methods allowed the formulation of this explicit objective.

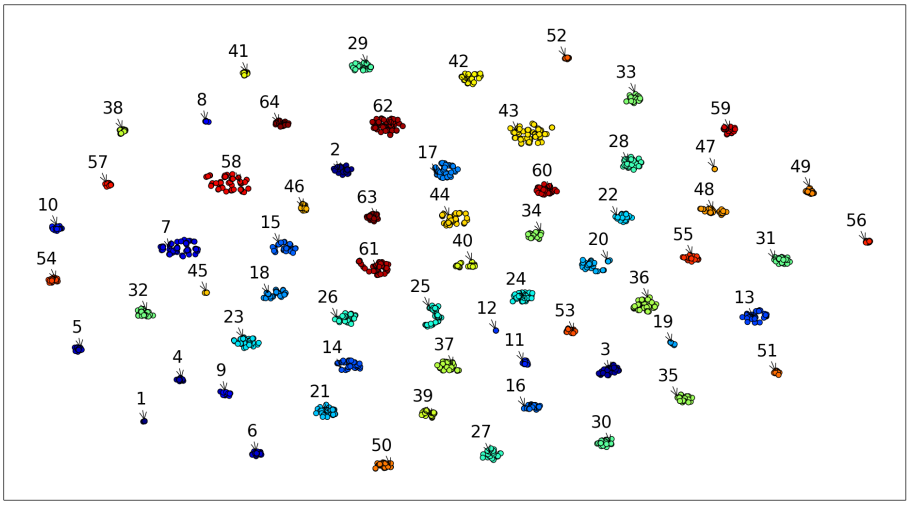
In order to help make these models more interpretable, we show a visualisation of the (dimensionality-reduced) latent space of the Mulit-Task VAE with t-SNE (Maaten and Hinton [49]) in Figure 5.7. First, note how particular clusters for the training+validation data correspond to individual classes. While performing the experiments, we first expected more non-linear transformations to be necessary in the classification branch. However, given that the classification accuracy during training was 100% for a linear softmax-regression classifier on the latent space, we can infer linear separability (as suggested by the plots). As expected, the results on the test set do not cluster as clearly, but most similar observations are nearby. This explains the good performance of the model. Note that, since t-SNE is a stochastic method, particular classes will in general, not appear in the same parts of the plot for two sequential runs. The clustering of the data however, is the important property which can be clearly observed for both cases.

In order to explain observation (2), we visualised the latent space of the conv. Multi-task VAE on both CMU and HDM05 (test) data. Rather than colouring each class, we distinguish only between CMU and HDM05 data. We observe that, in comparison to Figure 5.7b, the clusters of the HDM05 data are not as cleanly separated, explaining the decreased performance. Also, the vast majority of CMU data does not show much similarity to any HDM05 data points in the latent space. Clearly, the additional unlabelled does not help the model to learn more useful properties of the data. This might be due to the direct classification of the latent space. By forcing the latent space to correspond to linearly separable classes, this might limit the amount of additional information that might be learnt from the unlabelled data.

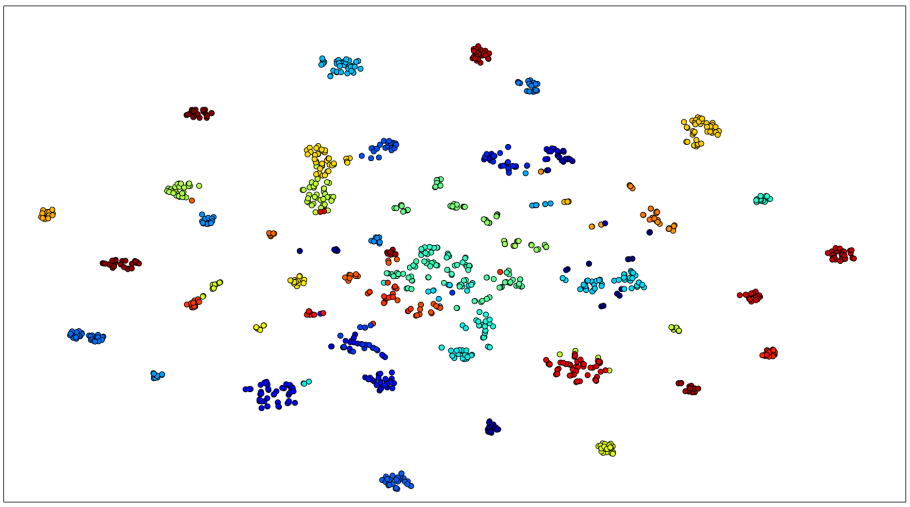
We discuss future research targeted at improving the conv. Multi-task VAE in semi-supervised settings in chapter 6.

5.5.1 Visualising the generative process

Due to the generative nature of the variational autoencoder, we can visualise its behaviour by sampling from the prior $p(\mathbf{z})$ and passing these samples through the decoder network. However, if this is done for a VAE trained in an unsupervised fashion, it is impossible to predict what motion each sample will correspond to (unless the latent space has been explored). In Multi-task learning, on the other hand, we explicitly force



(a) Training + Validation set



(b) Test set

Fig. 5.7 t-SNE visualisations of the latent space of a Conv. Multi-task VAE.

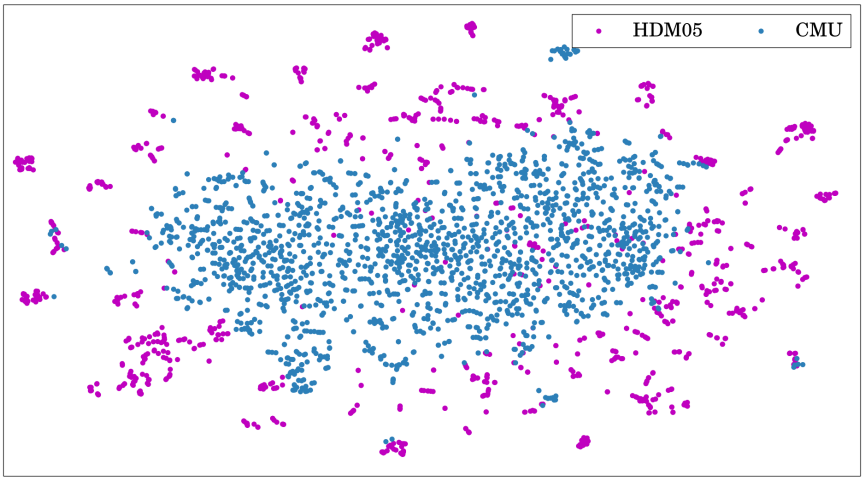


Fig. 5.8 HDM05 and CMU data in the latent space of a conv. Multi-task VAE.

the latent space to correspond to features useful for the classification task. In fact, the observations in Figure 5.7a, suggest the presence of clusters in the high-dimensional latent space. Thus, given a means to sample from each of the clusters, motions of a particular class can be generated.

In order to generate such samples, we fitted a mixture of 65 Gaussians (one for each class) to the latent space. The parameters of each of these distributions can be easily estimated by Maximum-Likelihood estimation given all instances of the class in latent space. Thus, a sample from a particular class-conditional Gaussian gives us insight into what the model is learning about each class. We show two such samples for class 45 ('Squat') in Figure 5.9.

While the generated motions are far from the quality of original HDM05 samples, the action in the motion can be clearly recognised. This shows that the latent space indeed captures enough information to make the motion clearly visible. Note that the poor motion quality is due to the early stopping in the final training. As shown in Table 5.4, the reconstruction error at this point was relatively high. We found however, that further training did not improve the classification accuracy even though the reconstruction error significantly decreased. Thus, we also expect the motion quality to improve with more training.

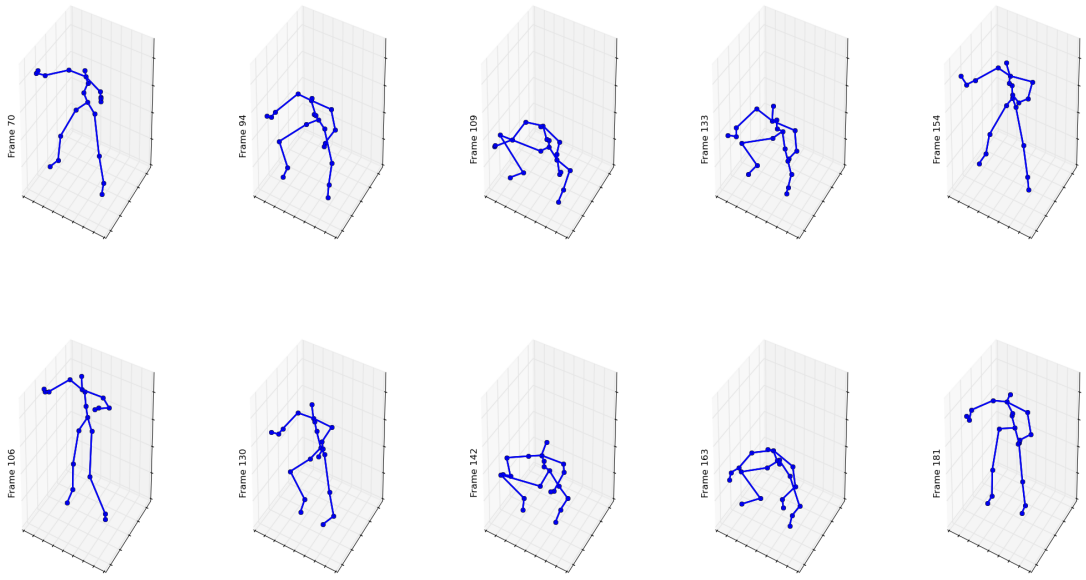


Fig. 5.9 Sampels from a convolutional Multi-task VAE. Shown are two instances of class 'squat'. We made a full animations available.[a] [b]

[a] http://www.jonathanschwarz.de/wp-content/uploads/2015/03/squad_1.gif
[b] http://www.jonathanschwarz.de/wp-content/uploads/2015/03/squad_2.gif

This method could be interesting for animation purposes, provided an improvement of the motion quality. The conv. Multi-task VAE would allow the creation of infinite samples of a particular motion. Animations of samples of other motions are shwon in Table 3.

5.5.2 Improving generalisation through Ensemble-classification

As a final experiment, we investigate the effect of ensemble predictions on the generalisation error. We do so as a means to further improve the performance of our final model and thus only apply this method to the conv. Multi-task VAE.

Ensembles are methods which combine the statistical strength of several probabilistic classifiers in a principled and simple way. Given the predictions of each individual method, the ensemble prediction simply corresponds to the mean of these all individual predictions. This can help improve the final performance in cases where each model makes different classification errors. Consider an example where method A incorrectly classifies a training example of a negative class with a low confidence of 54%. Another method B, however, fairly confidently predicts this instance (correctly) as positive with probability 88%. The mean of the predictions for the negative class is now merely 27%, thus the ensemble method correctly classify the instance as positive. Assuming each predictor makes slightly different mistakes, the ensemble method can help for a more accurate predictor.

Possible disadvantages of the method are increased training and testing times, since several models must be applied. Given the fast training and application of the Conv. Multi-task VAE, we argue that this negligible, provided a reasonable number of individual networks is applied as part of the ensemble.

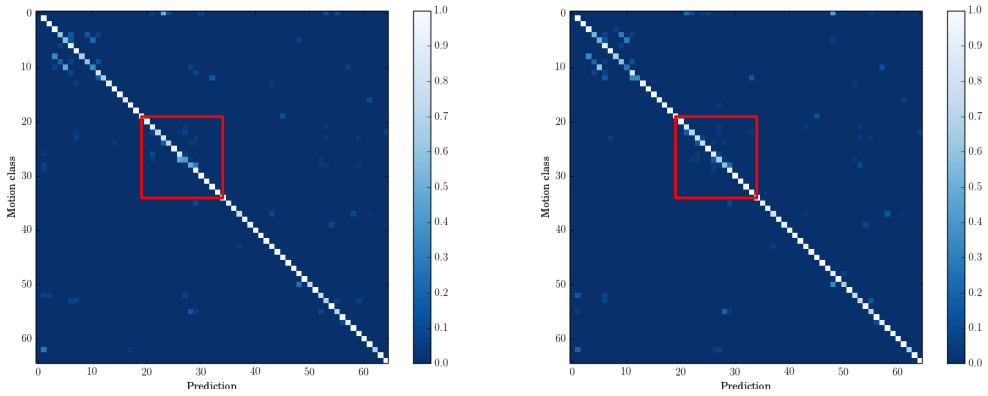
Application of neural network ensembles have been a well known trick for long (Krogh et al. [42]) and remains a practical method that is regularly applied (e.g. (He et al. [28])). Ensembles are in general only useful if individual predictors make different errors on a test set. This can easily happen in the training of neural networks when a different initialisation of the methods are chosen. We argue that the stochastic nature of variational autoencoders make this method particularly attractive, as individual networks are more likely to learn different latent features due to the sampling process.

Thus, we trained five conv. Multi-task VAEs with different random initialisations and evaluated their performance as an ensemble. As with other hyper-parameters, we

found the optimal number of networks through validation on a held-out set. Overall, we tested ensemble sizes of 2-10 networks. The individual performance of all models was within a range of [86.9%, 87.6%], making 87.3% a reasonable performance to report for a single network. The predictions of the networks mainly differed between classes 4-13 (deposit and grabbing actions) as well as classes 27-30 (different types of punching actions). As observed before, these classes were among the hardest to predict due to their strong similarity.

We found the ensemble to work well, resulting in an improvement of 0.7% over the individual predictor at 87.3%. This is higher than the accuracy of any of the individual networks (we trained 10 in total) were able to achieve. The confusion matrices between the network achieving 87.3% and the ensemble (Figure 5.10) also shows interesting patterns. Notice the areas highlighted with red rectangles. The ensemble is able to more accurately classify the instances of classes 27-30, explaining the slightly higher accuracy.

Thus our final model proposed as part of this thesis achieved an accuracy of 88% on HDM05.



(a) Conv. Multi-task VAE (b) Ensemble of 5 Conv. Multi-task VAEs

Fig. 5.10 (Normalised) Confusion matrices of deep generative models trained in a Multi-task setting.

Concluding the experiments carried out and providing a final overview, we summarise the best methods from each section in Table 5.5. The best overall model achieves an accuracy of 88%, which is an improvement of 2.36% to the current state-of-the-art in motion classification.

	HDM05 Test set Accuracy	HDM05 + CMU Test set Accuracy
Multi-task MLP [14]	81.64 (±2.41)	82.54 (±2.32)
Recurrent Encoder-Decoder [26]	80.24 (±2.55)	85.64 (±1.98)
Multi-task CNN	82.46 (±2.32)	85.07 (±2.04)
Conv. Multi-task VAE	87.3 (±1.78)	86.39 (±1.89)
Ensemble of 5 Conv. Multi-task VAEs	**88.0** (±1.70)	-

Table 5.5 An overview of results obtained in this thesis.

Chapter 6

Conclusions and Future work

In this thesis, we investigated different approaches to the problem of recognising and classifying human motions. Following recent research, we argued for a semi-supervised representation learning paradigm and evaluated several methods on the publicly available HDM05 data set. We conclude this work by summarising the performance of the approaches presented and give an outlook of future work in the area.

6.1 Sequential methods

In the first stage of our research, we considered the application of different neural network architectures in a purely supervised fashion. This provided us with a supervised baseline. We argued that, since motion classification is an inherently sequential problem, the choice of learning algorithm must be able to explicitly handle this property. As an example of architectures suitable for sequential data, we chose to compare convolutional and recurrent neural networks.

Both sequential methods showed considerably higher performance than a Multi-Layer Perceptron, which requires the treatment of a sequence as a single data point. We found that there was little difference in performance between the application of an LSTM network and a CNN. In these experiments, the best classification accuracy was achieved by the CNN, resulting in a performance of 82.05%.

We conclude that the application of sequential models is indeed an important consideration in the design of a system for motion classification. Following these encouraging conclusions, we designed all subsequent systems to explicitly handle sequential data. We furthermore found that, when comparing CNNs to LSTMs, there

seems to be no clearly superior model. Thus, the choice of architecture ought to be made by direct comparison on the respective motion data set. The consideration of further modules in the system may also be useful when making this decision.

6.2 Multi-task learning

Following the successful application of Multi-task learning architectures, we conducted experiments comparing such architectures to more traditional unsupervised pre-training and supervised fine-tuning schemes. Confirming previous research, we found that Multi-task learning could be applied successfully to motion classification. In the direct comparison to denoising and stacked-denoising autoencoders, Multi-task learning performed significantly better. We also found the Multi-task models capable of improving generalisation when unlabelled data was added. Surprisingly, the unsupervised pre-training schemes *performed worse* in comparison to the fully supervised methods, both when using only labelled and additionally labelled data. We compared the weight-matrices of the models and found clear differences between those models trained with access to the labelled data in comparison to the features learnt in mere unsupervised training. In agreement with previous results, we found sequence models to outperform MLPs-Layer perceptrons.

We conclude that guiding unsupervised learning through a supervised training objective significantly increases the performance of all architectures introduced. We found this to be the case for both purely supervised and semi-supervised training. Our best model at this stage was a CNN trained in a Multi-task fashion on both the HDM05 and CMU data sets. With an accuracy of 85.07%, the performance at this stage was only slightly worse in comparison to the current state-of-the-art.

6.3 Deep generative models

In the final stage of research, we investigated the performance of variational autoencoders, so-called deep generative models. We evaluated these models in two schemes: 1) The learning of latent features from unlabelled examples and the subsequent classification of the data in the latent space (this is equivalent to the autoencoder schemes without fine-tuning). 2) The simultaneous optimisation of unsupervised, variational and supervised objectives. Thus, this falls in the category of Multi-task learning.

We found the models trained in scheme 1) to perform rather poorly, resulting in clearly worse results in comparison to the supervised baseline. We thus conclude that the application of learning algorithms involving unsupervised pre-training are not successfully applicable to motion analysis.

Training scheme 2) on the other hand, lead to excellent performance. With a classification accuracy of 87.3% we achieved an improvement of the current state-of-the-art in classification accuracy on HDM05. We found that dimensionality-reduced visualisations of the latent space clearly corresponded to linearly separable classes, which was in agreement with a training accuracy of 100%. In addition, the formulation of generative model appears to allow for easier separability of the classes. Finally, the convolutional Multi-task VAE allows the generation of data points of particular classes. This could be of interest in motion synthesis.

An open question to this point is how the convolutional Multi-task VAE can be improved with additional unlabelled data. In the experiments carried out, we found that training the model in a semi-supervised setting failed to improve the performance. We propose future research targeted to address this problem in section 6.4.

As a final model, we proposed an ensemble of five convolutional Multi-task VAEs, arguing that this may help to reduce the miss-classification rate for fine-grained classes. This resulted in a **final performance of 88.0%**.

6.4 Future work

An important consideration for future work is the improvement of the convolutional Multi-task VAE for semi-supervised settings. As observed in section 5.5, the introduction of additional CMU data slightly *decreased the performance of the model*. We argued that while forcing the labelled space to correspond to linearly separable classes, this might limit the amount of additional information which can be extracted from unlabelled data. Thus, we propose to learn two separate sets of latent variables $\mathbf{z}_{us}, \mathbf{z}_s$. $\mathbf{z}_s$ are learnt using only the labelled data and serve as the input to the linear classifier. Variables $\mathbf{z}_{us}$ on the contrary, are trained using both the unlabelled and labelled data without a connection to the classifier. The decoding branch of the network is then to reconstruct the input given the concatenation of the two variables. This may allow the network to learn features useful for the classification task ($\mathbf{z}_s$) while simultaneously

extracting additional information in the unsupervised latent space ($\mathbf{z}_{us}$). During test time, an external classifier is applied given access to both sets of variables.

The following further avenues of research may lead to improvements in both the performance as well as the applicability of the proposed models:

- Along with variational autoencoders, an architecture denoted generative adversarial network (GAN, Goodfellow et al. [24]) has led to an increased interest in deep generative models (e.g. Denton et al. [17], Dosovitskiy et al. [19]). GANs are based on the idea of optimising the generative model by deceiving a discriminator which has been trained to distinguish between real and generated data. This has been shown to be a powerful idea for unsupervised feature learning (Radford et al. [57]). An extension of this idea with the notion of multi-task learning in mind could lead to improvements of the Multi-task CNN.

- In a similar vein, the combination of GANs with VAEs (Larsen et al. [44]) has led to promising results in motion synthesis (Ikhansul [35]). The combination with Convolutional Multi-task-VAEs could be useful for motion synthesis. This could help increase the quality of sampled motions of a particular class.

- In (Rasmus et al. [59]), the authors argue for skip-connections in deep autoencoders, as this may allow deep layers to focus on more abstract features during reconstruction. The addition of skip-connections to the Convolutional Multi-task VAE could be a simple yet promising experiment. The idea to combine such autoencoders with skip-connections has also been explored in (Sønderby et al. [63]), leading to good results.

- Finally, we observed numerical instability in stochastic networks following small changes to certain parts of the architecture (e.g. the learning rate or the activation function). In order to tackle these issues, we encourage the exploration of changes proposed by (Maaløe et al. [48]): The authors found that temperature weighting between discriminative and stochastic training objectives resulted in more stability. Moreover, the application of Batch-normalisation (Ioffe and Szegedy [36]) has been shown to help with with such issues.

References

[1] Altman, N. S. (1992). An introduction to kernel and nearest-neighbor nonparametric regression. *The American Statistician*, 46(3):175–185.

[2] Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.

[3] Bellman, R. (1956). Dynamic programming and lagrange multipliers. *Proceedings of the National Academy of Sciences*, 42(10):767–769.

[4] Bengio, I. G. Y. and Courville, A. (2016). Deep learning. Book in preparation for MIT Press.

[5] Bengio, Y. (2009). Learning deep architectures for ai. *Foundations and trends® in Machine Learning*, 2(1):1–127.

[6] Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828.

[7] Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

[8] Bottou, L. and Bousquet, O. (2008). The tradeoffs of large scale learning. In Platt, J., Koller, D., Singer, Y., and Roweis, S., editors, *Advances in Neural Information Processing Systems*, volume 20, pages 161–168. NIPS Foundation (http://books.nips.cc).

[9] Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological cybernetics*, 59(4-5):291–294.

[10] Burda, Y., Grosse, R., and Salakhutdinov, R. (2015). Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

[11] Caruana, R. (1998). Multitask learning. In *Learning to learn*, pages 95–133. Springer.

[12] Chen, X. and Koskela, M. (2013a). Classification of rgb-d and motion capture sequences using extreme learning machine. In *Scandinavian Conference on Image Analysis*, pages 640–651. Springer.

[13] Chen, X. and Koskela, M. (2013b). Classification of rgb-d and motion capture sequences using extreme learning machine. In *Scandinavian Conference on Image Analysis*, pages 640–651. Springer.

[14] Cho, K. and Chen, X. (2014). Classifying and visualizing motion capture sequences using deep neural networks. In *Computer Vision Theory and Applications (VISAPP), 2014 International Conference on*, volume 2, pages 122–130. IEEE.

[15] Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

[16] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255. IEEE.

[17] Denton, E. L., Chintala, S., Fergus, R., et al. (2015). Deep generative image models using a laplacian pyramid of adversarial networks. In *Advances in neural information processing systems*, pages 1486–1494.

[18] Doersch, C. (2016). Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

[19] Dosovitskiy, A., Tobias Springenberg, J., and Brox, T. (2015). Learning to generate chairs with convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1538–1546.

[20] Du, Y., Wang, W., and Wang, L. (2015). Hierarchical recurrent neural network for skeleton based action recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1110–1118.

[21] Erhan, D., Bengio, Y., Courville, A., Manzagol, P.-A., Vincent, P., and Bengio, S. (2010). Why does unsupervised pre-training help deep learning? *Journal of Machine Learning Research*, 11(Feb):625–660.

[22] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.

[23] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In *Aistats*, volume 15, page 275.

[24] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial nets. In *Advances in Neural Information Processing Systems*, pages 2672–2680.

[25] Graves, A. and Schmidhuber, J. (2005). Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610.

[26] Harvey, F. G. and Pal, C. (2015a). Semi-supervised learning with encoder-decoder recurrent neural networks: Experiments with motion capture sequences. *arXiv preprint arXiv:1511.06653*.

[27] Harvey, F. G. and Pal, C. (2015b). Semi-supervised learning with encoder-decoder recurrent neural networks: Experiments with motion capture sequences. *arXiv preprint arXiv:1511.06653*.

[28] He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*.

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

[30] Hinton, G. E. and Zemel, R. S. (1994). Autoencoders, minimum description length, and helmholtz free energy. *Advances in neural information processing systems*, pages 3–3.

[31] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[32] Holden, D., Saito, J., and Komura, T. (2015a). A deep learning framework for character motion synthesis and editing. *IEEE Transactions on Visualization and Computer Graphics*, 21:1.

[33] Holden, D., Saito, J., Komura, T., and Joyce, T. (2015b). Learning motion manifolds with convolutional autoencoders. In *SIGGRAPH Asia 2015 Technical Briefs*, page 18. ACM.

[34] Huang, G.-B., Zhou, H., Ding, X., and Zhang, R. (2012). Extreme learning machine for regression and multiclass classification. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 42(2):513–529.

[35] Ikhansul, H. (2016). Human motion synthesis using deep generative models. Master's thesis, The University of Edinburgh.

[36] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*.

[37] Joachims, T. (1999). Transductive inference for text classification using support vector machines. In *ICML*, volume 99, pages 200–209.

[38] Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

[39] Kingma, D. P., Mohamed, S., Rezende, D. J., and Welling, M. (2014). Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

[40] Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

[41] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[42] Krogh, A., Vedelsby, J., et al. (1995). Neural network ensembles, cross validation, and active learning. *Advances in neural information processing systems*, 7:231–238.

[43] Larochelle, H., Bengio, Y., Louradour, J., and Lamblin, P. (2009). Exploring strategies for training deep neural networks. *Journal of Machine Learning Research*, 10(Jan):1–40.

[44] Larsen, A. B. L., Sønderby, S. K., and Winther, O. (2015). Autoencoding beyond pixels using a learned similarity metric. *arXiv preprint arXiv:1512.09300*.

[45] Le Cun, Y. (1987). *Modèles connexionnistes de l'apprentissage*. PhD thesis, Paris 6.

[46] LeCun, Y. et al. (1989). Generalization and network design strategies. *Connectionism in perspective*, pages 143–155.

[47] Liu, G., Zhang, J., Wang, W., and McMillan, L. (2005). A system for analyzing and indexing human-motion databases. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 924–926. ACM.

[48] Maaløe, L., Sønderby, C. K., Sønderby, S. K., and Winther, O. (2016). Auxiliary deep generative models. *arXiv preprint arXiv:1602.05473*.

[49] Maaten, L. v. d. and Hinton, G. (2008). Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605.

[50] MacQueen, J. et al. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA.

[51] Müller, M. and Röder, T. (2006). Motion templates for automatic classification and retrieval of motion capture data. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 137–146. Eurographics Association.

[52] Müller, M., Röder, T., and Clausen, M. (2005). Efficient content-based retrieval of motion capture data. In *ACM Transactions on Graphics (TOG)*, volume 24, pages 677–685. ACM.

[53] Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., and Weber, A. (2007). Documentation mocap database hdm05. Technical Report CG-2007-2, Universität Bonn.

[54] Murray, I. A. (2007). *Advances in Markov chain Monte Carlo methods*. University of London.

[55] Pearson, K. (1901). Liii. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572.

[56] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[57] Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

[58] Raghavan, U. N., Albert, R., and Kumara, S. (2007). Near linear time algorithm to detect community structures in large-scale networks. *Physical review E*, 76(3):036106.

[59] Rasmus, A., Berglund, M., Honkala, M., Valpola, H., and Raiko, T. (2015). Semi-supervised learning with ladder networks. In *Advances in Neural Information Processing Systems*, pages 3546–3554.

[60] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

[61] Shotton, J., Sharp, T., Kipman, A., Fitzgibbon, A., Finocchio, M., Blake, A., Cook, M., and Moore, R. (2013). Real-time human pose recognition in parts from single depth images. *Communications of the ACM*, 56(1):116–124.

[62] Smolensky, P. (1986). Information processing in dynamical systems: Foundations of harmony theory. Technical report, DTIC Document.

[63] Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. (2015). Ladder variational autoencoders. *arXiv preprint arXiv:1511.06653*.

[64] Sutskever, I., Martens, J., Dahl, G. E., and Hinton, G. E. (2013). On the importance of initialization and momentum in deep learning. *ICML (3)*, 28:1139–1147.

[65] Sutskever, I., Martens, J., and Hinton, G. E. (2011). Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

[66] Tibshirani, R. (1996). Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288.

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

[68] Xia, L., Chen, C.-C., and Aggarwal, J. K. (2011). Human detection using depth information by kinect. In *CVPR 2011 WORKSHOPS*, pages 15–22. IEEE.

[69] Zhou, Y. and Chellappa, R. (1988). Computation of optical flow using a neural network. In *Neural Networks, 1988., IEEE International Conference on*, pages 71–78. IEEE.

[70] Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., and Xie, X. (2016). Co-occurrence feature learning for skeleton based action recognition using regularized deep lstm networks. *arXiv preprint arXiv:1603.07772*.

Appendix

Model specifications

MLP	CNN	Multi-Task MLP
Fully connected (512)	1D-Convolution (64, 25)	**Encoder**
ReLU activation	ReLU activation	Fully connected (1000)
Fully connected (256)	1D-Maxpooling(Window=2)	ReLU activation
ReLU activation	1D-Convolution (128, 25)	Fully connected (500)
Fully connected (128)	ReLU activation	ReLU activation
ReLU activation	1D-Maxpooling(Window=2)	**Classifier**
Classifier	1D-Convolution (256, 25)	Fully connected (65)
Fully connected (65)	ReLU activation	Softmax activation
Softmax activation	1D-Maxpooling(Window=2)	**Decoder**
	Fully connected (500)	Inverse of Encoder
	ReLU activation	
	Classifier	
	Fully connected (65)	
	Softmax activation	

Multi-Task CNN	SDAE Pre-training (CNN)	DAE Pre-training (CNN)
Encoder	(same as Multi-Task CNN	(Same as CNN)
(same as CNN	without classification)	
without classifier)		
Classifier		
Fully connected (65)		
Softmax activation		
Decoder		
Inverse of Encoder		

Table 1 Specifications of discriminative models trained during the experiments. We write

MLP-VAE	CNN-VAE	Conv. Multi-Task VAE
Encoder	**Encoder**	**Encoder**
Fully connected (1000)	(same as CNN	(same as CNN-VAE)
ReLU activation	without classifier)	**Classifier**
Fully connected (500)	Fully connected (500)	Fully connected (65)
ReLU activation	ReLU activation	Softmax activation
Variational layer	**Variational layer**	**Variational layer**
Estimation of $\boldsymbol{\mu}_z$:	(same as MLP-VAE)	(same as MLP-VAE)
Fully connected (150)	**Decoder**	**Decoder**
Estimation of $\boldsymbol{\sigma}_z^2$:	Inverse of Encoder	Inverse of Encoder
Fully connected (150)		
$\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_z, \boldsymbol{\sigma}_z^2)$:		
Decoder		
Inverse of Encoder		

Table 2 Specifications of generative models trained during the experiments

Additional samples from a convolutional Multi-Task VAE

Note: Please add the rest of the URL in Table 3 to the follwoing prefix:
`http://www.jonathanschwarz.de/wp-content/uploads/2015/03/`

Description	URL
rotateArmsLForward (1)	rotate_1.gif
rotateArmsLForward (2)	rotate_2.gif
squat (1)	squad_1.gif
squat (2)	squad_2.gif
squat (3)	squad_3.gif
jogging (1)	jogging_1.gif
jogging (2)	jogging_2.gif

Table 3 Samples from a convolutional Multi-Task VAE.

Motion classes in the HDM05 data base

Table 4 shows a list of motion classes in the evaluation of this thesis.

Index	Motion	Index	Motion
1	cartwheel	36	rotateArmsRForward
2	clap	37	runOnPlace
3	clapAbove	38	shuffle
4	depositFloorR	39	sitDownChair
5	depositHighR	40	sitDownFloor
6	depositLowR	41	sitDownKneelTieShoes
7	depositMiddleR	42	sitDownTable
8	elbowToKnee	43	ski
9	grabFloorR	44	sneak
10	grabHighR	45	squat
11	grabLowR	46	staircaseDown3Rstart
12	grabMiddleR	47	standUpKneelToStand
13	hitRHandHead	48	standUpLieFloor
14	hopBothLegs	49	standUpSitC
15	hopLLeg	51	standUpSitF
16	hopRLeg	52	standUpSitT
17	jogLeftCircle	53	throwBasket
18	jogOnPlace	54	throwFarR
19	jogRightCircle	55	throwSitting
20	jumpDown	56	throwStanding
21	jumpingJack	57	turnLeft
22	kickLFront	58	turnRight
23	kickLSide	59	walk
24	kickRFront	60	walkBackwards
25	kickRSide	61	walkLeft
26	lieDownFloor	62	walkLeftCircle
27	punchLFront	63	walkOnPlace
28	punchLSide	64	walkRightCircle
29	punchRFront	65	walkRightCross
30	punchRSide		
31	rotateArmsBothBackward		
32	rotateArmsBothForward		
33	rotateArmsLBackward		
34	rotateArmsLForward		
35	rotateArmsRBackward		

Table 4 Motion classes in the HDM05 data base [53] and their corresponding class indices. We have merged repetitions of single motions into a single class as proposed in (Cho and Chen [14]).