DEGREE PROJECT IN COMPUTER SCIENCE AND ENGINEERING,
SECOND CYCLE, 30 CREDITS
STOCKHOLM, SWEDEN 2017

Perceptual facial expression representation

OLGA MIKHEEVA

KTH ROYAL INSTITUTE OF TECHNOLOGY
SCHOOL OF COMPUTER SCIENCE AND COMMUNICATION

Perceptual facial expression representation

OLGA MIKHEEVA

Master in Machine Learning
Date: November 7, 2017
Supervisor: Hedvig Kjellström
Examiner: Patric Jensfelt
School of Computer Science and Communication

Abstract

Facial expressions play an important role in such areas as human communication or medical state evaluation. For machine learning tasks in those areas, it would be beneficial to have a representation of facial expressions which corresponds to human similarity perception.

In this work, the data-driven approach to representation learning of facial expressions is taken. The methodology is built upon Variational Autoencoders and eliminates the appearance-related features from the latent space by using neutral facial expressions as additional inputs. In order to improve the quality of the learned representation, we modify the prior distribution of the latent variable to impose the structure on the latent space that is consistent with human perception of facial expressions.

We conduct the experiments on two datasets and the additionally collected similarity data, show that the human-like topology in the latent representation helps to improve the performance on the stereotypical emotion classification task and demonstrate the benefits of using a probabilistic generative model in exploring the roles of latent dimensions through the generative process.

Sammanfattning

Ansiktsuttryck spelar en viktig roll i områden som mänsklig kommunikation eller vid utvärdering av medicinska tillstånd. För att tillämpa maskininlärning i dessa områden skulle det vara fördelaktigt att ha en representation av ansiktsuttryck som bevarar människors uppfattning av likhet.

I det här arbetet används ett data-drivet angreppssätt till representationsinlärning av ansiktsuttryck. Metodologin bygger på s. k. Variational Autoencoders och eliminerar utseende-relaterade drag från den latenta rymden genom att använda neutrala ansiktsuttryck som extra input-data. För att förbättra kvaliteten på den inlärda representationen så modifierar vi a priori-distributionen för den latenta variabeln för att ålägga den struktur på den latenta rymden som är överensstämmande med mänsklig perception av ansiktsuttryck.

Vi utför experiment på två dataset och även insamlad likhets-data och visar att den människolika topologin i den latenta representationen hjälper till att förbättra prestandan på en typisk emotionsklassificeringsuppgift samt fördelarna med att använda en probabilistisk generativ modell när man undersöker latenta dimensioners roll i den generativa processen.

Contents

Notations

$\mathbf{X} \in \mathbb{R}^D$ - input data

$\mathbf{Y} \in \mathbb{R}^D$ - additional input (neutral faces)

$\mathbf{Z} \in \mathbb{R}^K$ - latent space

D - input data dimensionality

K - latent dimensionality

N - size of a data set

T - size of a triplet data set

B - batch size

V - triplet batch size

L - number of samples

M - number of samples (for the triplet part)

Chapter 1

Introduction

Any machine learning or data analysis method highly depends on the quality of the input data. The quality does not only concern noise or missing data, but also, and often, more importantly, the representation. Representation includes the dimensionality of the data, type of the features (discrete, continuous, ordinal, etc.), redundancy, properties of the input space, such as a correlation between the features, the manifold of the data within the space and so on. It is often beneficial to transform the input data into a different representation before feeding it as input for a model (e.g classifier). The class of methods used to learn this preliminary transformation is called "representation learning". The resulting representation is usually lower-dimensional than the original one and tries to capture the underlying structure in the original data.

Facial expressions play an important role in such areas as human communication or medical state evaluation. For machine learning tasks in those areas, it would be beneficial to have a representation of facial expression that corresponds to human perception, i.e. similarity measures of those latent representations should cohere with similarity evaluation of facial expressions by a human. In this project, we are interested in finding this useful representation of facial expressions.

Raw input data are usually high-dimensional (images, videos, geometric facial features) for the applications where facial expressions are of interest. Such dimensionality is clearly redundant. Facial Action

Coding System (FACS) is a commonly used representation of facial expressions, that is based on manually selected features corresponding to facial muscle activations (for more details see section 2.1). As feature engineering might not result in an optimal representation, in this work we take a different path to learning latent representation and use a fully data-driven approach to learn a low-dimensional representation that would capture all major variations in data while also giving some desirable properties to the latent embeddings.

1.1 Research question

In facial expressions, geometric distances often do not directly correspond to how humans perceive those expressions, e.g. two smiling faces with different degree of smiling convey much more similar feelings than a sarcastic smile and a genuine smile even if the two latter are closer geometrically. It is our hypothesis, that incorporating human expertise in our model as a prior knowledge for the latent space can guide latent representation to be more similar to that of a human and therefore allow for easier interpretation and potentially higher efficiency in further usage.

The main contribution of this work is a model that incorporates human knowledge into the topology of the latent representation. An additional contribution is the collected data set of partial similarity rankings in the form of triplets (see 5.2).

In this project, we use facial landmarks already available in the data set as initial facial expression feature vectors. Methods for extracting facial landmarks lie outside of the scope of this project and are not discussed.

1.2 Ethics, societal aspects and sustainability

This particular work is quite technical and concerns representation learning of facial expression. Nevertheless, the purpose of any representation learning is to further use it as a building block in a possibly wide range of tasks. In the case of facial expressions, those tasks include facial expression recognition and classification, mood detection, health state assessment, non-verbal communication and much more.

Some of the target tasks may be highly beneficial for society, for example, detection of depression, stroke or other diseases. At the same time, we should be aware that systems with such functionality would require a lot of data with high privacy concerns (mostly video), which should be handled and stored with great care since it can not be fully anonymized.

Another ethical consideration is that the data collected for a different reason can be used to detect people's emotions and reactions without their knowledge or consent (e.g. a security camera in a supermarket could be used to detect how people react to a specific product).

A more controversial area that relies on facial recognition concerns monitoring and tracking systems. Those systems can be of a great value for the police in enforcing the law (e.g. when searching for the suspect). At the same time, there is a danger of abuse of such a powerful instrument and a clear privacy concern.

In our opinion, sustainable use of technologies based on facial features can have positive societal impact. However, to ensure long-term sustainability and the continuous improvement of the methods the right balance should be reached between privacy and potential benefits.

1.3 Overview of the thesis

The structure of the thesis is as follows. We begin with the overview of the related work including state of the art research in Chapter 2 and choose the primary method for our task, Variational Autoencoder (VAE). Chapter 3 starts with some necessary background information on neural networks and autoencoders, and, finally, a thorough introduction to VAE is given. The developed methodology built upon VAE is presented in Chapter 4. We describe the two data sets used in the training and the process of additional data collection (to acquire perceptual similarity data using crowd-sourcing) in Chapter 5. All the experiments, including the evaluation metrics, parameters and configurations used for training, and the results on both data sets are presented in Chapter 6. In Chapter 7 we summarise the results and discuss future work.

Chapter 2

Related Work

The classic approach to feature extraction for facial expression related applications (e.g. automatic classification) used to be feature engineering with the most common one being action unit detection based on facial geometry (Facial Action Coding System) [10], [23]. Lately with the increasing popularity of deep neural networks there appeared some research using convolutional neural networks for the task of facial expression recognition [21]. The main goal of this project is to find a facial representation with similar properties as that of a human in a fully data-driven manner using latest advances in deep representation learning, therefore, if successfully completed, combining best properties of the two approaches, i.e. human-like (possibly interpretable) features without explicit feature engineering.

2.1 Facial Action Coding System

Facial Action Coding System (FACS) is a system that taxonomizes human facial movements, developed by Ekman and Friesen [10]. Using this system facial expression can be described using a set of facial action units (AU), where each action unit corresponds to some muscle movement (e.g. "Inner Brow Raiser", "Upper Lip Raiser", etc.). There are 28 main action units, but the full list is about 100 including head and eye movement. The intensity of activations is measured on a discrete 5-level scale (A-E). Nowadays FACS is a commonly accepted standard to represent and further physical expression of emotion.

This approach to representation falls into the category of feature engineering because all the features (action units in this case) were manually chosen to represent facial expressions. Automatic detection of AUs is possible and some systems for detection from videos have been developed (usually ruled-based on facial geometry), but they often only detect a subset of action units.

The FACS representation is manually engineered and might not be the optimal one, it is also discrete which might not be ideal for some tasks. In this work, we want to take another approach, a fully data-driven one, and using unsupervised machine learning techniques for representation learning find a continuous representation with the similarity measure on latent space coherent with human perception of facial expressions.

2.2 Data-driven approach

The simplest methods for representation learning include linear dimensionality reduction methods, such as Principal Component Analysis (PCA). However, the representation power of simple linear methods is very limited and often is not enough to get latent representation with desirable properties such as smoothness, temporal and spatial coherence, disentanglement, sparsity, high-level abstract features [2].

With the popularity of deep learning methods, the two major tracks in representation learning research have been: the one based on probabilistic models and the one based on neural networks. The main difference is whether the deep layered architecture is interpreted as a computational graph or a probabilistic graphical model. The neural network approach has been mostly represented by a different variations of autoencoders (sparse, denoising, etc.) [13], [27]. Main examples of the probabilistic approach are Restricted Boltzmann Machines, Gaussian process latent variables models (gp-lvm) and Deep Gaussian processes (stacked bayesian GP-LVMs) [25], [7].

Lately, as the models get deeper and the inference in probabilistic models becomes increasingly more complicated and usually even intractable, there seems to be a trend to merge those two approaches by using neural networks as an approximation mechanism for the inference.

2.2.1 Variational autoencoders

Variational autoencoder framework is the prime example of this emerging trend [16], [9]. Even though it is called an "autoencoder", fundamentally it is a probabilistic directed graphical model with latent random variables and observed random variables. Generating process is modelled as a function of latent variable $\mathbf{z}$ via a neural network with added Gaussian noise:

$$\mathbf{x} \sim \mathcal{N}(f(\mathbf{z}; \theta), \sigma^2)$$

where f is a neural network with parameters θ. Prior over latent space is usually chosen to be a spherical Gaussian, but in principle, a different distribution can be chosen if satisfies some constraints (for more details see [16]).

The exact inference over latent variables in such model is intractable and the classic approach is a variational inference that allows finding an analytical approximation to the posterior distribution of latent variables. VAE framework uses a neural network to approximate parameters of a posterior, e.g. in case of a Gaussian posterior mean and variance are computed.

$$p(\mathbf{z}|\mathbf{x}) \approx q(\mathbf{z}|\mathbf{x}) \sim \mathcal{N}(g(\mathbf{x}; \phi))$$

where g is a neural network. In contrast to Gaussian Process models (e.g. gp-lvm, deepGP) where the number of variational parameters grows with the number of data points, approximating posterior as a function of observed variables has a clear advantage in scalability. The model is trained jointly (both generating and inference network) by maximizing evidence lower bound (ELBO) on the whole data set

$$\log p(\mathbf{X}) \geq \mathcal{L}(\mathbf{X}) = E_{\mathbf{Z} \sim q(\mathbf{Z}|\mathbf{X})}[\log p(\mathbf{X}|\mathbf{Z})] - KL(q(\mathbf{Z}|\mathbf{X}) || p(\mathbf{Z}))$$

via backpropagation. The name of the model is therefore based on the fact that it has inference and generating networks similar to encoding and decoding networks in classic autoencoders.

The variational autoencoder framework has become a foundation for much other related research. The adversarial autoencoder is using an additional adversarial network rather than a Kullback–Leibler divergence to incorporate a prior on the latent space [18]. Importance weighted autoencoder modifies the model objective to get a tighter lower bound [4]. Ladder variational autoencoder uses an improved inference mechanism for models with more than one layer of latent random variables [24]. Even the Gaussian Processes research community came up with a modification of deep GPs that uses a neural network as inference mechanism [6].

Another recent line of research that the VAE can benefit from is addressing the problem of using simple posterior distributions for variational inference by specifying more flexible, complex and scalable approximate posterior distributions using normalizing flow and inverse autoregressive flow [15], [22]. In contrast, Higgins et al. [12] are advocating for the importance of disentanglement and modify the VAE force the approximated posterior to be closer to prior by putting much more weight on the KL-divergence term.

2.2.2 Topology

Naturally, for many applications there is some prior knowledge about the topology (e.g. smoothness or similarity preservation), and it would be clearly beneficial to incorporate this knowledge into the model. Urtasun et al. [26] do this for the modelling of human motion tasks with Gaussian processes by putting explicit constraints on the embedding, more specifically they formulate the prior in the form of

$$p(\mathbf{Z}) \propto e^{-\frac{1}{\gamma}\Phi(\mathbf{Z})}$$

where $\Phi(\mathbf{Z})$ is the energy function modelling specific topological constraints.

In the neural network approach to representation learning it is also possible to impose such constraints by modifying the objective (adding penalizing term for violation of the constraints), but often the most challenging part is to modify a network with minimum loss in computational capacity. For the case of partial similarity measure constraints in the form of triplets (exactly the case in this project) Hoffer and Ailon [14] propose a "triplet network", that has a three part of the network sharing weights and then additional layer for distance comparison (figure 2.1).

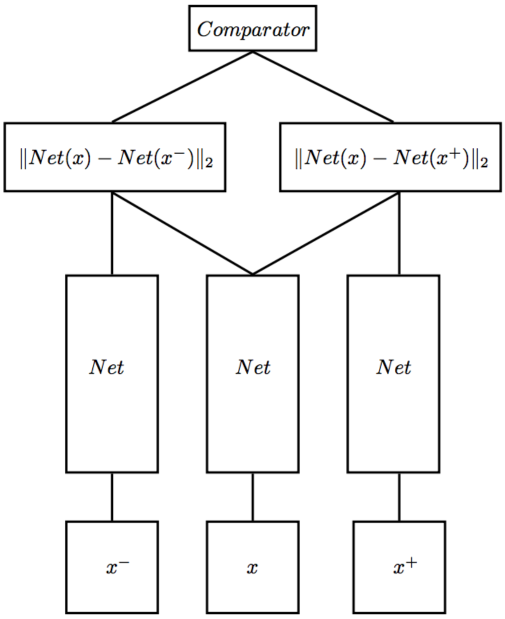


Figure 2.1: "Triplet network" architecture (from [14]).

Based on the literature study shortly summarized above, Variational Autoencoder Framework (VAE) has been chosen to be the primary class of methods to focus on in this project. There has not been any work on incorporating topological constraints in the VAE framework, but combining the two described approaches seems feasible.

Chapter 3

Background

After conducting the literature study (see chapter 2) variational autoencoder (VAE) was chosen as a primary method for learning a latent representation of facial expressions. VAE is a probabilistic model, but similar in spirit to standard non-probabilistic autoencoder. Both models use neural networks as their encoders and decoders. With those connections in mind, a short introduction to neural networks and autoencoders will be given in this chapter before more detailed description of VAE on which the methodology for this project is build upon.

3.1 Neural networks

The feed-forward neural network is a model that can be described as a series of functional transformations. First, the M linear combinations of the input vector $x = (x_1, x_2, ..., x_D)$ constructed:

$$a_j^{(1)} = \sum_{i=1}^{D} w_{j,i}^{(1)} x_i + w_{j,0}^{(1)}, \forall j = 1, .., M \tag{3.1}$$

where the superscript (1) denotes the first layer of the network.

Then each activation a_j is transformed using a differentiable, non-linear transformation function $z_j = h(a_j)$, the resulting vector $z =$

$(z_1, ..., z_M)$ is called a "hidden" layer (as opposed to the observed input and output layers). Components of that hidden vector are further used to create K linear combinations, where K is the dimensionality of the output vector $y = (y_1, ..., y_K)$:

$$a_k = \sum_{j=1}^{M} w_{k,j}^{(2)} z_j + w_{k,0}^{(2)}, \forall k = 1, .., K \tag{3.2}$$

That is the the predicted output for the regression task $\widehat{y} = a$ and the objective function we need to minimize to fit the model is the squared prediction error over the whole data set $loss = \sum_{i=1}^{N} (y_i - \widehat{y}_i)^2$. The objective function is minimised using gradient descent. The neural network described is shown in figure 3.1.

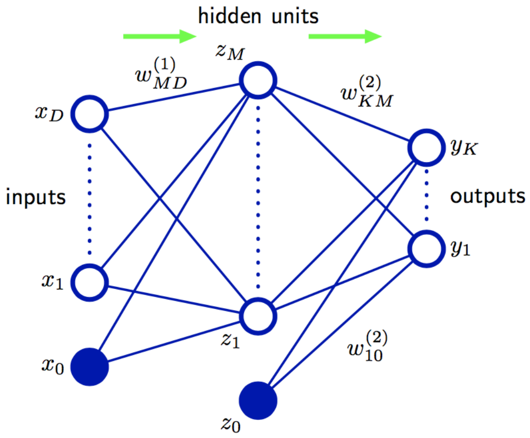


Figure 3.1: Neural network with 1 hidden layer. From Bishop [3]

It is possible to include multiple hidden layers in exactly the same way as the first one chaining them together. This increases the model complexity. Neural networks that have many of those hidden layer (typically more than 3) are called "deep neural networks".

3.2 Autoencoder

Autoencoder is an unsupervised learning algorithm, the aim of which is to find a latent representation of the data, typically with the purpose of dimensionality reduction. The model is based on two symmetrical neural networks: one mapping from the input $x = (x_1, ..., x_D)$ to a latent output $z = (z_1, ..., z_K)$ called encoder, and second network (decoder) mapping from the latent vector back the input.

The idea behind an autoencoder is to map input data back to itself but to do so through the bottleneck in the network, the lower dimensional representation in the middle layer. This capacity limit forces the model to capture in the learned representation the most important feature variations of the original data so that it is possible to map the latent code back to the original data with minimum loss. The whole combined network is trained jointly with the objective function being the reconstruction loss - the difference between the original input and the reconstructed version.

In the simplest case, the autoencoder can have a single hidden layer that is the latent representation. But most commonly encoder (and symmetrical decoders) with more layers are used. Autoencoders with a large (relatively speaking) number of hidden layers are commonly referred to as deep autoencoders.

An example of an autoencoder with 5 hidden layers (2 in the encoder, 2 in the decoder and a latent representation in the middle layer) is shown in figure 3.2.

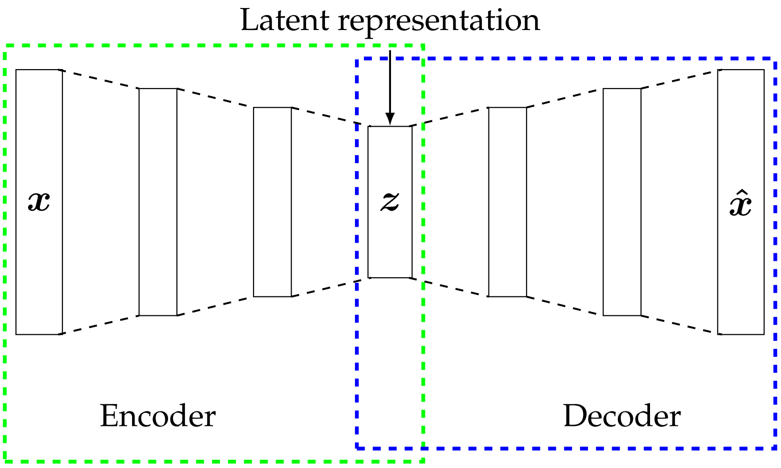


Figure 3.2: Autoencoder with 5 hidden layers.

3.3 Variational autoencoder

Variational autoencoder (VAE) is a probabilistic generative model. It is similar in spirit to standard autoencoder model described in section 3.2, but here input and latent representation are treated as random variables characterised by some probability distributions. Main advantages as compared to the non-probabilistic version include prior distribution on the latent space allowing to incorporate our beliefs of preferences about latent representation in a Bayesian way and the ability to generate new samples by drawing samples from the prior and applying the generative function (decoder).

The problem formulation for the VAE model is as follows. There is a data set $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ of N i.i.d. samples of an observed random variable $\mathbf{x}$ (continuous or discrete). The assumed generative process, involving a hidden continuous random variable $\mathbf{z}$, is presented in figure 3.3 (solid lines) in the form of a probabilistic graphical model (PGM). This PGM corresponds to the following factorization:

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \tag{3.3}$$

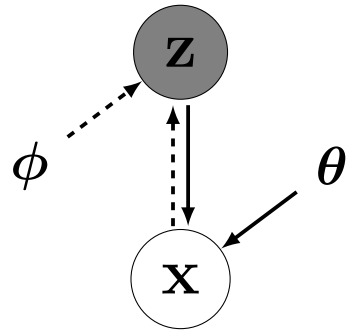


Figure 3.3: Graphical model. Solid lines show the generative process, dashed line show the inference process.

According to the PGM each data point is generated by first sampling a value $\mathbf{z}^{(i)}$ from some prior distribution $p(\mathbf{z})$ and than sampling $\mathbf{x}^{(i)}$ from some conditional distribution $p_\theta(\mathbf{x}|\mathbf{z})$. Both the prior and the likelihood are assumed to be from parametric families of distributions differentiable w.r.t. $\mathbf{z}$ and θ.

The prior distribution over the hidden variable $\mathbf{z}$ is usually taken to be an isotropic Gaussian $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$, but in principle can be different (with some constraint that will be discussed below). The likelihood is also a Gaussian:

$$p_\theta(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|f(\mathbf{z},\theta), \sigma^2\mathbf{I}) \tag{3.4}$$

where $f(\mathbf{z},\theta)$ is a neural network.

To be able to infer the latent variable values for the observed data we need the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x})$. The posterior is intractable, which is common for a complicated likelihood like the one we use involving a neural network.

The common solution to this problem is to approximate the true intractable posterior with a simpler distribution. This approach is the base of the Variational Bayesian Inference methods. That is where the "variational" in the VAE comes from.

The approximate posterior distribution is chosen to be a Gaussian with a diagonal structure:

$$q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}(\mathbf{x},\phi), \boldsymbol{\sigma}^{\mathbf{2}}(\mathbf{x},\phi)\mathbf{I}) \tag{3.5}$$

where the mean and the variance are computed using neural networks.

The model is trained by maximizing the evidence lower bound (ELBO), which is derived below. The marginal likelihood factorizes as follows:

$$\log p_\theta(\mathbf{X}) = \log \prod_{i=1}^{N} p_\theta(\mathbf{x}^{(i)}) = \sum_{i=1}^{N} \log p_\theta(\mathbf{x}^{(i)}) \tag{3.6}$$

Then the usual variational inference trick, multiplying and dividing by the approximate posterior, is used to get an expectation under the approximate distribution:

$$\begin{aligned} \log p_\theta(\mathbf{x}^{(i)}) &= \log \int p_\theta(\mathbf{x}^{(i)},\mathbf{z})d\mathbf{z} \\ &= \log \int \frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}d\mathbf{z} \\ &= \log E_{z\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \end{aligned} \tag{3.7}$$

Then the Jensen's inequality (equation 3.8) is applied to obtain a lower bound:

$$\psi(E[x]) \geq E[\psi(x)], \text{ if } \psi \text{ is a concave function} \tag{3.8}$$

$$\log E_{z \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\frac{p_\theta(\mathbf{x}^{(i)}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \geq E_{z \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \tag{3.9}$$

The expression under the expectation can be rewritten as follows:

$$\begin{aligned}
\mathcal{L}(\mathbf{x}^{(i)}) &= E_{z \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}, \mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= E_{z \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log \frac{p_\theta(\mathbf{x}^{(i)}|\mathbf{z})p(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \\
&= E_{z \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) + \log p(\mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x}^{(i)})\right] \\
&= E_{z \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] - \mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p(\mathbf{z})\Big)
\end{aligned} \tag{3.10}$$

where $\mathcal{KL}$ is the Kullback–Leibler divergence.

The ELBO for the whole data set is then:

$$\mathcal{L}(\mathbf{X}) = \sum_{i=1}^{N}\left[E_{z \sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z})\right] - \mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p(\mathbf{z})\Big)\right] \tag{3.11}$$

To maximize the objective function the gradients are needed, but it is not possible to take derivatives of a distribution with respect to its parameters. To tackle this problem Kingma and Welling [16] suggested a "reparameterization trick". The approximate posterior $\widetilde{\mathbf{z}} = q_\phi(\mathbf{z}|\mathbf{x})$ is reparameterized using a differentiable transformation of an auxiliary noise variable ϵ

$$\begin{aligned}
&\widetilde{\mathbf{z}} = g_\phi(\epsilon, \mathbf{x}) = \boldsymbol{\mu}(\mathbf{x}, \phi) + \boldsymbol{\sigma}(\mathbf{x}, \phi)\boldsymbol{\epsilon}, \\
&\text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})
\end{aligned} \tag{3.12}$$

Now the expectation can be taken using Monte Carlo estimates:

$$\widetilde{\mathcal{L}}(\mathbf{X}) = \sum_{i=1}^{N} \left[\frac{1}{L} \sum_{l=1}^{L} \left[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}) \right] - \mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p(\mathbf{z}) \Big) \right], \\ \text{where } \mathbf{z}^{(i,l)} = \boldsymbol{\mu}(\mathbf{x}^{(i)}, \phi) + \boldsymbol{\sigma}(\mathbf{x}^{(i)}, \phi)\boldsymbol{\epsilon}^{(i,l)} \tag{3.13}$$

The KL-divergence term can be computed analytically when both distributions are Gaussian:

$$\begin{aligned} \mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}) || p(\mathbf{z}) \Big) = -\frac{1}{2} \sum_{k=1}^{K} \Big(& 1 + \log \Big((\boldsymbol{\sigma}(\mathbf{x}^{(i)}, \phi)_{(k)})^2 \Big) \\ & - (\boldsymbol{\mu}(\mathbf{x}^{(i)}, \phi)_{(k)})^2 \\ & - (\boldsymbol{\sigma}(\mathbf{x}^{(i)}, \phi)_{(k)})^2 \Big) \end{aligned} \tag{3.14}$$

For the Gaussian likelihood reconstruction term has the following form:

$$\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)}) = -\frac{1}{2\sigma^2} \Big(\mathbf{x}^{(i)} - f(\mathbf{z}^{(i,l)}, \theta) \Big)^2 + constant \tag{3.15}$$

The objective is optimized using stochastic gradient descend. The estimation of the EBLO from a random data batch of size B is :

$$\mathcal{L}(\mathbf{X}) \approx \; \widetilde{\mathcal{L}}(\mathbf{X}^B) = \frac{N}{B} \sum_{i=1}^{B} \widetilde{\mathcal{L}}(\mathbf{x}^{(i)}) \tag{3.16}$$

The computational graph is shown in figure 3.4. To compute the value of the objective function, the approximate posterior distribution is computed, then the latent variable is sampled exploiting the reparameterization trick, and used to compute the mean of the output via the generative neural network. The loss function consists of two terms: the reconstruction loss and the KL-divergence of the approximate posterior from the prior. The procedure very much resembles the standard autoencoder (see section 3.2), with the reconstruction part (the approximate posterior) and the generative part playing the roles of encoder and decoder respectively. That is why the model is called Variational Autoencoder.

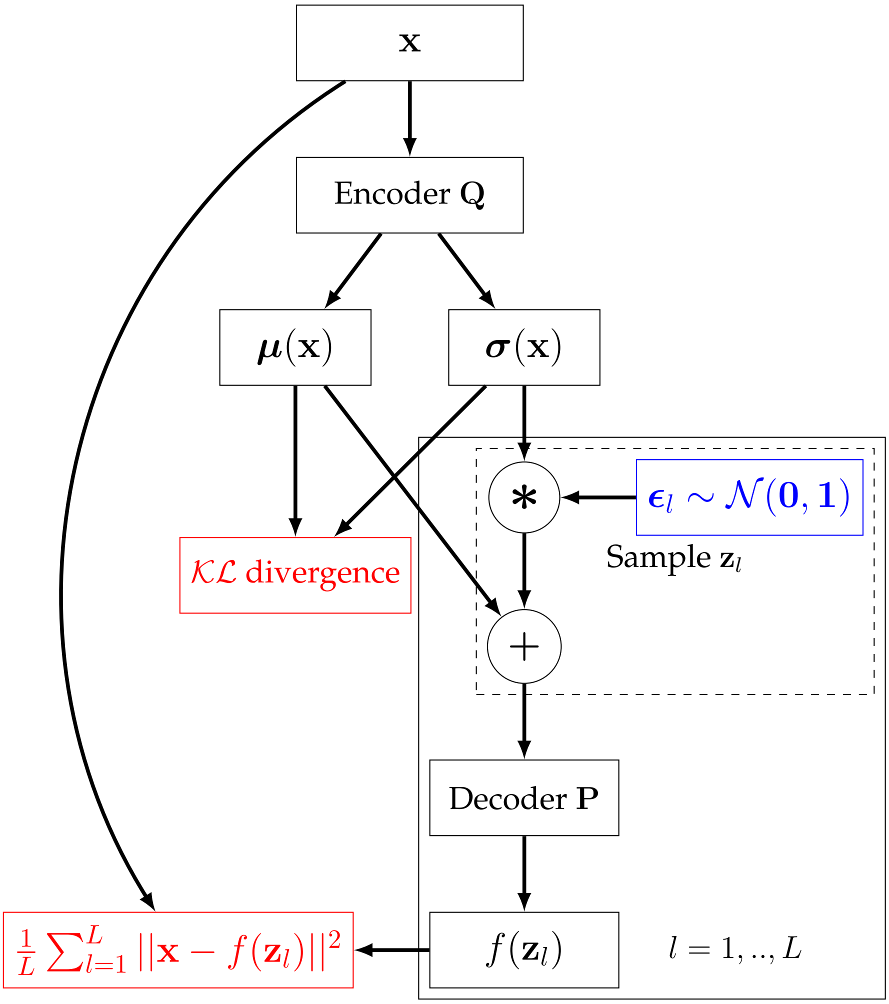

$\mathbf{x}$
Encoder **Q**
$\boldsymbol{\mu}(\mathbf{x})$
$\boldsymbol{\sigma}(\mathbf{x})$
$*$
$\boldsymbol{\epsilon}_l \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$
Sample $\mathbf{z}_l$
$\mathcal{KL}$ divergence
$+$
Decoder **P**
$\frac{1}{L}\sum_{l=1}^{L} ||\mathbf{x} - f(\mathbf{z}_l)||^2$
$f(\mathbf{z}_l)$
$l = 1, .., L$

Figure 3.4: Architecture of Variational Autoencoder

Chapter 4

Method

Variational autoencoder has been chosen as a primary framework mostly due to its elegant construction combining Bayesian approach (allowing priors) with neural networks for fast and easy inference. We will begin with the standard variational autoencoder and gradually build on that.

To begin with, in section 4.1 we put standard Variational Autoencoder (described in section 3.3) in the context of our task of facial expression representation learning (Model 1). We proceed by incorporating a neutral face as an additional input to the model to remove the appearance-related features from the latent space in section 4.2 (Model 2). In order to force the topology of the latent space to be consistent with human perception, the latter model is further extended by imposing topological constraints in the form of an additional component in the prior (section 4.3, Model 3).

4.1 Model 1: Standard VAE

We will start with the standard Variational Autoencoder (VAE, see section 3.3). In the context of our problem the input data set $\mathbf{X} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}$ consists of data points each of which is a set of 83 3-dimensional facial landmarks. So, the dimensionality of the observed variable $\mathbf{x}$ is $83 * 3 = 249$. The dimensionality K of the latent variable $\mathbf{z}$ is not fixed at this point and will be explored during the experiments.

We use the isotropic Gaussian prior on the latent space. The likelihood and the approximate posterior distribution are also Gaussian with the parameters computed via neural networks. The model is trained using Stochastic Gradient Variational Bayes (SGVB) on mini-batches of data with Adam optimizer [17]. The loss function approximation (based on a mini-batch of size B) we need to minimize in order to fit the model has the following form:

$$
\begin{aligned}
&Loss(\mathbf{X}) \approx Loss(\mathbf{X^B}) = -\widetilde{\mathcal{L}}(\mathbf{X}^B) = -\frac{N}{B}\sum_{i=1}^{B}\widetilde{\mathcal{L}}(\mathbf{x}^{(i)}) \\
&= \frac{N}{B}\sum_{i=1}^{B}\Bigg[-\frac{1}{L}\sum_{l=1}^{L}\Big[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i,l)})\Big] + \mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)})||p(\mathbf{z})\Big)\Bigg] \\
&= \frac{N}{B}\sum_{i=1}^{B}\Bigg[\frac{1}{2\sigma^2 L}\sum_{l=1}^{L}\Big[\Big(\mathbf{x}^{(i)} - f(\mathbf{z}^{(i,l)},\theta)\Big)^2\Big] \\
&- \frac{1}{2}\sum_{k=1}^{K}\Big(1 + \log\Big((\boldsymbol{\sigma}(\mathbf{x}^{(i)},\phi)_{(k)})^2\Big) - (\boldsymbol{\mu}(\mathbf{x}^{(i)},\phi)_{(k)})^2 - (\boldsymbol{\sigma}(\mathbf{x}^{(i)},\phi)_{(k)})^2\Big)\Bigg], \\
&\text{where } \mathbf{z}^{(i,l)} = \boldsymbol{\mu}(\mathbf{x}^{(i)},\phi) + \boldsymbol{\sigma}(\mathbf{x}^{(i)},\phi)\boldsymbol{\epsilon}^{(i,l)}
\end{aligned}
\tag{4.1}
$$

The computational graph is the standard VAE (figure 3.4 in section 3.3).

4.2 Model 2: VAE with neutral facial expressions

The main problem of the standard autoencoder model for this application is that the whole face is generated from the latent space, i.e. latent representation contain information not only about facial expression (which is of interest to us) but also about individual features (shape of the person's nose, eyes, etc.).

To eliminate the effect of individual facial characteristics one possible solution could be splitting latent space into individual features and expression-related features, but there is not enough variability in appearances in the data set. We propose a different approach - using a neutral facial expression of a person as an additional input to the model and therefore modelling only the difference or rather transformation of the neutral face into any other expression of the same person.

The corresponding probabilistic graphical model is shown in figure 4.1, where solid lines show the generative process and the dashed lines show the inference process. There $\mathbf{z}$ as previously denotes the hidden variable of dimensionality K, $(\mathbf{X}, \mathbf{Y}) = \{(\mathbf{x}^{(i)}, \mathbf{y}^{(i)})\}_{i=1}^{N}$ is a data set of facial expressions, where $\mathbf{y}^{(i)}$ is a neutral face of a person and $\mathbf{x}^{(i)}$ is any facial expression of the same person. Both $\mathbf{x}$ and $\mathbf{y}$ have the dimensionality $D = 249$.

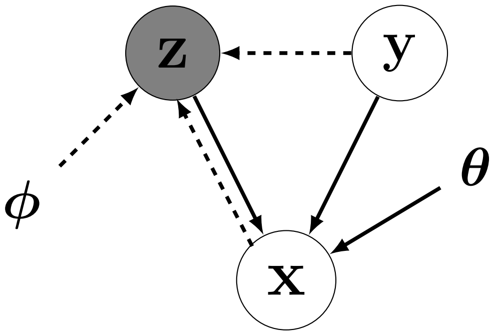


Figure 4.1: The graphical representation of Model 2. Solid lines show the generative process, dashed line show the inference process.

The model factorizes as follows :

$$p_\theta(\mathbf{x}, \mathbf{z}|\mathbf{y}) = p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z})p(\mathbf{z}) \tag{4.2}$$

The prior over the latent variable is the same isotropic Gaussian as before $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, \mathbf{I})$, but the likelihood now also depends on the neutral face:

$$p_\theta(\mathbf{x}|\mathbf{y}, \mathbf{z}) = \mathcal{N}(\mathbf{x}|f(\mathbf{y}, \mathbf{z}, \theta), \sigma^2\mathbf{I}) \tag{4.3}$$

where $f(\mathbf{y}, \mathbf{z}, \theta)$ is a neural network.

From the PGM (figure 4.1) we can see that $\mathbf{y}$ and $\mathbf{z}$ are connected through a "V-structure" and therefore not independent given $\mathbf{z}$. The posterior distribution over the latent variable is $p_\theta(\mathbf{z}|\mathbf{x}, \mathbf{y})$, which is intractable. As previously, we approximate it with a different distribution, which now also depends on the neutral face $\mathbf{y}$:

$$q_\phi(\mathbf{z}|\mathbf{x},\mathbf{y}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}(\mathbf{x},\mathbf{y},\phi), \boldsymbol{\sigma}^{\mathbf{2}}(\mathbf{x},\mathbf{y},\phi)\mathbf{I}) \tag{4.4}$$

where the mean and the variance are computed using neural networks.

The computational graph for this model is shown in figure 4.2. As we can see, the only difference from the Model 1 (standard VAE, figure 3.3) is the corresponding neutral face, that is an additional input to both the reconstruction (encoding) network and the generative network.

For this model evidence lower bound (EBLO) on the whole data set has the following form:

$$\begin{aligned}\mathcal{L}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{N} \Bigg[& E_{z\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)},\mathbf{y}^{(i)})}\Big[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{y}^{(i)},\mathbf{z})\Big] \\ & - \mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)},\mathbf{y}^{(i)})||p(\mathbf{z})\Big)\Bigg]\end{aligned} \tag{4.5}$$

The loss function approximation based on a mini-batch of size B is:

$$\begin{aligned}& Loss(\mathbf{X},\mathbf{Y}) \approx Loss(\mathbf{X}^B,\mathbf{Y}^B) = -\widetilde{\mathcal{L}}(\mathbf{X}^B,\mathbf{Y}^B) = -\frac{N}{B}\sum_{i=1}^{B}\widetilde{\mathcal{L}}(\mathbf{x}^{(i)},\mathbf{y}^{(i)}) \\ & = \frac{N}{B}\sum_{i=1}^{B}\Bigg[-\frac{1}{L}\sum_{l=1}^{L}\Big[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{y}^{(i)},\mathbf{z}^{(i,l)})\Big] + \mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)},\mathbf{y}^{(i)})||p(\mathbf{z})\Big)\Bigg] \\ & = \frac{N}{B}\sum_{i=1}^{B}\Bigg[\frac{1}{2\sigma^2 L}\sum_{l=1}^{L}\Big[\Big(\mathbf{x}^{(i)} - f(\mathbf{z}^{(i,l)},\mathbf{y}^{(i)},\theta)\Big)^2\Big] \\ & - \frac{1}{2}\sum_{k=1}^{K}\Bigg(1 + \log\Big((\boldsymbol{\sigma}(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\phi)_{(k)})^2\Big) - (\boldsymbol{\mu}(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\phi)_{(k)})^2 \\ & \qquad - (\boldsymbol{\sigma}(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\phi)_{(k)})^2\Bigg)\Bigg], \\ & \text{where } \mathbf{z}^{(i,l)} = \boldsymbol{\mu}(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\phi) + \boldsymbol{\sigma}(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\phi)\boldsymbol{\epsilon}^{(i,l)}\end{aligned} \tag{4.6}$$

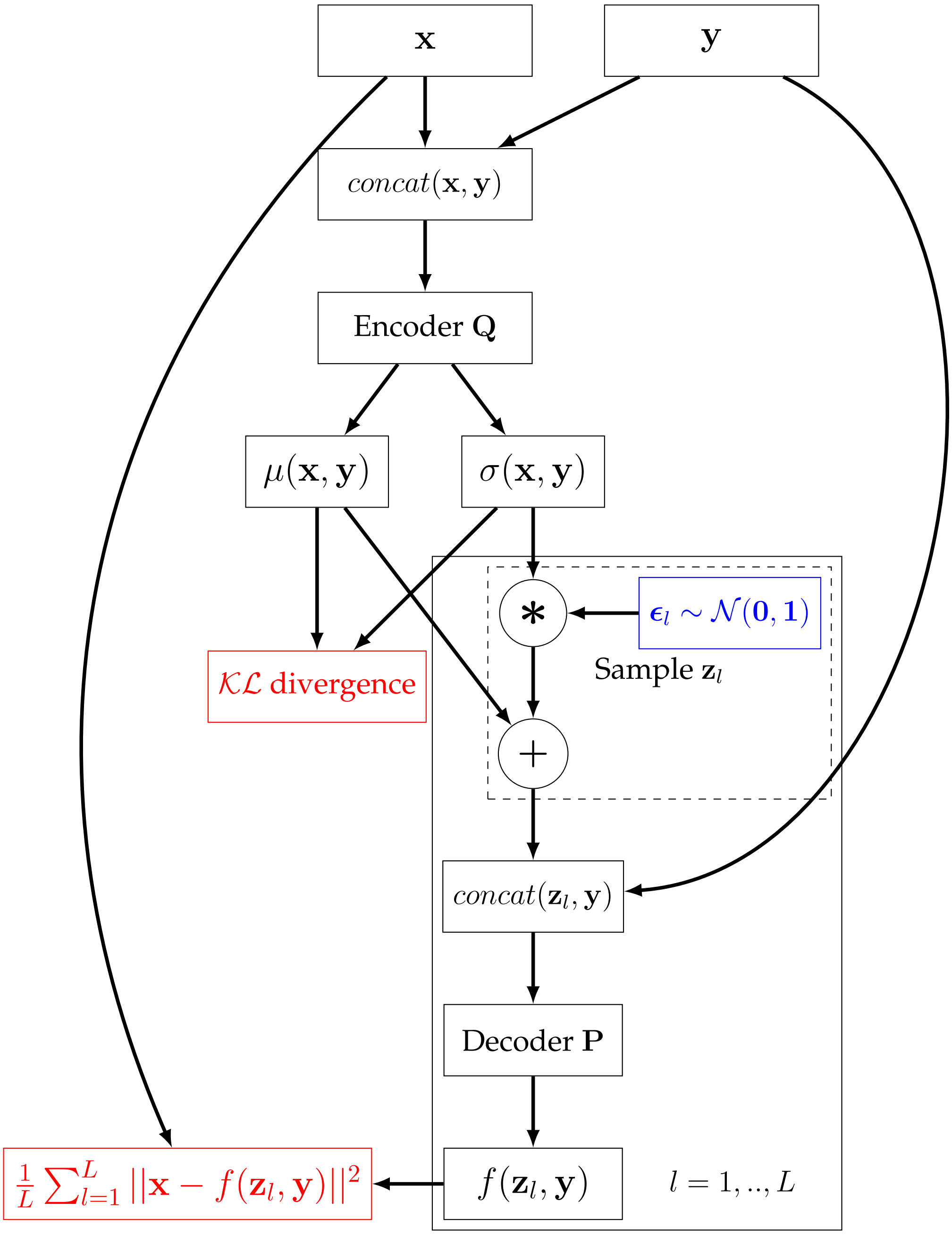

$\mathbf{x}$
$\mathbf{y}$
$concat(\mathbf{x},\mathbf{y})$
Encoder $\mathbf{Q}$
$\mu(\mathbf{x},\mathbf{y})$
$\sigma(\mathbf{x},\mathbf{y})$
$\mathcal{KL}$ divergence
$*$
$\boldsymbol{\epsilon}_l \sim \mathcal{N}(\mathbf{0},\mathbf{1})$
Sample $\mathbf{z}_l$
$+$
$concat(\mathbf{z}_l,\mathbf{y})$
Decoder $\mathbf{P}$
$f(\mathbf{z}_l,\mathbf{y})$
$l = 1, .., L$
$\frac{1}{L}\sum_{l=1}^{L}||\mathbf{x} - f(\mathbf{z}_l,\mathbf{y})||^2$

Figure 4.2: Computational graph for Model 2

4.3 Model 3: VAE with neutral facial expressions and topology

Our hypothesis is that imposing human-like topological constraints on the latent space will result in learning a better representation. We combine ideas of Urtasun et al. [26] and Hoffer and Ailon [14] on how to impose a topology on latent representation, and propose a new model based on the previous one with modified prior on the latent space.

Topological constraints on the latent space are represented as a set of T triplets, where each triplet consists of a reference face and two other faces with one of those faces marked as being more similar to the reference one based on a human opinion. The resulting triplet data set is

$$\mathbf{S} = \left\{ (\mathbf{s}^{(t,ref)}, \mathbf{s}^{(t,+)}, \mathbf{s}^{(t,-)}) : d(\mathbf{h}^{(s_t^{ref})}, \mathbf{h}^{(s_t^{+})}) \leq d(\mathbf{h}^{(s_t^{ref})}, \mathbf{h}^{(s_t^{-})}) \right\}_{t=1}^{T} \tag{4.7}$$

where each of $\mathrm{s}^{(t,ref)}, \mathrm{s}^{(t,+)}, \mathrm{s}^{(t,-)}$ corresponds to some index $i \in \{1, ..N\}$ in the original data set of facial expression, d is the Euclidean distance and $\mathbf{h}^{(i)}$ is some (human-like) representation of the facial expression $\mathbf{x}^{(i)}$.

To fulfil these topological constraints over triplets on the latent representation $\mathbf{z}$ we want to minimize the following expression:

$$\Phi(\mathbf{Z}, \mathbf{S}) = \sum_{i=1}^{T} max\big(0; d(\mathbf{z}^{(s_t^{ref})}, \mathbf{z}^{(s_t^{+})}) - d(\mathbf{z}^{(s_t^{ref})}, \mathbf{z}^{(s_t^{-})})\big) \tag{4.8}$$

Instead of using $f(x) = max(0; x)$ to penalize incorrect distances we will use its smooth approximation $f(x) = ln(1+e^x)$ called "softplus" to force a little margin on the distance difference. For additional flexibility each triplet can have a weigh w_t (e.g. corresponding to a reliability level for each triplet if the triplets are collected from people).

$$\Phi(\mathbf{Z}, \mathbf{S}) = \sum_{i=1}^{T} w_t \ln\big(1 + \exp(d(\mathbf{z}^{(s_t^{ref})}, \mathbf{z}^{(s_t^{+})}) - d(\mathbf{z}^{(s_t^{ref})}, \mathbf{z}^{(s_t^{-})}))\big) \tag{4.9}$$

This can be interpreted as a prior [26] that forces to fulfil as many constraints as possible:

$$p_T(\mathbf{Z}|\mathbf{S}) \propto e^{-\frac{1}{\gamma}\Phi(\mathbf{Z},\mathbf{S})} \tag{4.10}$$

where γ is a "topological variance" and the smaller the value, the larger the penalty for incorrect topology.

This topological prior can be factorised over triplets:

$$\begin{aligned} p_T(\mathbf{Z}|\mathbf{S}) &= \prod_{t=1}^{T} p_T(\mathbf{Z}|\mathbf{s}^{(t)}) = \prod_{t=1}^{T} p_T\Big(\mathbf{z}^{(s_t^{ref})}, \mathbf{z}^{(s_t^{+})}, \mathbf{z}^{(s_t^{+})}|\mathbf{s}^{(t)}\Big) \\ &\propto \prod_{t=1}^{T} \exp\Big(-\frac{1}{\gamma}\Phi(\mathbf{z}^{(s_t^{ref})}, \mathbf{z}^{(s_t^{+})}, \mathbf{z}^{(s_t^{+})})\Big) \end{aligned} \tag{4.11}$$

The topological prior on latent variable z can be added to the standard Gaussian prior we used in the previous models:

$$\begin{aligned} p(\mathbf{Z}) &= p_T(\mathbf{Z}|\mathbf{S})p_{\mathcal{N}}(\mathbf{Z}) \\ &= \prod_{t=1}^{T} p_T\Big(\mathbf{z}^{(s_t^{ref})}, \mathbf{z}^{(s_t^{+})}, \mathbf{z}^{(s_t^{+})}|\mathbf{s}^{(t)}\Big) \prod_{i=1}^{N} \mathcal{N}(\mathbf{z}^{(i)}|\mathbf{0}, \mathbf{I}) \end{aligned} \tag{4.12}$$

Given that now the prior is not factorizable over the data points, we derive evidence lower bound on the whole data set.

$$\begin{aligned} \log p_\theta(\mathbf{X}|\mathbf{Y}) &= \log \int p_\theta(\mathbf{X}, \mathbf{Z}|\mathbf{Y}) d\mathbf{Z} \\ &= \log \int q_\phi(\mathbf{Z}|\mathbf{X}, \mathbf{Y}) \frac{p_\theta(\mathbf{X}, \mathbf{Z}|\mathbf{Y})}{q_\phi(\mathbf{Z}|\mathbf{X}, \mathbf{Y})} d\mathbf{Z} \\ &\geq E_{\mathbf{Z}\sim q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\left[\log \frac{p_\theta(\mathbf{X}, \mathbf{Z}|\mathbf{Y})}{q_\phi(\mathbf{Z}|\mathbf{X}, \mathbf{Y})}\right] = \mathcal{L}(\mathbf{X}, \mathbf{Y}) \end{aligned} \tag{4.13}$$

where the last line (ELBO) is the result of applying Jensen's inequality (equation 3.8).

$$
\begin{aligned}
\mathcal{L}(\mathbf{X},\mathbf{Y}) &= E_{\mathbf{Z}\sim q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\Bigg[\log\frac{p_\theta(\mathbf{X},\mathbf{Z}|\mathbf{Y})}{q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\Bigg] \\
&= E_{\mathbf{Z}\sim q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\Bigg[\log\frac{p_\theta(\mathbf{X}|\mathbf{Z},\mathbf{Y})p_T(\mathbf{Z}|\mathbf{S})p_{\mathcal{N}}(\mathbf{Z})}{q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\Bigg] \\
&= E_{\mathbf{Z}\sim q_\phi(\mathbf{Z}|\mathbf{X},\mathbf{Y})}\Bigg[\log\prod_{i=1}^{N}p_\theta(\mathbf{x}^{(i)}|\mathbf{z}^{(i)},\mathbf{y}^{(i)}) \\
&\qquad + \log\prod_{t=1}^{T}p_T\Big(\mathbf{z}^{(s_t^{ref})},\mathbf{z}^{(s_t^{+})},\mathbf{z}^{(s_t^{+})}\Big) \\
&\qquad + \log\prod_{i=1}^{N}p_{\mathcal{N}}(\mathbf{z}^{(i)}) - \log\prod_{i=1}^{N}q_\phi(\mathbf{z}^{(i)}|\mathbf{x}^{(i)},\mathbf{y}^{(i)})\Bigg] \\
&= \sum_{i=1}^{N}E_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)},\mathbf{y}^{(i)})}\Big[\log p_\theta(\mathbf{x}^{(i)}|\mathbf{z},\mathbf{y}^{(i)})\Big] \\
&- \sum_{i=1}^{N}\mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)},\mathbf{y}^{(i)})||p_{\mathcal{N}}(\mathbf{z})\Big) \\
&+ \sum_{t=1}^{T}E_{\left\{\begin{array}{ll}\mathbf{z_r} & \sim q_\phi(\mathbf{z}_r|\mathbf{x}^{(s_t^{ref})},\mathbf{y}^{(s_t^{ref})}) \\ \mathbf{z}_+ & \sim q_\phi(\mathbf{z}_+|\mathbf{x}^{(s_t^{+})},\mathbf{y}^{(s_t^{+})}) \\ \mathbf{z}_- & \sim q_\phi(\mathbf{z}_-|\mathbf{x}^{(s_t^{-})},\mathbf{y}^{(s_t^{-})})\end{array}\right\}}\Big[\log p_T\Big(\mathbf{z}_r,\mathbf{z}_+,\mathbf{z}_-\Big)\Big]
\end{aligned}
\tag{4.14}
$$

We further write the exact density functions and use the reparameterization trick to derive a differentiable lower bound:

$$
\begin{aligned}
\mathcal{L}(\mathbf{X},\mathbf{Y}) = \sum_{i=1}^{N}\Bigg[&-\frac{1}{2\sigma^2L}\sum_{l=1}^{L}\Big(\mathbf{x}^{(i)}-f(\mathbf{z}^{(i,l)},\mathbf{y}^{(i)})\Big)^2 \\
&-\mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)},\mathbf{y}^{(i)})||p_{\mathcal{N}}(\mathbf{z})\Big)\Bigg] \\
-\frac{1}{\gamma M}\sum_{t=1}^{T}w_t\sum_{m=1}^{M}&\ln\Big(1+\exp\big(d(\mathbf{z}^{(s_t^{ref},l)},\mathbf{z}^{(s_t^{+},l)}) \\
&\qquad - d(\mathbf{z}^{(s_t^{ref},l)},\mathbf{z}^{(s_t^{-},l)})\big)\Big) \\
+\, \mathit{constant}&
\end{aligned}
\tag{4.15}
$$

where $\mathbf{z}^{(i,l)} = \boldsymbol{\mu}(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\phi)+\boldsymbol{\sigma}(\mathbf{x}^{(i)},\mathbf{y}^{(i)},\phi)\cdot\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon}\sim\mathcal{N}(\mathbf{0},\mathbf{I})$,

$f(\mathbf{y}, \mathbf{z})$ - the generative neural network.

To maximize the evidence lower bound with batch-wise stochastic gradient descend the objective is reformulated. We use separate data batches and batches of triplets. The approximation of the loss function based on a data batch of size B and a batch of triplets of size V can be computed in the following way:

$$\begin{aligned}
Loss(\mathbf{X}, \mathbf{Y}, \mathbf{S}) \approx Loss(\mathbf{X}^B, \mathbf{Y}^B, \mathbf{S}^V) &= -\widetilde{\mathcal{L}}(\mathbf{X}^B, \mathbf{Y}^B, \mathbf{S}^V) \\
&= \frac{N}{B} \sum_{i=1}^{B} \Bigg[\frac{1}{2\sigma^2 L} \sum_{l=1}^{L} \Big(\mathbf{x}^{(i)} - f(\mathbf{z}^{(i,l)}, \mathbf{y}^{(i)}) \Big)^2 \\
&\qquad + \mathcal{KL}\Big(q_\phi(\mathbf{z}|\mathbf{x}^{(i)}, \mathbf{y}^{(i)}) || p_\mathcal{N}(\mathbf{z}) \Big) \Bigg] \\
&+ \frac{T}{V} \frac{1}{\gamma M} \sum_{t=1}^{V} w_t \sum_{m=1}^{M} \ln \Big(1 + \exp \big(d(\mathbf{z}^{(s_t^{ref}, m)}, \mathbf{z}^{(s_t^{+}, m)}) \\
&\qquad - d(\mathbf{z}^{(s_t^{ref}, m)}, \mathbf{z}^{(s_t^{-}, m)}) \big) \Big)
\end{aligned} \tag{4.16}$$

Note, that in fact the batch-wise objective also depends on the data corresponding to the triplet batch.

The architecture of the computational graph for this model is a combination of the one used in the previous model (figure 4.2 in section 4.2) and the part shown in figure 4.3, which is responsible for the triplet term in the loss function and was inspired by the "triplet network" of Hoffer and Ailon [14]. All the parts of the loss function are shown in red in the pictures.

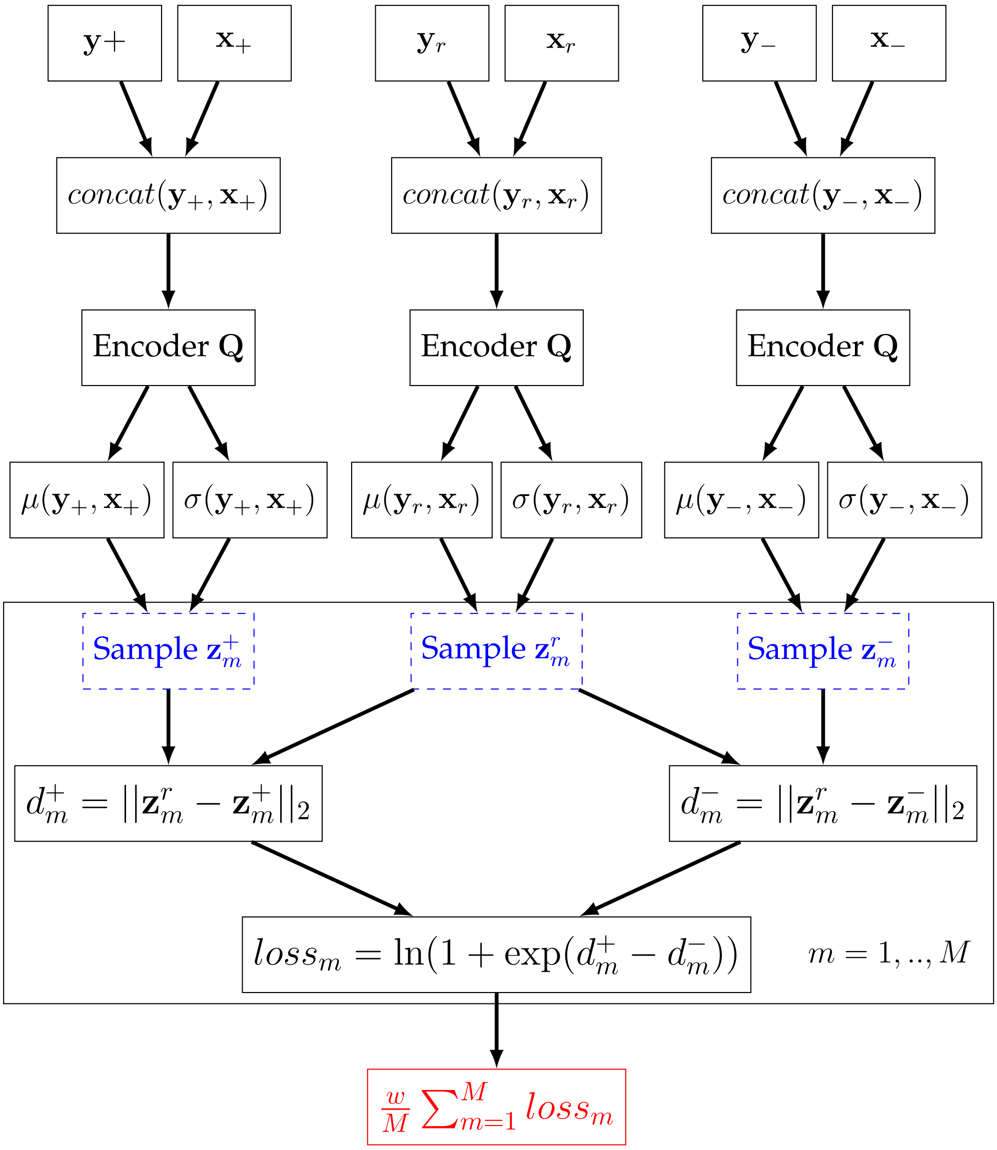

$\mathbf{y}+$
$\mathbf{x}_+$
$\mathbf{y}_r$
$\mathbf{x}_r$
$\mathbf{y}_-$
$\mathbf{x}_-$
$concat(\mathbf{y}_+, \mathbf{x}_+)$
$concat(\mathbf{y}_r, \mathbf{x}_r)$
$concat(\mathbf{y}_-, \mathbf{x}_-)$
Encoder Q
Encoder Q
Encoder Q
$\mu(\mathbf{y}_+, \mathbf{x}_+)$
$\sigma(\mathbf{y}_+, \mathbf{x}_+)$
$\mu(\mathbf{y}_r, \mathbf{x}_r)$
$\sigma(\mathbf{y}_r, \mathbf{x}_r)$
$\mu(\mathbf{y}_-, \mathbf{x}_-)$
$\sigma(\mathbf{y}_-, \mathbf{x}_-)$
Sample $\mathbf{z}_m^+$
Sample $\mathbf{z}_m^r$
Sample $\mathbf{z}_m^-$
$d_m^+ = ||\mathbf{z}_m^r - \mathbf{z}_m^+||_2$
$d_m^- = ||\mathbf{z}_m^r - \mathbf{z}_m^-||_2$
$loss_m = \ln(1 + \exp(d_m^+ - d_m^-))$
$m = 1, .., M$
$\frac{w}{M} \sum_{m=1}^{M} loss_m$

Figure 4.3: Architecture of the "triplet" part of the model 3

Chapter 5

Data

The models described in chapter 4 require facial expression data as input. Models 2 and 3 also require neutral expressions as additional input. Model 3 imposes the human-like similarity metric on the latent space needs triplet data to constrain the latent space. Two data sets will be used in this work, one containing posed static facial expressions and the second containing dynamic spontaneous expressions. Triplet data will be collected using crowd-sourcing service (Amazon Mechanical Turk).

5.1 Data sets

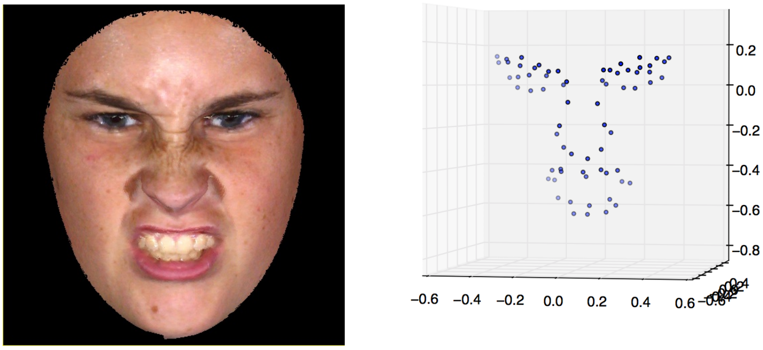
5.1.1 Static posed data set with stereotypical facial expressions

This is a dataset with 3d facial landmarks (83 points for each face) consisting of 100 individuals, each posed with stereotypical facial expressions: "neutral", "angry", "disgust", "sad", "happy", "surprised", "fear" [29]. All expressions except neutral have 4 degrees, so in total each person has 25 data points and the total size of the dataset is 2 500 facial expressions with dimensionality $83 * 3 = 249$. This data set is quite small and don't have enough variability, but have labels which can be useful for evaluation. Examples from this data set are shown in figure

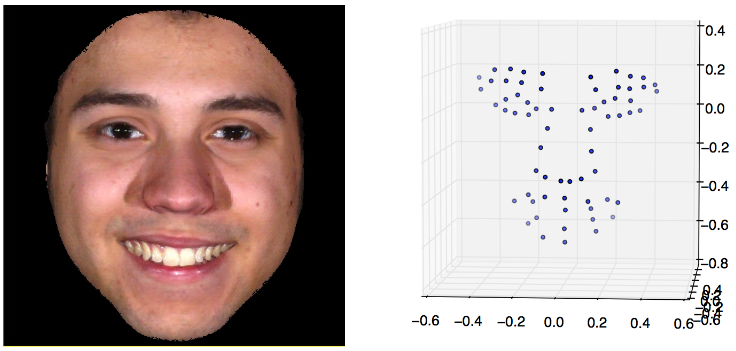
5.1. The data set is split into training, validation and test as specified in table 5.1.

	Train	Validation	Test	All
Number of people	80	10	10	100
Data set size	2000	250	250	2500

Table 5.1: Posed data set statistics



(a) Image (example 1) (b) 3D landmarks (example 1)



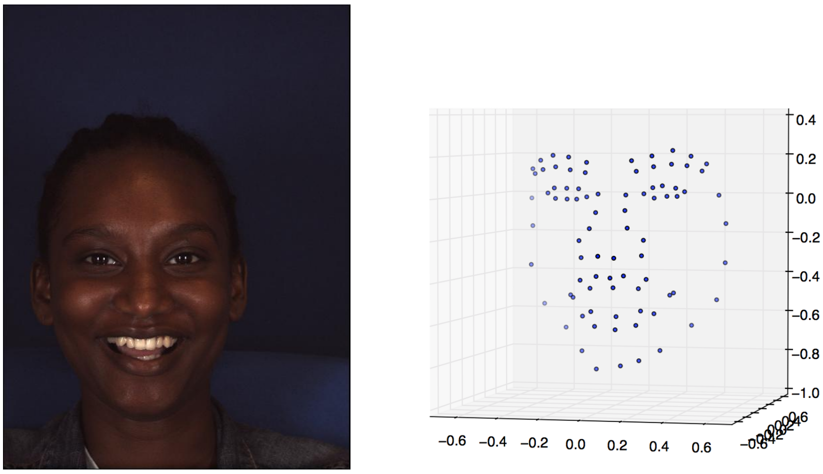
(c) Image (example 2) (d) 3D landmarks (example 2)

Figure 5.1: Examples of data from the static dataset (images and corresponding 3D facial landmarks

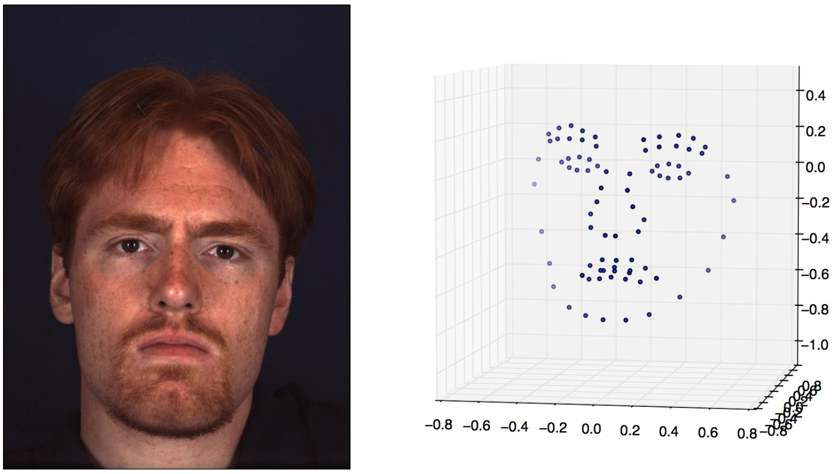
5.1.2 Dynamic spontaneous data set

For this data set 41 individuals were asked to participate in 8 tasks, each task has an intended emotion (e.g. "sing a song" for "embarrassment") [30]. Only the most emotional part if saved for each sequence and the rest is discarded. Each time frame has 83 3D facial landmarks.

There are totally 367 492 facial expressions (time frames). This data set is much larger than the first one and has more variability and smoothness. Examples of images and the corresponding 3D facial landmark can be seen in figure 5.2.



(a) Image (example 1) (b) 3D landmarks (example 1)



(c) Image (example 2) (d) 3D landmarks (example 2)

Figure 5.2: Examples of data from spontaneous dataset (images and corresponding 3D facial landmarks

We split the data set into 3 subsets person-wise: train, validation, test (table 5.2).

In this dataset for each sequence, 20 seconds were manually annotated with action units (AU) by specialists. Not all possible AUs are labeled, and some AUs are extremely rare, so we decided to only use action

	Train	Validation	Test	All
Number of people	25	8	8	41
Data set size	223 883	72 173	71 436	367 492
Labeled size	88 570	29 450	28 632	146 652

Table 5.2: Spontaneous data set statistics

units that are present in reasonable amounts in each train, validation, test data set, 12 in total (AU 1, 2, 4, 6, 7, 10, 12, 14, 15, 17, 23, 24). For the detailed class balance of action units see table 5.3.

AU	FACS code	Train	Validation	Test	All
1	1	0.216	0.227	0.18	0.221
2	2	0.179	0.154	0.161	0.17
3	4	0.217	0.173	0.185	0.201
4	6	0.479	0.396	0.472	0.448
5	7	0.54	0.473	0.657	0.516
6	10	0.633	0.567	0.502	0.61
7	12	0.582	0.515	0.548	0.558
8	14	0.503	0.395	0.425	0.462
9	15	0.16	0.135	0.233	0.151
10	17	0.353	0.256	0.404	0.3162
11	23	0.165	0.092	0.244	0.137
12	24	0.169	0.08	0.17	0.135

Table 5.3: AU class balance (share of positive occurrences) in the spontaneous data set

5.2 Triplets

Model 3 uses our hypothesis that incorporating knowledge about human perception of facial expression will help to learn a better latent representation. In order to formalise this knowledge we collect triplet data, where people choose which facial expressions are more similar.

5.2.1 Artificial triplets

As a preparation for the collection of real triplets and a proof of concept, we conduct experiments on the "fake" (artificial) triplets. Only the first data set is used.

These triplets are generated from the true labels of the posed data set using the following rules:

- Expressions from the same class are closer than from different classes.
- Within the same class: the closer the degree of expression the smaller the distance.
- A neutral expression is closer than an expression from a different class.

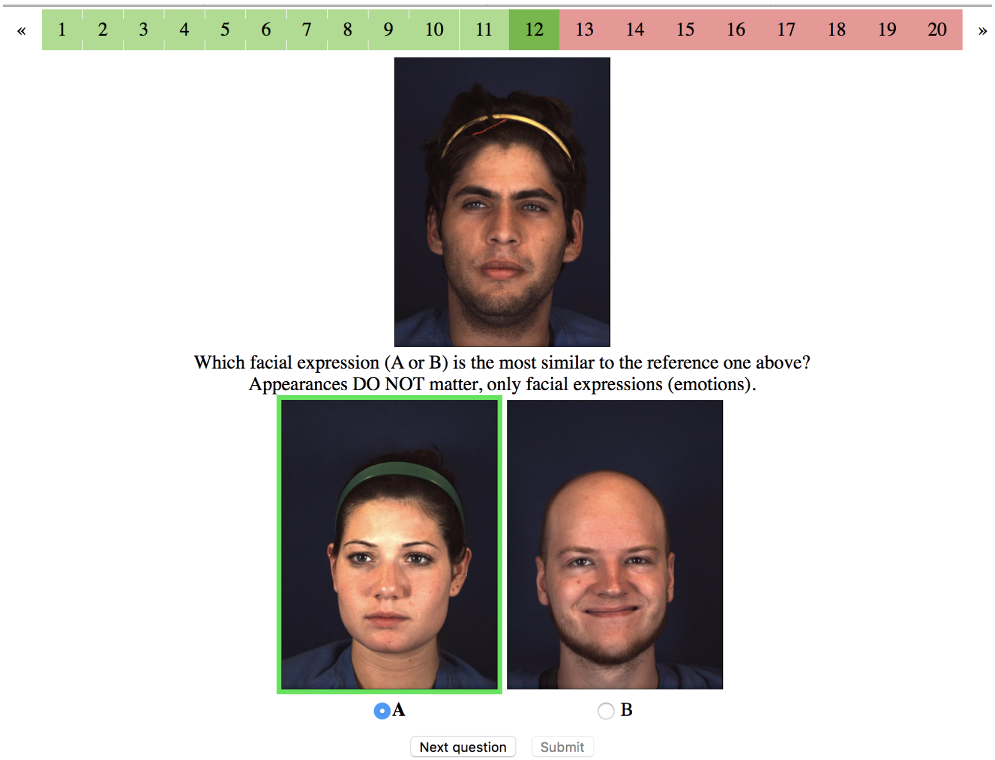
For the train data set (2000 data points), 8000 triplets were generated. For the validation and test sets (250 data points each) - 1000 triplets per set. Every data point is present in at least one triplet.

5.2.2 Design choices for data set of triplets

Triplets are collected using Amazon Mechanical Turk. Participants are asked to chose which of the 2 facial expressions A or B looks more similar to the reference one (5.3). They are provided with pictures rather than a set of 3D points. Those triplets are collected only on the second data set (spontaneous).

During preliminary tests (on ourselves) it was noticed, that in cases where the reference image and one of the possible answers belong to the same person there is a tendency to chose similar appearance rather than expression. To eliminate this bias, it was decided to use only either triplets where all images are from the same person or all images are from different persons.

The number of triplets to collect depends on several factors such as the number of facial expressions, desired connectivity in the resulting graph and financial restrictions.



Which facial expression (A or B) is the most similar to the reference one above?
Appearances DO NOT matter, only facial expressions (emotions).

Figure 5.3: Example of a question for the triplet collection with Amazon Mechanical Turk

Selection of images

The dataset we use for triplets contains sequences of video frames with spontaneous reactions. Size of the data set is quite large and we would need too many triplets to cover it. At the same time, it is often the case that a person has the same expression for several seconds (especially neutral expression) and using all those expressions for triplet collection is unnecessary. Another thing to consider is that some of the images have a blinking person. Blinking does not matter much in videos, but in a static image, it makes understanding facial expression harder.

Therefore we have a task of selecting a subset of the data that have a reasonable size and ideally cover the true latent space evenly. We exploit the fact that the most expressive part of each sequence was annotated with AUs. The number of images is further minimized by removing blinking based on the distances between lower and upper eyelids in the corresponding 3D point cloud of facial landmarks.

For the final subset of images for each person, we select images with unique vectors of action units occurrences. In total 5602 images were selected.

Selected images were brightened (the original images are a bit too dark), resized and uploaded to Amazon S3.

Selection of triplet connections

For each subset of data (train, validation and test) separate sets of triplets were generated in such a way that each image is in 3 triplets with the same person and in 3 multi-person triplets. This produces quite densely connected graph. The selection was performed for each data set separately in a greedy fashion with a restart if some constraints were hit before all the triplets were selected. The total number of triplets is $5602 * 6/3 = 11204$.

Redundancy

Given that data we are collecting are not the ground truth but rather opinions, and often there is no obviously correct answer (e.g. in cases where all 3 facial expressions are completely different) we decided to collect 5 answers from different people in order to get statistics for the trustworthiness of each triplet. An odd number (5) was chosen so that we can always get a majority vote and therefore use all the data. Possible results for the majority answer are $5-0, 4-1, 3-2$, and can be equivalently summarized as the difference in the number of votes for each answer (5, 3 and 1 respectively).

The reliability statistic can be used in two conceptual ways:

- use majority vote as a ground truth in triplets with the desired reliability level (e.g only 3 and 5)
- use the difference as a weighting coefficient

The second option will be used for training because it utilizes all the collected data. The first one, however, can be useful for the evaluating the results in terms of the number of "satisfied" triplets in the learned latent space.

5.2.3 Data collection with Amazon Mechanical Turk

Triplets were randomly split into sets of 19. In each set, we also added one of the 17 test triplets manually chosen so that one of the options is obviously correct. The total number of human intelligence tasks (HITs, Mturk terminology) is 589. The example of HIT is shown in figure 5.3. Each HIT is assigned to 5 different workers.

As workers tend to do multiple HIT of the same type, it is possible to collect accuracy statistics on the test questions. We only use data from workers who have at least 75% accuracy and post new HITs until the desired number of sufficient quality answers is achieved.

5.2.4 Statistics of the collected data

Total number of collected triplets is 11191, 5 answers for each.

Data subset	**Difference**			**Total**
	5	3	1	
Train:	**2 426**	**2 364**	**2 206**	**6 996**
one person	1 299	1 181	1 016	3 496
different persons	1 127	1 183	1 190	3 500
Validation:	**652**	**686**	**643**	**1 981**
one person	364	328	298	990
different persons	288	358	345	991
Test:	**728**	**767**	**719**	**2 214**
one person	418	367	322	1 107
different persons	310	400	397	1 107
All subsets:	**3 806**	**3 817**	**3 568**	**11 191**
one person	2 081	1 876	1 636	5 593
different persons	1 725	1 941	1 932	5 598

Table 5.4: Agreement statistics for collected triplets

The distribution of differences between two possible answers in the collected data is shown in table 5.4. As we can see, results are split in almost equal proportions between levels of agreement in all data sets (train, validation and test). In triplets where all images belong to the same person the distribution is slightly skewed towards higher agreement and the opposite is true for triplets with all different persons. This is expected because even though people are able to abstract facial expression from an image of a face, it is easier to make the correct decision when the only difference is the facial expression itself.

Chapter 6

Experiments

We have developed the methodology for the given task of representation learning of facial expressions (see chapter 4) and collected additional data necessary for the imposing human-like topological constraints onto the latent space (see chapter 5). In this chapter all the models are trained on the two data sets, posed and spontaneous, to test our methods and hypotheses.

6.1 Software

All the code for this project is written in Python. Tensorflow framework [1] is used for the main training of the models. Scikit-learn framework [20] is used for the principal component analysis (PCA) and support vector machines (SVM).

6.2 Preprocessing

The comparison of the data points from the two data sets has shown, that even though the dimensionality is the same, the placement of the facial landmarks is slightly different between the data sets in the chin and nose areas. This moves the original idea of enhancing the results by combining posed and spontaneous expressions into the domain of the knowledge transfer, which falls out of the scope of this project. It was therefore decided to conduct the experiments separately on each data set.

In the spontaneous data set, there is naturally a head tilt present in some data points. While it can be useful for detecting emotions, there is not much head pose variation in the data set. Therefore it was decided to rotate all data points to straight positions using the estimation of the angle of pitch, yaw and roll provided in the dataset.

In the posed data set the head pose relatively straight throughout the data set and no rotation is performed on these data.

The preliminary exploratory analysis of the data showed that the scale and shift are not consistent throughout the datasets. The preprocessing pipeline to make the data points comparable includes:

- rotating the point cloud to a straight position (only for the spontaneous data set),
- shifting the point cloud so that the point between eyes coincide with the origin,
- scaling the point cloud so that the distance between the eyes is 1.

The position of each eye was computed as a centre of points corresponding only to the lower eyelid to avoid shifting due to blinking.

For the training procedure, the data is further centred using the mean of the training portion of the corresponding data along each dimension for the purpose of numerical stability of the neural network.

Choosing neutral faces for the spontaneous data set

As we try to eliminate the effect of appearance an only encode facial expressions in the latent representation, the neutral facial expression is needed for each individual (sections 4.2, 4.3). Each individual in the posed data set has a corresponding neutral face. The spontaneous dataset contains no specifically marked neutral facial expressions. We decided to leverage the action units labelling available in the dataset to select one neutral face for each person. For each person, we selected time frames which have AU labels, but all AU are marked as non-present. We further select a single face with the minimal sum of distances to the other neutral faces for each individual. Therefore actual data points are selected and there is no averaging.

6.3 Evaluation

Since "usefulness" of the latent representation is not clearly defined, the evaluation is a bit tricky. We decided to use classification as the target task for evaluation mostly due to its interpretability and availability of the labels in the data sets.

The two data sets have different labelling. Labels in the posed data are facial expressions (emotion) and the task is a multi-class classification. In the spontaneous data, each labelled data point has an associated binary vector where each component shows presence or absence of the corresponding facial action unit (AU) and the task is, therefore, a multi-label classification.

Another evaluation technique that will be used is the number of satisfied triplets. This measure will reflect the topological coherency of the representation space.

6.3.1 Baseline

Our methods fall into the category of nonlinear dimensionality reduction techniques. One natural baseline in a representation learning is the original 249-dimensional data with no transformation. As a comparison, the most common dimensionality reduction method, linear Principal Component Analysis (for the method details see 12.1 in Bishop [3]), is also used with 50, 40, 30, and 20 dimensions.

The method of choice for the classification tasks is the Support Vector Machines with linear kernel (for the method details see 7.1 in Bishop [3]). On the one hand, in the original high-dimensional space everything is far away and often linearly separable and the results are reasonably good, on the other hand, to satisfy the topological constraints the representation should probably be linearly separable in the latent space (especially for the expression classification) and the nonlinear transformation should ideally eliminate the need for additional nonlinear transformation during the classification step.

To summarize, the compared representation are:

- no dimensionality reduction;
- linear dimensionality reduction using linear PCA;
- nonlinear dimensionality reduction using encoding part of our trained model (mapping from the original data-space to the learned latent representation);

6.3.2 Classification of stereotypical facial expressions on the posed data set

The posed data set contains 4 degrees of all 6 stereotypical facial expressions ("angry", "disgust", "sad", "happy", "surprised", "fear") and a neutral expression. The task of classifying these facial expressions will be one of the evaluation methods for the learned representation. In a multi-class setting, the SVM classifier is usually implemented as a set of one-vs-rest classifiers, one for each class.

Since the classes are relatively balanced in this data set, the standard accuracy will be used as a performance measure:

$$accuracy = \frac{\text{Number of correctly classified instances}}{\text{Total number of instances}}$$

The facial expression classification performance on the static posed data set for the original data and linear dimensionality reduction using Principal Component Analysis (linear PCA) is presented in table 6.1. As we can see, the performance has not decreased substantially for the dimensionality of 50 compared to the original 249-dimensional data space. But further lowering the number of dimensions leads to a noticeable performance drop. So even with the linear reduction method, it is possible to keep the majority of relevant characteristics in a much lower dimensional representation.

Dimensionality reduction	Dim	Train	Val	Test
None	249	0.874	0.684	0.72
Linear PCA	50	0.871	0.648	**0.724**
	40	0.769	0.652	0.708
	30	0.743	0.636	0.676
	20	0.701	0.548	0.676

Table 6.1: Facial expression classification accuracy on the posed data set

6.3.3 Classification of Action Units on spontaneous data set

Part of the data is labelled with facial action units (see 2.1). These data can be used as an additional evaluation tool. Only 12 AUs with the number of positive examples at least 10 % of the labelled data will be used. This is a multi-label binary classification task which is commonly addressed by training a separate classifier for each label (action unit in our case). We use linear Support Vector Machines (SVM) classifier.

The classes (1 - "present" or 0 - "absent" for each action unit) are unbalanced for most of the action units (for the detailed class balance see table 5.3). Therefore we use the "balanced" version of the SVM that put weights proportionally to the number of class occurrences. Also, the standard accuracy of classification is not an appropriate measure in this case, because always classifying AUs as absent results in a quite high accuracy. The F1 score is a much more informative measure. It is a harmonic mean of accuracy and precision, which is calculated as follows:

$$Precision = \frac{T_p}{T_p + F_p} \tag{6.1}$$

$$Recall = \frac{T_p}{T_p + F_n} \tag{6.2}$$

$$F1 = 2\frac{Precision * Recall}{Precision + Recall} \tag{6.3}$$

where T_p, F_p, F_n are the number of true positive, false positive and false negative predictions respectively.

The intuition is that accuracy is the ability not to classify positive example as a negative and precision is the ability to find all positive examples. The F1 score ties them together, ranging from 0 to 1 (the worst and the best value correspondingly). The metrics above are computed for each label (facial action unit) separately. To combine them in a single metric summarizing performance over all labels the weighted averages of precision and recall are computed over the labels. The weights proportional to the positive instances for each label to eliminate imbalance between different action units. Then the F1 score is computed.

Another commonly used metric for binary classifiers based on precision and recall is Area Under the Curve (AUC), which is computed based on different thresholds of the classifier. The weighted version summarizing all individual classifiers will be reported.

The F1 score and AUC for the baseline is summarized in table 6.2. We can see the substantial drop in classification quality only appears on the train data set, while validation and test data sets have almost no change in performance. Therefore the dimensionality of the original data can be substantially reduced with almost no loss in terms of the performed classification and topology tests even with a linear transformation.

Dimensionality reduction	**Dim**	**Train**		**Val**		**Test**	
		F1	**AUC**	**F1**	**AUC**	**F1**	**AUC**
None	249	0.782	0.898	0.598	0.766	0.633	0.766
Linear PCA	50	0.727	0.848	0.584	0.762	0.627	0.77
	40	0.72	0.839	0.574	0.76	0.627	**0.771**
	30	0.708	0.824	0.588	0.766	**0.646**	**0.771**
	20	0.694	0.808	0.59	0.756	0.635	0.757

Table 6.2: F1 score and AUC for the AU classification on the spontaneous data set

6.3.4 Distance preservation

Given the hypothesis about topology another test we will use is the number of satisfied triplets. It indicates the degree of similarity of the learned latent space topology and that of an assumed internal human representation, which can be useful for a number of applications. This evaluation will be conducted on both data sets (the generated "artificial" triplets will be used on the posed data set).

The number of satisfied triplets is the number of triplets where the Euclidean distance from the reference facial expression to the more similar one is smaller than the distance to the other expression in the triplet:

$$sat((i_{ref}, i_+, i_-)) = I(||z_{i_{ref}} - z_{i_+}||_2 \leq ||z_{i_{ref}} - z_{i_-}||_2) \tag{6.4}$$

The collected triplet data for the spontaneous data set have 3 levels of "confidence" (see "Redundancy" section in 5.2.2), and the number of satisfied triplets will be reported separately for each confidence level. Since the triplets for the posed data set were generated according to the reasonable rules (see 5.2.1), they all considered having the 100% reliability level (all the weights are 1).

The baseline performance on this test for the posed and spontaneous data sets are presented in tables 6.3 ,6.4. As we can see, there is no drop in the number of satisfied triplets with the decreased dimensionality.

Dimensionality reduction	**Dim**	**Train**	**Val**	**Test**
None	249	0.614	0.564	0.593
Linear PCA	50	0.613	0.566	0.593
	40	0.613	0.564	0.594
	30	0.613	0.566	0.592
	20	0.612	0.564	0.591

Table 6.3: The share of satisfied triplets on the posed data set

6.4 Training

Architecture

Both the encoding (reconstruction) and the decoding (generative) parts of the model are approximated with neural networks. All the lay-

Dimensionality reduction		None	Linear PCA			
Dimensions		249	50	40	30	20
Train	**5**	0.724	0.727	0.727	0.725	0.724
	3	0.611	0.609	0.612	0.611	0.613
	1	0.539	0.541	0.54	0.543	0.537
Validation	**5**	0.678	0.678	0.678	0.678	0.675
	3	0.605	0.608	0.606	0.608	0.608
	1	0.487	0.488	0.49	0.501	0.51
Test	5	0.747	0.747	**0.751**	0.743	**0.751**
	3	0.615	**0.618**	**0.618**	0.608	0.614
	1	0.549	0.545	0.549	0.549	0.548

Table 6.4: The share of satisfied triplets on the spontaneous data set

ers are fully connected with exponential linear unit (ELU, [5]) non-linearity. During training a fixed dropout is used (table 6.5). Different layer configurations are used during training, the details are reported in the corresponding sections. For the model 1 the encoding and decoding part is symmetrical, for the model 2 and model 3 it is not possible as the additional neutral face is added to both encoder and decoder input.

Annealing of the divergence term

It is typical for the VAE-model to "over-regularize", more specifically, the KL-divergence loss is closer in the network to the input data (the gradient chain is shorter) than reconstruction loss and as a result during training the network first learns to map input to prior and only then slowly tries to reconstruct from that encoded values. This behaviour results in turning off some latent dimensions early so that the model does not use the full allowed capacity.

Most common way to improve the learning is to use a modified objective function [28]:

$$\mathcal{L}(\mathbf{X}) = -E_{q_\phi(Z|X)}\Big[\log p_\theta(X|Z)\Big] + \beta * KL(q_\phi(Z|X)||p(Z)) \tag{6.5}$$

and slowly increase β from 0 to 1 over a number of iterations. When

$\beta = 0$ the objective is the maximum likelihood estimation and equivalent to that of the standard autoencoder model (see 3.2). $\beta = 1$ corresponds to the normal VAE objective.

In all experiments, β is increased linearly with the number of iterations.

The usage of each dimension is monitored with:

- the KL divergence (both prior and approximated posterior is factorizable over dimensions);
- the norm of the corresponding part of weights of the first decoding layer.

Small KL divergence indicates the posterior is very close to the prior and we will see, it is always the case that the weight norm is also close to zero meaning the dimension is not used in the reconstruction (or the generative process).

Parameters

The Adam algorithm is used for optimization [17]. The training parameters that do not change during training and there fixed values are listed in table 6.5.

Parameter	**Notation**	**Posed**	**Spontaneous**
learning rate		1e-4	1e-4
dropout		0.9	0.9
batch size	B	200	250
triplet batch size	V	10	10
sample size	L	3	3
triplet sample size	M	10	10

Table 6.5: Fixed training parameters

The batch size is chosen to be 200/250, preliminary experiments showed that the number of iterations is more important than the number of epochs, e.g. the number of iterations to convergence and quality of the results for 500 sized batches are the same as for 200 sized batches but each iteration is more computationally expensive and takes longer.

Authors of the original VAE paper [16] say one sample from the posterior distribution is enough as long as the batch size is at least 100. We use 3 samples.

For the model 3 with triplets, we use batches of 10 triplets with the latent representation sampled 10 times from the posterior.

All models are trained for 100 000 iterations. Each 2500 iterations the model is evaluated by computing evidence lower bound (ELBO) on the validation data set. After the training is finished a saved model with the highest validation ELBO is considered the final model and used for further evaluations.

The values of EBLO are not reported because some hyper parameters, namely the reconstruction variance and the topological prior variance, affect ELBO and make it conceptually incomparable between different settings of those hyper parameters. So it is only used to select the best iteration during training.

6.5 Results on the static posed data set

We begin by training all the models on the posed data set of stereotypical facial expressions. This data set has some disadvantages, namely small size and the fact that the expressions are not natural or spontaneous. Nevertheless, all the data are labelled with emotion types (sad, happy, etc.) which can be used for evaluating the representation quality and visualization of the latent space.

6.5.1 Model 1

The first model we conduct experiments on is the standard variational autoencoder model (section 4.2) applied to the posed data set. We vary the layer architecture, reconstruction variance and the number of annealing iterations (iterations it takes for the objective to change from maximum likelihood to the normal VAE objective). "No" annealing iterations means $\beta = 1$ from the start.

The parameter configurations are listed in table 6.6. The results are shown in table 6.7 (the first row corresponds to the results on original

#	**Architecture**		**Reconstr variance**	**Anneal iter**
	Encoding	**Decoding**		
1	[249, 120, 60, 30]	symmetrical	0.0001	no
2	[249, 120, 60, 30]	symmetrical	0.0001	50 000
3	[249, 120, 60, 30]	symmetrical	0.0001	100 000
4	[249, 240, 120, 60, 30]	symmetrical	0.0001	no
5	[249, 240, 120, 60, 30]	symmetrical	0.0001	50 000
6	[249, 240, 120, 60, 30]	symmetrical	0.0001	100 000
7	[249, 120, 60, 30]	symmetrical	0.001	no
8	[249, 120, 60, 30]	symmetrical	0.001	50 000
9	[249, 120, 60, 30]	symmetrical	0.001	100 000
10	[249, 240, 120, 60, 30]	symmetrical	0.001	no
11	[249, 240, 120, 60, 30]	symmetrical	0.001	50 000
12	[249, 240, 120, 60, 30]	symmetrical	0.001	100 000

Table 6.6: Parameter configurations for the model 1

Dimensionality reduction	**Dim**	**Accuracy**	**Triplets**
None	249	0.684	0.564
config 1	30	0.544	0.563
config 2	30	0.52	0.571
config 3	30	0.544	0.565
config 4	30	0.544	0.565
config 5	30	**0.556**	**0.575**
config 6	30	0.536	0.572
config 7	20	0.536	0.572
config 8	25	0.544	0.557
config 9	29	0.544	0.56
config 10	17	0.512	0.556
config 11	21	0.524	0.554
config 12	27	0.516	0.557

Table 6.7: Results for the model 1 on the validation data set

data with no dimensionality reduction). The dimensionality reported in the results is the number of active dimensions used by the generative part of the model.

As we can see from the results 6.7, small reconstruction variance leads to all dimensions being active. When the variance is higher slower annealing results in higher effective latent dimensionality. In the classification accuracy is lower than the baseline models, the number of satisfied triplets is similar to the baseline.

6.5.2 Model 2

The latent space in model 1 tries to model not only the features relevant to facial expressions but also the appearance-related features (e.g. round face, wide nose, etc.), because the latent representation is further used to reconstruct the original face as close as possible to the original. In the model 2 (see 4.2) we try to eliminate the individual (appearance-related) features from the latent space by providing an additional input in the form of a neutral facial expression for each person. The idea is that the model does not have to encode all the details to successfully reconstruct a face, only the difference between an expression and a corresponding neutral face.

The configurations of parameters and the corresponding results are presented in tables 6.8 and 6.9.

#	Architecture		Reconstr	Anneal
	Encoding	**Decoding**	**variance**	**iter**
1	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0001	no
2	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0001	50 000
3	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0001	100 000
4	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0005	no
5	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0005	50 000
6	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0005	100 000
7	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	no
8	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	50 000
9	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	100 000

Table 6.8: Parameter configurations for the model 2

Dimensionality reduction	**Dim**	**Accuracy**	**Triplets**
None	249	0.684	0.564
config 1	30	0.62	0.594
config 2	30	0.628	0.598
config 3	30	**0.664**	0.595
config 4	16	0.616	0.593
config 5	21	0.636	0.58
config 6	25	0.624	0.572
config 7	13	0.592	**0.606**
config 8	15	0.616	0.581
config 9	17	0.62	0.572

Table 6.9: Results for the model 2 on the validation data set

As we can see from the result, both performance measures, classification accuracy and the number of satisfied triplets, improved. The improvement in accuracy is about 10 %, but it is still worse than the baseline, the topology performance metric has also increased and is higher than the baseline.

6.5.3 Model 3

We try to further improve the quality of the latent representation by enforcing a specific topology based on the triplets (in the posed data set they were "artificially" generated, for more detail see 5.2.1). The intuition behind this adjustment is that the latent space with human-like similarity measure will also help to improve on other tasks, e.g. the facial expression classification.

In this model, there is an additional parameter, the topological variance. The smaller this parameter the higher the penalty for not satisfying the triplets in the latent space.

There are to possible ways to modify the objective function to perform the annealing (6.4):

- only anneal the KL-divergence term,
- anneal the whole prior, i.e. sum of the KL-divergence term and

the topology term.

We try both options to compare.

Anneal only the KL-divergence

The tested parameter configurations are listed in the table 6.10 and the corresponding results on the validation data set are shown in table 6.11.

#	Architecture		Variance		Anneal iter
	Encoding	Decoding	Reconstr	Topology	
1	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.01	no
2	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.01	50 000
3	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.01	100 000
4	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	no
5	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	50 000
6	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	100 000
7	[498, 480, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	100 000
8	[498, 480, 240, 120, 60, 30]	[279, 300, 500, 300, 249]	0.001	0.04	100 000
9	[498, 600, 400, 200, 100, 50]	[299, 300, 400, 300, 249]	0.001	0.04	100 000
10	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0005	0.04	100 000
11	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0001	0.04	100 000

Table 6.10: Parameter configurations for the model 3 with annealing only the KL-divergence

Anneal the KL-divergence and the topology term

The tested parameter configurations are listed in the table 6.12 and the corresponding results on the validation data set are shown in table 6.13.

The is no definite answer to which annealing scheme is the best. In general, the second option seems better.

The number of satisfied triplets increased, with is expected since it was introduced as a part of the loss function. The classification accuracy is also increased substantially compared to the previous model and is now higher than the baseline.

Dimensionality reduction	**Dim**	**Accuracy**	**Triplets**
None	249	0.684	0.564
config 1	30	0.62	0.742
config 2	30	0.636	0.748
config 3	30	0.644	0.754
config 4	24	0.66	0.741
config 5	22	0.604	0.734
config 6	30	0.664	0.75
config 7	30	**0.724**	**0.761**
config 8	30	0.676	0.751
config 9	39	0.708	0.76
config 10	30	0.68	0.75
config 11	30	0.656	0.693

Table 6.11: Results for the model 3 with annealing only the KL-divergence on the validation data set

#	**Architecture**		**Variance**		**Anneal iter**
	Encoding	**Decoding**	**Reconstr**	**Topology**	
1	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	50 000
2	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	100 000
3	[498, 480, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	50 000
4	[498, 600, 400, 200, 100, 50]	[299, 300, 400, 300, 249]	0.001	0.04	50 000
5	[498, 400, 200, 100, 50]	[299, 300, 400, 300, 249]	0.001	0.04	50 000
6	[498, 480, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	100 000
7	[498, 480, 240, 120, 60, 30]	[279, 300, 500, 300, 249]	0.001	0.04	100 000
8	[498, 600, 400, 200, 100, 50]	[299, 300, 400, 300, 249]	0.001	0.04	100 000
9	[498, 480, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.04	100 000
10	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0005	0.04	100 000
11	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0001	0.04	100 000

Table 6.12: Parameter configurations for the model 3 with annealing the KL-divergence and the topology term

Dimensionality reduction	**Dim**	**Accuracy**	**Triplets**
None	249	0.684	0.564
config 1	22	0.676	0.747
config 2	27	0.68	0.73
config 3	25	0.676	0.746
config 4	22	0.704	0.729
config 5	26	0.7	0.725
config 6	25	**0.708**	**0.76**
config 7	27	0.704	0.748
config 8	33	0.696	0.74
config 9	27	0.696	0.745
config 10	30	0.656	0.714
config 11	30	0.648	0.674

Table 6.13: Results for the model 3 with annealing the KL-divergence and the topology term on the validation data set

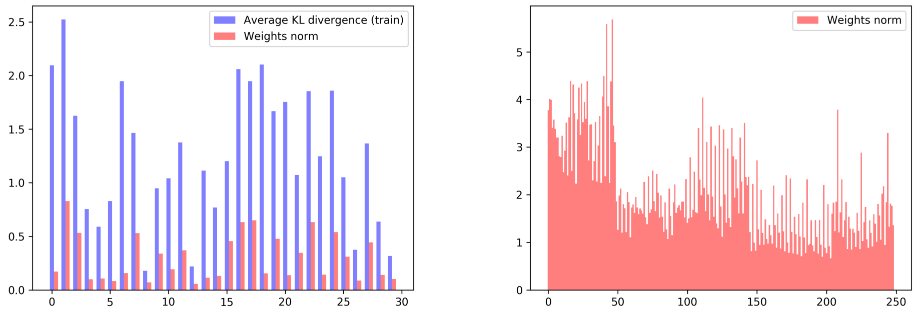
6.5.4 Selection and analysis of the best model

The best model on validation data set is the model 3 with annealing only the KL-divergence term with the following configuration (config 7): encoding [498, 480, 240, 120, 60, 30], decoding [279, 300, 400, 300, 249], reconstruction variance 0.001, topological variance 0.04, annealing over 100 000 iterations.

As we can see in the comparison in table 6.14, our model outperforms both baselines (the original data and linear PCA).

Dimensionality reduction	**Dim**	**Accuracy**	**Triplets**
None	249	0.72	0.593
PCA	30	0.676	0.592
Our best model 1	30	0.64	0.592
Our best model 2	30	0.72	0.67
Our best model 3	30	**0.736**	**0.798**

Table 6.14: Performance comparison on the test data set. Best performing models are taken from tables 6.1, 6.3, 6.7, 6.9, 6.11



(a) Latent space

(b) Neutral face

Figure 6.1: KL-divergence and weight norm for the first decoding layer

We explore the learned latent representation by plotting the KL-divergence and the norms of the first layer of the generative (decoding) part of the model (figure 6.1). As we can see all the dimensions of neutral faces is used to reconstruct a face. In the latent space, some dimensions are used more heavily with higher divergence from the prior, but the model used the full allowed capacity. Example of a reconstructed face is shown in figure 6.2. The resulting reconstruction is more smooth than the original face since the model assumes Gaussian noise in the output.

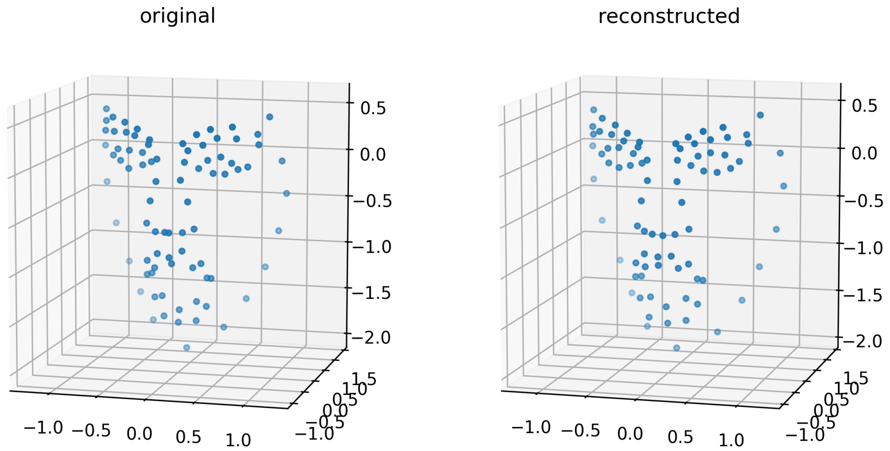
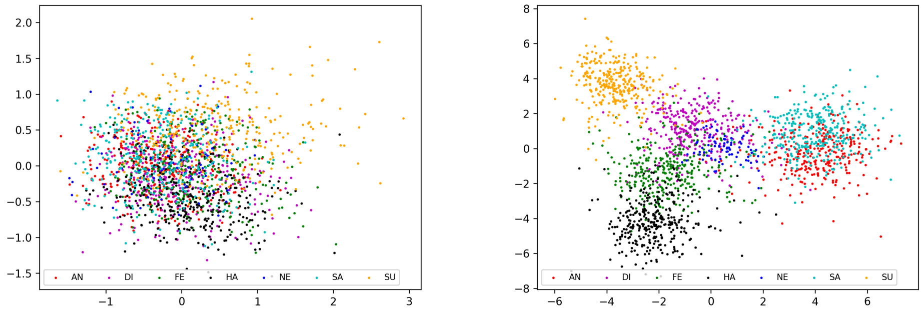


Figure 6.2: Reconstruction example on a test data point of the posed data set



(a) Original data

(b) Our best model

Figure 6.3: Visualization in 2D using linear PCA

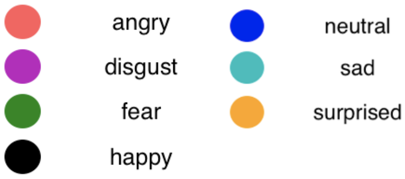
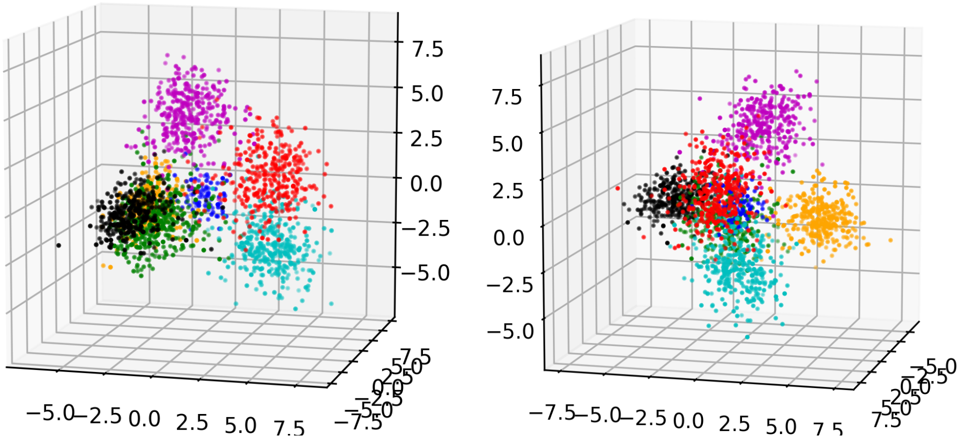


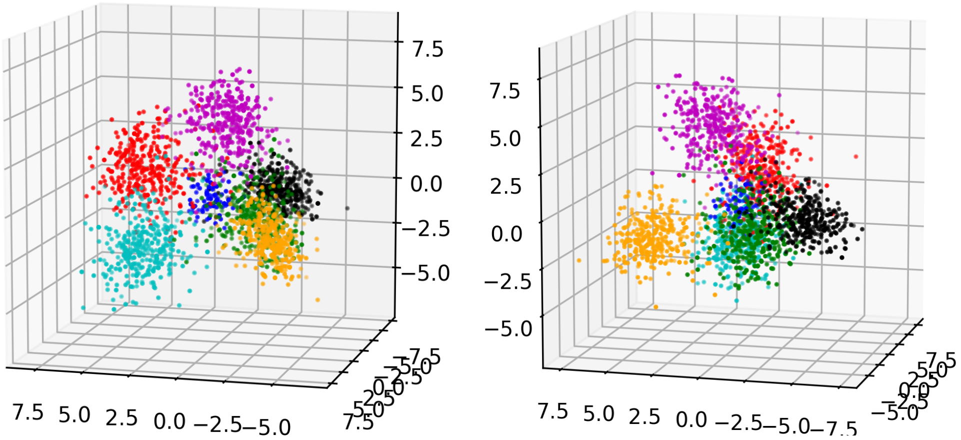
Figure 6.4: Classes of stereotypical facial expressions on the posed data set

Figure 6.3 compares the original data and the learned representation on the train part of the posed data set. Both representations were transformed using linear PCA, the colors correspond to the stereotypical facial expressions (figure 6.4). We already can see that our model provides a good linear separation between classes while the original data does not. This representation is further explored in 3d in figure 6.5, where the 3 principal components of the latent space are shown from 4 with different view angles with rotation around the 3rd component (z-axis in the picture).

One of the main advantages of using a generative model is the possibility to explore the role of each dimension by varying one latent dimension at a time generating new data points. The 2 dimensions with the clearest influence on the different part of a face are shown in figure 6.6 along with the neutral face (one of the train persons). The reconstructed neutral face (with the latent vector set to zeros) looks very similar to the original, one dimension is clearly responsible for



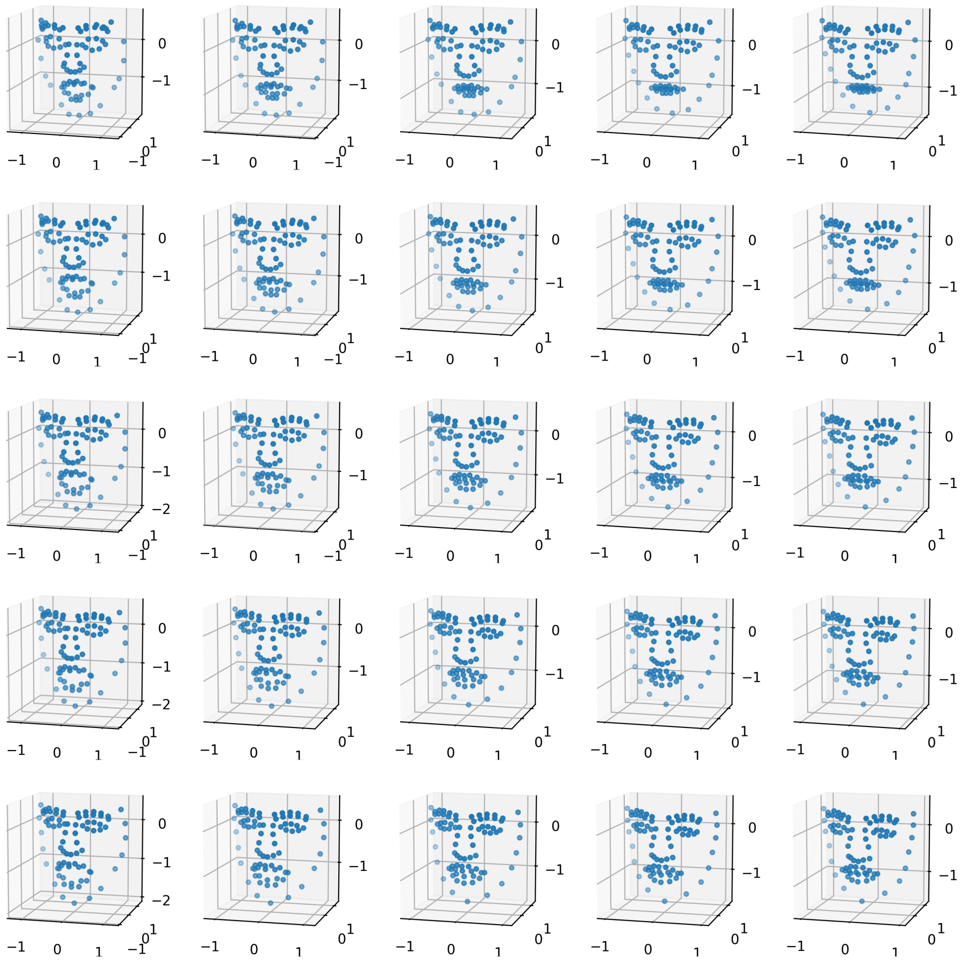
(a) Rotation 1 (b) Rotation 2



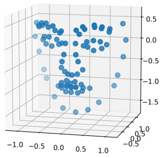
(c) Rotation 3 (d) Rotation 4

Figure 6.5: Visualization of our model on the train data set in 3D using linear PCA (every 90°)

the mouth opening and the other for eyes opening.



(a) Generated faces



(b) Neutral input face

Figure 6.6: Generation of a new facial expressions given one neutral face on the posed data set, z_{16}, z_2 are changing along horizontal and vertical axes respectively, the grid is $[-5, -2.5, 0, 2.5, 5]$ for both, all other latent dimensions are set to 0

6.6 Results on spontaneous data set

Our models are also trained on the spontaneous data set. To learn a disentangled representation it is important to have data with variations along the assumed disentangles latent dimensions. For example, if in the whole data set very few blinking expressions occur, the model less likely to learn blinking as a separate independent dimension. The main advantage of these spontaneous data is the more natural variations in facial expressions as opposed to the stereotypical and static expressions in the posed data set. For this data set, we collected the triplet data from real people using crowd-sourcing (see 5.2.2).

6.6.1 Model 1

All the experiments presented in this section correspond to the standard VAE model applied to the spontaneous data set(4.1).

Since the VAE can potentially "turn off" some dimensions (in which case the posterior for those dimensions is the same as the prior and the corresponding decoding weights are almost 0), the number of "active" dimensions is reported for all the models.

The tested parameter configurations are listed in the table 6.17 and the corresponding results on the validation data set are shown in table 6.18. The results are worse than the baseline.

#	**Architecture**		**Reconstruction variance**	**Anneal iterations**
	Encoding	**Decoding**		
1	[249, 120, 60, 30]	symmetrical	0.0001	no
2	[249, 120, 60, 30]	symmetrical	0.0001	50 000
3	[249, 120, 60, 30]	symmetrical	0.0001	100 000
4	[249, 240, 120, 60, 30]	symmetrical	0.0001	no
5	[249, 240, 120, 60, 30]	symmetrical	0.0001	50 000
6	[249, 240, 120, 60, 30]	symmetrical	0.0001	100 000
7	[249, 300, 200, 100, 50]	symmetrical	0.0001	no
8	[249, 300, 200, 100, 50]	symmetrical	0.0001	50 000
9	[249, 300, 200, 100, 50]	symmetrical	0.0001	100 000

Table 6.15: Parameter configurations for the model 1

Dimensionality reduction	**Dim**	**F1**	**AUC**	**Triplets**		
				5	**3**	**1**
None	249	0.598	0.766	0.678	0.605	0.487
config 1	30	0.576	0.748	0.667	0.592	0.524
config 2	30	0.591	0.758	**0.672**	0.589	0.499
config 3	30	0.594	0.759	**0.672**	0.585	0.512
config 4	30	0.584	0.75	0.652	0.586	0.519
config 5	30	0.589	0.765	0.667	0.595	0.51
config 6	30	**0.597**	**0.769**	0.655	0.582	0.505
config 7	50	0.58	0.746	0.67	0.576	0.512
config 8	50	0.588	0.75	0.663	0.59	0.507
config 9	50	0.586	0.753	0.667	0.567	0.521

Table 6.16: Results for the model 1 on the validation data set

6.6.2 Model 2

All the experiments presented in this section correspond to the model with neutral faces as an additional input (4.2).

We see that the additional input in the form of a neutral face is used by the model. The typical norm per dimension graph of the weights of the first decoding layer is shown in figure 6.7. The graph has a very similar appearance for all trained models with neutral faces (model 2 and model 3).

#	**Architecture**		**Reconstruction variance**	**Anneal iterations**
	Encoding	**Decoding**		
1	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.0001	no
2	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.01	no
3	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.01	50 000
4	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.01	100 000
5	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	no
6	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	50 000
7	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	100 000

Table 6.17: Parameter configurations for the model 2

The tested parameter configurations are listed in the table 6.17 and the corresponding results on the validation data set are shown in table 6.18. As we can see in general smaller variance results in larger

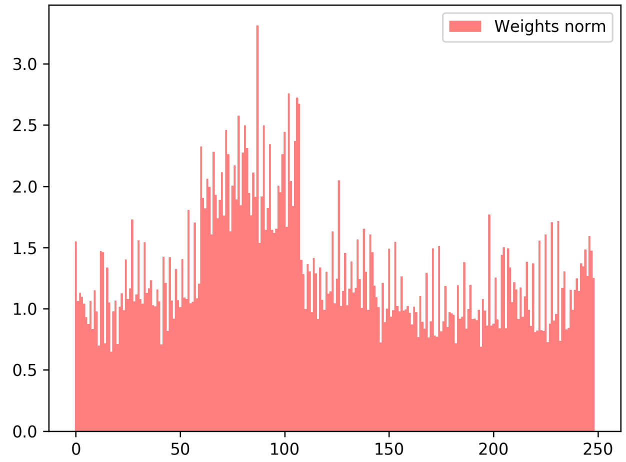


Figure 6.7: Typical weights norms of the first decoding layer from neutral face

Dimensionality reduction	**Dim**	**F1**	**AUC**	**Triplets**		
				5	**3**	**1**
None	249	0.598	0.766	0.678	0.605	0.487
config 1	30	**0.603**	**0.746**	0.635	0.59	0.502
config 2	5	0.576	0.691	**0.637**	0.598	0.505
config 3	9	0.588	0.715	0.63	0.57	0.526
config 4	11	0.587	0.728	0.626	0.573	0.527
config 5	13	0.602	0.736	0.613	0.564	0.502
config 6	16	0.594	0.73	0.601	0.582	0.498
config 7	19	0.596	0.734	0.609	0.574	0.501

Table 6.18: Results for the model 2 on the validation data set

number of active dimension, annealing has a similar effect.

The does not seem to be an improvement compared to the previous model. This might be due to the fact, that the neutral faces were chosen automatically and their quality is lower than that of posed neutral expressions. Therefore the help provided for the model by a low-quality "neutral" face is much smaller.

6.6.3 Model 3

All the experiments presented in this section correspond to the model with neutral faces as additional input and an additional topological prior with triplets (4.3).

There are two possible ways to modify the objective function to perform the annealing (6.4):

- only anneal the KL-divergence term,
- anneal the whole prior, i.e. sum of the KL-divergence term and the topology term.

We try both options to compare.

Anneal only the KL-divergence

The tested parameter configurations are listed in the table 6.19 and the corresponding results on the validation data set are shown in table 6.20.

#	Architecture		Variance		Anneal iterations
	Encoding	**Decoding**	**Reconstr**	**Topology**	
1	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.001	no
2	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.001	50 000
3	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.001	100 000
4	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.0001	no
5	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.0001	50 000
6	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.0001	100 000

Table 6.19: Parameter configurations for the model 3 with annealing only the KL-divergence

Dimensionality reduction	**Dim**	**F1**	**AUC**	**Triplets**		
				5	**3**	**1**
None	249	0.598	0.766	0.678	0.605	0.487
config 1	17	**0.624**	0.759	0.758	0.655	0.518
config 2	20	0.61	0.754	0.741	0.659	0.512
config 3	24	0.618	0.752	0.762	0.65	0.523
config 4	28	0.622	0.765	**0.837**	0.729	0.544
config 5	30	0.618	**0.768**	0.834	0.742	0.566
config 6	30	0.614	0.753	0.848	0.736	0.543

Table 6.20: Results for the model 3 with annealing only the KL-divergence on the validation data set

Anneal the KL-divergence and the topology term

The tested parameter configurations are listed in the table 6.21 and the corresponding results on the validation data set are shown in table 6.22.

#	**Architecture**		**Variance**		**Anneal iterations**
	Encoding	**Decoding**	**Reconstr**	**Topology**	
1	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.001	50 000
2	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.001	100 000
3	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.0001	50 000
4	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.0001	100 000
5	[498, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.0005	50 000
6	[498, 480, 240, 120, 60, 30]	[279, 300, 400, 300, 249]	0.001	0.0005	50 000
7	[498, 600, 400, 200, 100, 50]	[299, 300, 400, 300, 249]	0.001	0.0005	50 000
8	[498, 400, 200, 100, 50]	[299, 300, 400, 300, 249]	0.001	0.0005	50 000

Table 6.21: Parameter configurations for the model 3 with annealing the whole prior

As we can see from the results, additional topological constraints help improve performance on the action units classification task, and also naturally increase the topological coherence measure which, as we saw on the posed data set, connected to the facial expression classification performance given that the triplets are consistent with the rules we used to generate the artificial triplets on the posed data set (see 5.2.1).

Dimensionality reduction	**Dim**	**F1**	**AUC**	**Triplets**		
				5	**3**	**1**
None	249	0.598	0.766	0.678	0.605	0.487
config 1	16	0.605	0.749	0.73	0.65	0.519
config 2	22	0.606	0.749	0.736	0.649	0.527
config 3	30	**0.627**	**0.764**	**0.839**	0.726	0.544
config 4	28	0.62	0.763	0.839	0.727	0.555
config 5	19	0.621	0.759	0.779	0.682	0.526
config 6	25	0.603	0.75	0.767	0.695	0.543
config 7	21	0.618	0.755	0.756	0.687	0.516
config 8	22	0.62	0.763	0.779	0.678	0.529

Table 6.22: Results for the model 3 with annealing the whole prior on the validation data set

6.6.4 Selection and analysis of the best model

The best model on validation data set is the model 3 with annealing the whole prior and the following configuration: encoding [498, 240, 120, 60, 30], decoding [279, 300, 400, 300, 249], reconstruction variance 0.001, topological variance 0.0001, annealing over 50 000 iterations.

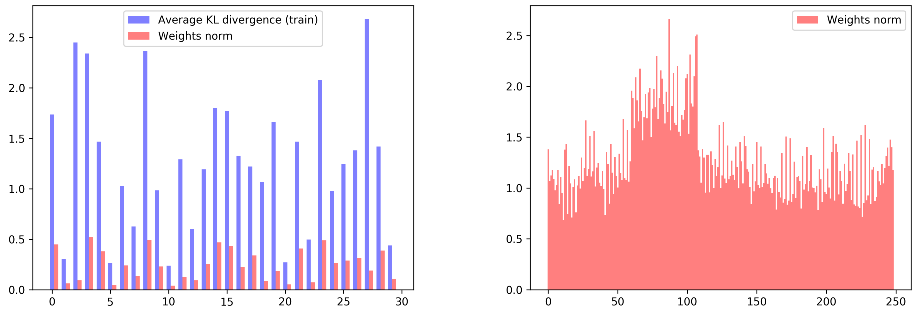
The performance comparison with the baseline models on the test part of the spontaneous data set is given in the table 6.23. On the AU classification task, our model outperforms the baselines on the F1 score, but the Area Under the Curve measure is slightly lower than that of the PCA. The triplet coherency is much higher than the baseline for the high and medium confidence triplets.

Figure 6.9 shows an example of facial reconstruction of a test data point. As we can see, the reconstructed facial expression is very similar to the original one. Since the reconstructed face is the mean of the Gaussian in the generative process, i.e. the original data is assumed to have Gaussian noise, the reconstructed facial landmarks are smoother than true data.

We explore the latent dimensionality via the generative process in the same manner as for the posed data set in 6.5.4. Figure 6.10 shows faces generated by changing the values of two latent dimensions while all

Dimensionality reduction	**Dim**	**F1**	**AUC**	**Triplets**		
				5	**3**	**1**
None	249	0.633	0.766	0.747	0.615	0.549
PCA	30	0.646	**0.771**	0.743	0.608	0.549
Our model 1	30	0.637	0.753	0.706	0.593	0.494
Our model 2	30	0.644	0.752	0.647	0.584	0.495
Our model 3	30	**0.658**	0.764	**0.849**	**0.716**	0.556

Table 6.23: Performance comparison on the test data set. Best performing models are taken from tables 6.2, 6.4, 6.16, 6.18, 6.22



(a) Latent space

(b) Neutral face

Figure 6.8: KL-divergence and weight norm for the first decoding layer

other dimensions are fixed (zeros). We can see, that the vertical axis corresponds to smiling and the horizontal axis to lowering/raising eyebrows. Those dimensions were chosen for visibility purposes, as many other dimensions correspond to subtle changes in facial expressions.

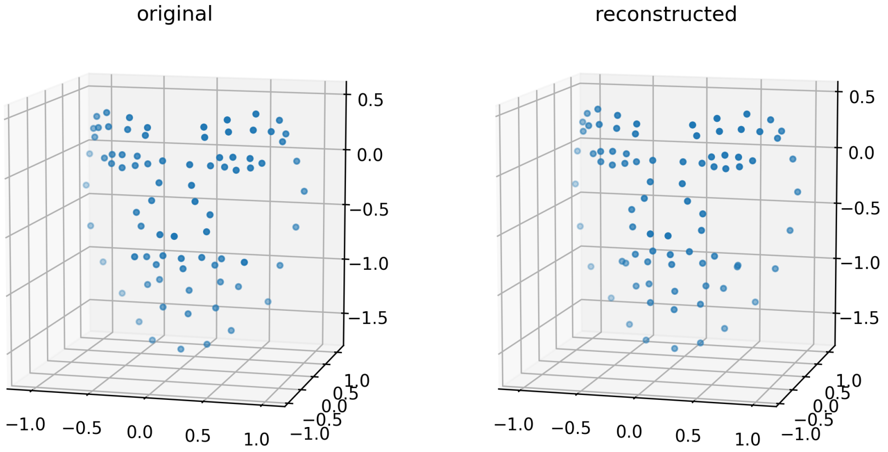
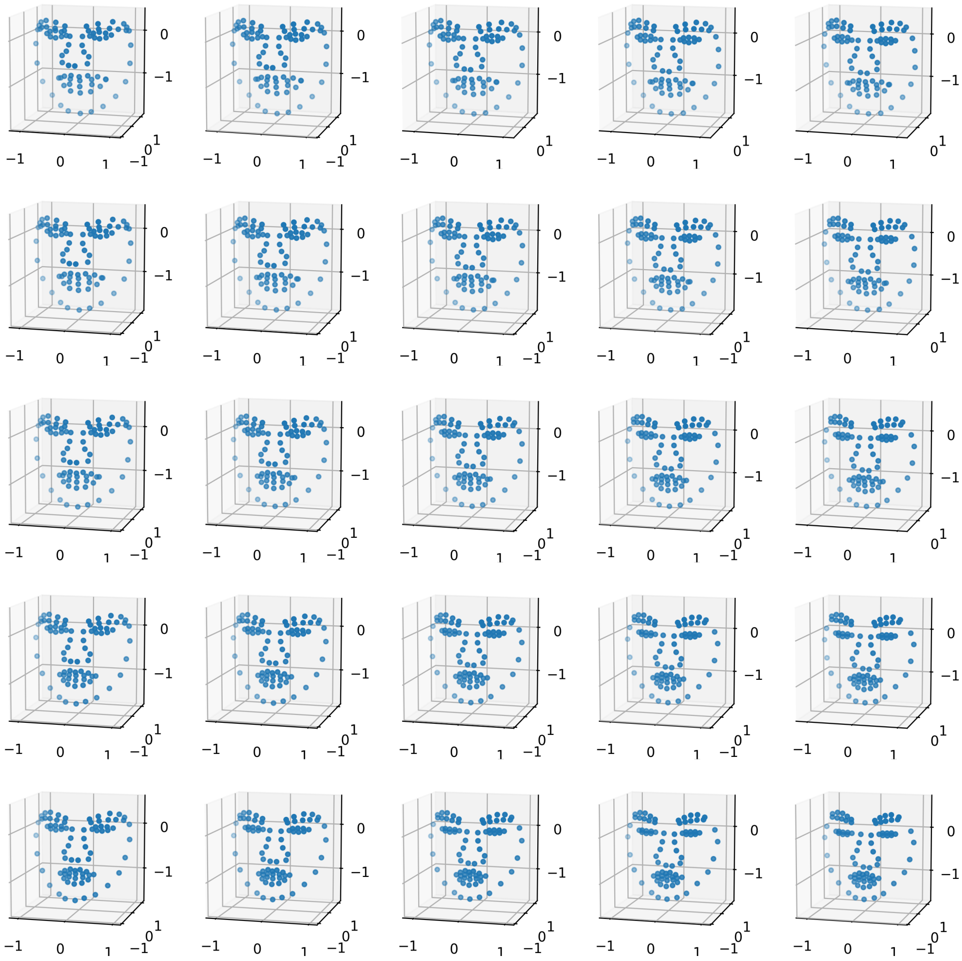
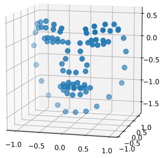


Figure 6.9: Reconstruction example on a test data point of the posed data set



(a) Generated faces



(b) Neutral input face

Figure 6.10: Generation of a new facial expressions given one neutral face on the spontaneous data set, z_{15}, z_2 are changing along horizontal and vertical axes respectively, the grid is $[-5, -2.5, 0, 2.5, 5]$ for both, all other latent dimensions are set to 0

Chapter 7

Conclusions

7.1 Summary

The goal of this works was to find a data-driven latent representation for the human facial expressions and to test the hypothesis that imposing a topological structure on the latent space similar to that of an assumed internal human representation would be beneficial on different tasks, such as classification of facial expressions.

The methodology was iteratively built on the basis of variational autoencoder. To eliminate the individual features classic VAE was modified to include neutral faces so that the model can focus on learning only the deviation of a facial expression from a neutral one. We incorporated the topological constraints as an additional component in the prior distribution of the latent variable. The original representation and linear PCA were used as baseline models.

The models were trained and tested on two data sets. While the data formats in the data sets are the same, the size, the labelling, and the variation in the data are different. On the posed data we saw that the standard VAE was performing worse than the baselines, but adding a neutral face increased the classification accuracy up to the baseline levels. Including the topological prior helped structure the latent space in a way that is coherent with a human similarity assessment and raised the classification accuracy above the baseline.

The latent representation of the posed data set showed clear linear separation between classes of stereotypical facial expressions. Even though the triplets for that data set were artificially generated based on labels, the assumptions were very mild and did not concern the global ordering of classes.

In the spontaneous data set, the only labels were facial action units. On this data set, the action unit classification performance of our model was very similar to the baseline. This probably can be explained by the fact that humans consider the holistic facial expression when making similarity comparison and therefore the number of satisfied triplets correlates much stronger with the emotion classification performance, rather than with the more granular action units.

On the both data sets including the topological component in the latent prior increased the number of satisfied triplets on the test part of the data (unseen during training). This fact indicates that the model does not just overfit the training triplets, but rather uses them to structure the latent space in a way that is more consistent with human similarity assessments.

One of the benefits of the generative probabilistic model is the ability to explore the role of each dimension through generations of new samples. As we saw on both data sets it is possible to identify dimensions responsible for very specific facial actions, such as closing and opening the mouth, smiling, blinking.

The best learned representations perform better of the same on classification tasks, are nicely structured in the latent space and are interpretable through the means of the generative process.

7.2 Future work

Even though the result seem to be quite interesting in itself, where is a number of possibilities for the future work. So far we only worked with static facial expressions (even when using the dynamic data set) while they are dynamic by natural. Therefore incorporating temporal dynamics in the model is the most obvious next step that can help learning a temporally consistent representations (with smooth trajectories in the latent space). Possible ways to enforce smoothness in the latent space include using a temporal prior on the latent space [26] and using Variational recurrent autoencoders [11].

Another option to explore is different prior on the latent space. For example, each latent dimension could have an independent Beta distribution (variable in the range [0,1]) as a prior to mimic action unit activations. The main limitation here is that the distribution should be reparameterizable as a differentiable transformation of the distribution parameters and an auxiliary noise variable to allow backpropagation. Beta distribution can not be reparameterized in this way, nevertheless it can be approximated by the Kumaraswamy distribution which will allow easy sampling and a closed-form approximation of its KL-divergence from the Beta distribution. This property is exploited in the work of Nalisnick and Smyth [19] on Stick-Breaking Variational Autoencoder (SB-VAE). Another possibility is using a Gaussian Mixture Model as a prior distribution on the latent space [8].

This work can also be improved by finding more evaluation tasks to test on and compare the learned latent representation. For example, the spontaneous data set does not have labels for stereotypical facial expressions (emotions), which would give an interesting insight into the latent space. Those labels can be collected using crowd sourcing.

Bibliography

[1] Martin Abadi et al. "Tensorflow: Large-scale machine learning on heterogeneous distributed systems". In: *arXiv preprint arXiv:1603.04467* (2016).

[2] Yoshua Bengio, Aaron Courville, and Pascal Vincent. "Representation learning: A review and new perspectives". In: *IEEE transactions on pattern analysis and machine intelligence* 35.8 (2013), pp. 1798–1828.

[3] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[4] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. "Importance weighted autoencoders". In: *arXiv preprint arXiv:1509.00519* (2015).

[5] Djork-Arné Clevert, Thomas Unterthiner, and Sepp Hochreiter. "Fast and accurate deep network learning by exponential linear units (elus)". In: *ICLR*. 2016.

[6] Zhenwen Dai et al. "Variational auto-encoded deep Gaussian processes". In: *arXiv preprint arXiv:1511.06455* (2015).

[7] Andreas Damianou and Neil Lawrence. "Deep gaussian processes". In: *Artificial Intelligence and Statistics*. 2013, pp. 207–215.

[8] Nat Dilokthanakul et al. "Deep unsupervised clustering with gaussian mixture variational autoencoders". In: *arXiv preprint arXiv:1611.02648* (2016).

[9] Carl Doersch. "Tutorial on variational autoencoders". In: *arXiv preprint arXiv:1606.05908* (2016).

[10] Paul Ekman and Wallace V Friesen. "Facial action coding system". In: (1977).

[11] Otto Fabius and Joost R van Amersfoort. "Variational recurrent auto-encoders". In: *arXiv preprint arXiv:1412.6581* (2014).

[12] Irina Higgins et al. "Early visual concept learning with unsupervised deep learning". In: *arXiv preprint arXiv:1606.05579* (2016).

[13] Geoffrey E Hinton and Ruslan R Salakhutdinov. "Reducing the dimensionality of data with neural networks". In: *science* 313.5786 (2006), pp. 504–507.

[14] Elad Hoffer and Nir Ailon. "Deep metric learning using triplet network". In: *International Workshop on Similarity-Based Pattern Recognition*. Springer. 2015, pp. 84–92.

[15] Diederik P Kingma, Tim Salimans, and Max Welling. "Improving variational inference with inverse autoregressive flow". In: *arXiv preprint arXiv:1606.04934* (2016).

[16] Diederik P Kingma and Max Welling. "Auto-encoding variational bayes". In: *arXiv preprint arXiv:1312.6114* (2013).

[17] Diederik Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *ICLR*. 2015.

[18] Alireza Makhzani et al. "Adversarial autoencoders". In: *arXiv preprint arXiv:1511.05644* (2015).

[19] Eric Nalisnick and Padhraic Smyth. "Stick-Breaking Variational Autoencoders". In: *arXiv preprint arXiv:1605.06197* (2016).

[20] Fabian Pedregosa et al. "Scikit-learn: Machine learning in Python". In: *Journal of Machine Learning Research* 12.Oct (2011), pp. 2825–2830.

[21] Christopher Pramerdorfer and Martin Kampel. "Facial Expression Recognition using Convolutional Neural Networks: State of the Art". In: *arXiv preprint arXiv:1612.02903* (2016).

[22] Danilo Jimenez Rezende and Shakir Mohamed. "Variational inference with normalizing flows". In: *arXiv preprint arXiv:1505.05770* (2015).

[23] Georgia Sandbach et al. "Static and dynamic 3D facial expression recognition: A comprehensive survey". In: *Image and Vision Computing* 30.10 (2012), pp. 683–697.

[24] Casper Kaae Sønderby et al. "Ladder variational autoencoders". In: *Advances in Neural Information Processing Systems*. 2016, pp. 3738–3746.

[25] Michalis K Titsias and Neil D Lawrence. “Bayesian Gaussian process latent variable model”. In: *International Conference on Artificial Intelligence and Statistics*. 2010, pp. 844–851.

[26] Raquel Urtasun et al. “Topologically-constrained latent variable models”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1080–1087.

[27] Pascal Vincent et al. “Extracting and composing robust features with denoising autoencoders”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 1096–1103.

[28] Serena Yeung et al. “Tackling Over-pruning in Variational Autoencoders”. In: *arXiv preprint arXiv:1706.03643* (2017).

[29] Lijun Yin et al. “A 3D facial expression database for facial behavior research”. In: *Automatic face and gesture recognition, 2006. FGR 2006. 7th international conference on*. IEEE. 2006, pp. 211–216.

[30] Xing Zhang et al. “A high-resolution spontaneous 3d dynamic facial expression database”. In: *Automatic Face and Gesture Recognition (FG), 2013 10th IEEE International Conference and Workshops on*. IEEE. 2013, pp. 1–6.

www.kth.se