Modeling biomedical breathing signals with convolutional deep probabilistic autoencoders

Oscar Pastor-Serrano, Danny Lathouwers, PhD, and Zoltán Perkó, PhD

Delft University of Technology,
Department of Radiation Science and Technology, Delft, Netherlands

October 30, 2020

Abstract

One of the main problems with biomedical signals is the limited amount of patient-specific data and the significant amount of time needed to record a sufficient number of samples for diagnostic and treatment purposes. We explore the use of Variational Autoencoder (VAE) and Adversarial Autoencoder (AAE) algorithms based on one-dimensional convolutional neural networks in order to build generative models able to capture and represent the variability of a set of unlabeled quasi-periodic signals using as few as 10 parameters. Furthermore, we introduce a modified AAE architecture that allows simultaneous semi-supervised classification and generation of different types of signals. Our study is based on physical breathing signals, i.e. time series describing the position of chest markers, generally used to describe respiratory motion. The time series are discretized into a vector of periods, with each period containing 6 time and position values. These vectors can be transformed back into time series through an additional reconstruction neural network and allow to generate extended signals while simplifying the modeling task. The obtained models can be used to generate realistic breathing realizations from patient or population data and to classify new recordings. We show that by incorporating the labels from around 10-15% of the dataset during training, the model can be guided to group data according to the patient it belongs to, or based on the presence of different types of breathing irregularities such as baseline shifts. Our specific motivation is to model breathing motion during radiotherapy lung cancer treatments, for which the developed model serves as an efficient tool to robustify plans against breathing uncertainties. However, the same methodology can in principle be applied to any other kind of quasi-periodic biomedical signal, representing a generically applicable tool.

1 Introduction

Biomedical data is the driving force behind most modern advances in medicine. The use of biomedical data is however associated to a series of problems such as the lack of reliable models capable of simulating data with clinical precision, the absence of personalized models for diagnosis, or the lack of labeled samples due to labels containing personal features that compromise privacy or simply not being recorded [1]. Some of the initial efforts to model biomedical data include symbolic approaches: e.g., McSharry et al. [2] developed an electrocardiogram (ECG) model based on three coupled ordinary differential equations, and George et al. [3] introduced a sinusoidal model to represent breathing.

Recent advances in Deep Learning and the introduction of frameworks such as the Variational Autoencoder (VAE) [12] or Generative Adversarial Networks (GANs) [16] have resulted in the development of a wide variety of methods capable of generating and classifying biomedical signals, most of them applied to ECG data. Yildirim et al. [4] propose an efficient algorithm based on autoencoder ANNs that compresses ECG signals but lacks generative capabilities. Both Zhu et al. [5] and Delany et al. [6] propose generative models that combine different artificial neural network (ANN) architectures (recurrent and...
2.1 Probabilistic generative models. Consider \( x \in \mathbb{R}^M \) to be a random vector over a vector space \( X \), with unknown underlying probability distribution \( p_{\text{data}}(x) \). Given a dataset \( D = \{x^{(i)}\}_{i=1}^{N_D} \) with \( N_D \) independent and identically distributed (i.i.d) data points, the goal is to model a probability distribution \( p_\theta(x) \) that approximates the unknown true probability distribution generating the data using a probabilistic graphical model with parameters \( \theta \). Let this probabilistic model be a latent variable model, which conditions the observed variable \( x \) to the unobserved random vector \( z \in \mathbb{R}^N \) over the latent space \( Z \) containing \( N \) latent variables that are assumed to capture the principal factors of variation in the data. The latent variable model represents the joint distribution of observed and unobserved variables and factorizes as \( p_\theta(x, z) = p_\theta(x|z)p(z) \). The (target) marginal distribution of the observed variables can be recovered as

\[
p_\theta(x) = \int_Z p_\theta(x, z)dz = \int_Z p_\theta(x|z)p(z)dz, \tag{1}
\]

where \( p(z) \) is the prior probability distribution over \( Z \) and \( p_\theta(x|z) \) is a conditional distribution of observed variables that can be parametrized using neural networks. In principle, the prior could be any function and it is not conditioned on the observations. New data can be generated by sampling a vector \( z \) from the prior, and then sampling a data point \( x \) from the distribution \( p_\theta(x|z) \) conditioned on \( z \).

The parameters \( \theta \) of the latent variable model are estimated via the maximum likelihood approach, by maximizing the (log-) marginal distribution of the observed data as

\[
\theta^* = \arg\max_{\theta} \sum_{x \in D} \log (p_\theta(x)) \simeq \arg\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}(x)} \log(p_\theta(x)), \tag{2}
\]

2 Background

2.1 Probabilistic generative models. Consider \( x \in \mathbb{R}^M \) to be a random vector over a vector space \( X \), with unknown underlying probability distribution \( p_{\text{data}}(x) \). Given a dataset \( D = \{x^{(i)}\}_{i=1}^{N_D} \) with \( N_D \) independent and identically distributed (i.i.d) data points, the goal is to model a probability distribution \( p_\theta(x) \) that approximates the unknown true probability distribution generating the data using a probabilistic graphical model with parameters \( \theta \). Let this probabilistic model be a latent variable model, which conditions the observed variable \( x \) to the unobserved random vector \( z \in \mathbb{R}^N \) over the latent space \( Z \) containing \( N \) latent variables that are assumed to capture the principal factors of variation in the data. The latent variable model represents the joint distribution of observed and unobserved variables and factorizes as \( p_\theta(x, z) = p_\theta(x|z)p(z) \). The (target) marginal distribution of the observed variables can be recovered as

\[
p_\theta(x) = \int_Z p_\theta(x, z)dz = \int_Z p_\theta(x|z)p(z)dz, \tag{1}
\]

where \( p(z) \) is the prior probability distribution over \( Z \) and \( p_\theta(x|z) \) is a conditional distribution of observed variables that can be parametrized using neural networks. In principle, the prior could be any function and it is not conditioned on the observations. New data can be generated by sampling a vector \( z \) from the prior, and then sampling a data point \( x \) from the distribution \( p_\theta(x|z) \) conditioned on \( z \).

The parameters \( \theta \) of the latent variable model are estimated via the maximum likelihood approach, by maximizing the (log-) marginal distribution of the observed data as

\[
\theta^* = \arg\max_{\theta} \sum_{x \in D} \log (p_\theta(x)) \simeq \arg\max_{\theta} \mathbb{E}_{x \sim p_{\text{data}}(x)} \log(p_\theta(x)), \tag{2}
\]
where the expected value is computed over the empirical data distribution \( \hat{p}_{\text{data}}(x) \) The empirical data distribution is different from the true underlying data generating distribution \( p_{\text{data}}(x) \) to which we do not have direct access, and it is defined as the mixture of Dirac delta distributions \( \delta(x) \) that assigns probability mass \( 1/N_D \) to each data point in \( D \) as

\[
\hat{p}_{\text{data}}(x) = \frac{1}{N_D} \sum_{i=1}^{N_D} \delta(x - x^{(i)}). \tag{3}
\]

In practice, computing the integral over \( Z \) in Equation 1 is intractable. The optimization in Equation 2 can be simplified by maximizing a lower bound on the marginal distribution. Two typical methods to do so are the Variational Autoencoder and the Adversarial autoencoder, which are briefly discussed in Section 2.2 and Section 2.3, respectively.

2.2 Variational Autoencoder. Kingma and Welling [12] present an algorithm that allows estimation of the latent variable model parameters via optimization of the Evidence Lower BOund (ELBO). The algorithm, known as Variational Autoencoder (VAE), requires an inference model that approximates the (also) intractable true posterior distribution \( p_{\theta}(z|x) \) through a family of conditional probability distributions of the latent variables on observed data points \( q_\phi(z|x) \), with parameters \( \phi \) shared across data points. The ELBO can be formulated as

\[
\log(p_\theta(x)) \geq \mathbb{E}_{z \sim q_\phi(z|x)}[\log(p_\theta(x|z))] - D_{KL}(q_\phi(z|x) \| p(z)) := \text{ELBO}(\theta, \phi, x). \tag{4}
\]

where the second term is the Kullback - Leibler (KL) divergence, denoted \( D_{KL}(\|\|) \). Essentially, the KL divergence quantifies how close two distributions are. For further details about the ELBO and how to compute the KL-divergence see Appendix A.1 and A.2.

In the VAE framework, the prior is the isotropic Gaussian \( p(z) = \mathcal{N}(z; 0, I) \), where \( I \) is the identity matrix. The likelihood conditional distribution \( p_{\theta}(x|z) \) is represented as a multivariate Gaussian probability distribution with identity covariance matrix \( p_{\theta}(x|z) = \mathcal{N}(x; f_\theta(z), I) \), where the function \( f_\theta(z) : Z \rightarrow \mathbb{R}^M \) is parametrized with an ANN referred to as the probabilistic decoder. With this formulation, \( p_{\theta}(x) \) is an infinite mixture of Gaussian distributions. In the same way as with the probabilistic decoder, it is possible to parametrize the inference model conditional distribution using a neural network that performs a mapping \( q_\phi(z|x) : X \rightarrow \mathbb{R}^Z \) and outputs the mean \( \mu \) and standard deviation \( \sigma \) of the Gaussian distribution \( q_\phi(z|x) = \mathcal{N}(z; \mu, \text{diag} \sigma^2) \).

Thus, the ELBO balances two terms: the first term encourages the probabilistic decoder to produce samples that resemble the observed data, while the second term forces the approximated posterior distribution obtained from the inference model to be close to the prior distribution. Using the negative ELBO, the optimization objective can be reformulated as a minimization problem:

\[
\theta^*, \phi^* = \arg \min_{\theta, \phi} \mathbb{E}_{x \sim \hat{p}_{\text{data}}(x)} \left[ - \mathbb{E}_{z \sim q_\phi(z|x)}[\log(p_\theta(x|z)) \right] + \beta D_{KL}(q_\phi(z|x) \| p(z)) \right]. \tag{5}
\]

where \( \beta \) is a hyperparameter that weighs the reconstruction and regularization terms [14]. First order stochastic methods such as Stochastic Gradient Descent (SGD) can be applied to minimize Equation 5. Details on the VAE algorithm and how to derive the ELBO and how to estimate its gradients can be found in [12][13].

2.3 Adversarial Autoencoder. Makhzani et al. [15] propose an alternative formulation to the ELBO, where the KL divergence is replaced by an adversarial loss that forces the aggregated posterior distribution \( q_\phi(z) \) to be close to the prior

\[
q_\phi(z) = \int_X q_\phi(z|x) \hat{p}_{\text{data}}(x) dx \simeq p(z). \tag{6}
\]

In the original paper, Makhzani et al. explore the use of both probabilistic encoders (e.g. a Gaussian distribution as in VAEs) and deterministic encoders with \( g_\phi(x) \) as a deterministic mapping. For such deterministic encoders, the aggregated posterior is computed as

\[
q_\phi(z) = \int_X \delta(z - g_\phi(x)) \hat{p}_{\text{data}}(x) dx, \tag{7}
\]
where the only source of stochasticity in $q_\phi(z)$ comes from the data. Both the probabilistic and deterministic version of the AAE achieve similar performance if the hyper-parameters of the neural network are carefully tuned.

The adversarial loss is based on the GANs, introduced by Goodfellow et al. [16]. Let the generator network be the encoder network $g_\phi(x)$ with parameters $\phi$ in a deterministic encoder setup that performs a mapping $g_\phi(x) : X \rightarrow \mathcal{Z}$, and the decoder be a deterministic mapping $f_\theta(z) : \mathcal{Z} \rightarrow \mathbb{R}^M$ that outputs the mean of the Gaussian distribution $p_\theta(x|z) = \mathcal{N}(x; f_\theta(z), I)$. A discriminator model is introduced, modeled also with an ANN with mapping function $d_\xi(z) : \mathcal{Z} \rightarrow [0,1]$ that outputs a single scalar representing the probability that $z$ is a sample from the prior distribution $p(z) = \mathcal{N}(z; 0, I)$ (true samples), rather than being a latent space mapping from the generator (fake samples).

This translates into a min-max optimization, where first the discriminator is trained to correctly distinguish between real and generator samples by maximizing the probability of its correct predictions, i.e. maximizing the probability of classifying real samples $x_r$ as real ($d_\xi(x_r) = 1$) and fake samples $x_f$ as false ($d_\xi(x_f) = 0$). Second, the generator is trained to minimize the probability $1 - d_\xi(x_f)$ that the discriminator identifies its samples $x_f$ as fake samples, where $d_\xi(x_f) = 1$ means that the discriminator classifies a fake sample as a true sample.

The min-max optimization can be formulated as

$$\min_{\phi} \max_\xi \mathbb{E}_{Z \sim p(z)}[\log(d_\xi(z))] + \mathbb{E}_{X \sim \hat{p}_{data}(x)}[\log(1 - d_\xi(g_\phi(x)))]$$

which can be solved iteratively with SGD in two phases, by first updating the parameters $\xi$ of the discriminator (so that true and fake samples and distinguished), and then updating the parameters $\phi$ so that the generator becomes better at its task. The AAE algorithm is based on training the probabilistic decoder $p_\theta(x|z)$, the inference model $q_\phi(z|x)$ and the discriminator $d_\xi(z_f)$ in 2 stages: the reconstruction phase encouraging the decoder to produce realistic samples by using the encoding produced by the inference model, and the regularization phase updating the parameters of the generator and discriminator. Mescheder et al. [17] present a theoretical analysis on the correctness and limitations of using the adversarial loss as a latent space regularizer, as well as the relationship between the AAE objective and the ELBO.

Including the reconstruction loss in Equation 8, the final AAE optimization is defined as

$$\min_{\theta, \phi} \left( \mathbb{E}_{X \sim \hat{p}_{data}(x)}[-\mathbb{E}_{X \sim q_\phi(z|x)}[\log(p_\theta(x|z)]] + \right.$$

$$\left.\max_\xi \left( \mathbb{E}_{Z \sim p(z)}[\log(d_\xi(z))] + \mathbb{E}_{X \sim \hat{p}_{data}(x)}[\log(1 - d_\xi(g_\phi(x)))] \right) \right)$$.  

\section{Methods and materials}

The presented VAE and AAE are used in order to obtain probabilistic models of respiratory motion on a patient-specific and population level that capture the variability in breathing of a specific patient or a population of patients, respectively. The breathing signals are physical time series representing the movement of the chest from lung cancer patients. The models are based on one-dimensional convolutions that exploit the order and local structure of the signals. For the patient-specific models, we use the standard AAE and VAE algorithms, while for the population models we present a modified AAE architecture that allows training a population breathing model capable of simultaneously classifying and generating specific types of breathing.

\subsection{Patient and population data}

Different breathing signals were obtained with the stereotactic radiosurgery system Cyberknife® (Accuray Inc., Sunnyvale CA, US). Cyberknife® tracks breathing movement using correspondence of markers positioned on the patient’s chest [18]. The data used in our study consists of long respiratory traces for 21 different patients, some of them corresponding to the same patient on different days. The optical device tracks data with a 26 Hz frequency, for a total duration between ten and thirty minutes. The breathing signals for 15 out of the 21 patients were obtained from the open-access database containing data recorded at Georgetown University Hospital (Washington D.C, United States) [19]. The respiratory traces from the remaining 6 patients come from signals recorded
3.1.1 Breathing signal samples. Abnormal behavior corresponding to errors in the acquisition of the signals (usually related to machine recalibration) are manually removed and disregarded as outliers. All breathing signals consisting of 3 coordinates are projected onto the main axis of movement (eigenvector with highest eigenvalue) and divided into different periods \( \tau_j \) (time between start of different inhales). We find that the projection onto the principal axis retains around 95% of the variance, and allows obtaining three-dimensional signals by projecting back each of the generated signals into the original axes. Each period \( j \) is discretized into 4 points with an \( A_{s,j} \) coordinate denoting position and a \( \Delta_{s,j} \) coordinate that represents the difference in time between consecutive points. Thus, a period is parametrized by the vector

\[
\tau_j = (A_{EE,j}, \Delta_{EE,j}, A_{MI,j}, \Delta_{EI,j}, A_{ME,j}),
\]

where \( s \) denotes the stage within each breathing period: EE for the end of exhale (or beginning of inhale), EI for the end of inhale (or beginning of exhale), and ME, MI for the 2 intermediate points between EE and EI. For simplicity, we omit the redundant \( \Delta_{ME,j} \) and \( \Delta_{MI,j} \) time coordinates, since they are equal to \( \Delta_{EI,j}/2 \) and \( \Delta_{EE,j}/2 \), respectively.

Figure 1a displays how a fragment of the time series is discretized into periods, and how each period is subdivided into time-position points. A breathing sample is obtained by concatenating consecutive periods for the desired length of the signal. Each sample is assumed to be i.i.d. from a random breathing signal and is characterized by a vector \( \mathbf{x} = (\tau_1, \tau_2, ..., \tau_{\mathcal{N}_T}) \in \mathbb{R}^{\mathcal{N}_T \times 6} \) formed by \( \mathcal{N}_T \) discretized periods. Period values range between 4 and 6 seconds, and this variability determines the number of periods \( \mathcal{N}_T \) needed to form a sample that represents a breathing signal of a a given length. We use vectors of length \( \mathcal{N}_T = 25 \) to model shorter signals of 1 to 2 minutes, and \( \mathcal{N}_T = 100 \) for longer signals of several minutes corresponding to the typical duration of proton therapy treatments.

3.2 Signal reconstruction & sampling. The output vectors from the deep generative models \( \hat{\mathbf{x}} \) have the same structure as the discretized input vector and must be transformed back into a time series in order to obtain full breathing signals. The transformation is effectively implemented by reconstructing the path between two consecutive position points. An easy-to-implement first order approximation is a simple linear interpolation between the four position points in each cycle, which requires little time but compromises accuracy. The points in the interpolated time series are spaced using the same frequency as in the original time series (26 Hz, or 38.46 ms).
Instead, we reconstruct a realistic breathing time series with the help of an additional model, denoted reconstruction ANN. The input is the linearly interpolated time series, and the ANN learns a general function that deforms the linear time series into realistic shapes. Thus, the input for the reconstruction ANN is no longer a vector of dimension $M = 6 \times N_T$, but a sequence of 120 position values from the linearly interpolated time series (see Figure 1b). The number 120 is a hyperparameter that is selected from a set of different candidate lengths. Thus, the interpolated breathing signal, with position values equally spaced in time, is divided in vectors of 120 elements that are sequentially transformed into realistic shapes. The output of the ANN is the first 100 transformed values of the input series. By adding 20 extra positions the network achieves better results and discontinuities that arise when concatenating consecutive fragments in order to obtain the final time series are avoided. Further description of the ANN architecture is included in Appendix A.6.

The training data for the reconstruction networks consists of slices with 120 elements of position values obtained from slicing the recorded breathing signals, and the corresponding linear interpolations that are input to the ANN. During training, the input linear and output slices are normalized to the interval $[0,1]$. We investigate whether training the reconstruction ANN using only a subset of the data (either data from a single patient or a subset of data from all the patients) suffices in order to provide precise reconstructions, which would make the time series reconstruction highly scalable by needing one single trained ANN for an entire population. For this, we train the reconstruction ANN using (1) data from one patient (PatBR model) and (2) a subset of data from the GUH data (PopBR model), while both models are tested using the EMC dataset.

### 3.3 Patient-specific models

The standard VAE and AAE algorithms can be applied to obtain latent variable models that generate patient-specific breathing signals. We train both models per patient using an isotropic Gaussian distribution as the prior $p(z) = \mathcal{N}(z;0,I)$. During training, 80% of the patient data containing all the breathing signals corresponding to a particular patient forms the training set, while the remaining 20% is equally split into the validation set and test set. AAE based models are trained in two phases as described in Section 2.3. For the VAE, we make use of the reparametrization trick as described in [12]. The parameter $\beta$ in Equation 5 is normalized with respect to the input dimension $M$ and latent dimension $N$ (which varies per model) as $\beta_n = (M/N) \cdot \beta$. We find that a $\beta_n$ value of 0.02 achieves good balance between reconstruction and regularization of the latent space. Details about the architecture of the different models in the VAE and AAE are shown in Appendix A.3 and Appendix A.4, respectively. As shown in Equation 23 in Appendix A.2, the reconstruction term is equivalent to the squared error (SE), calculated as

$$SE = \|x - \hat{x}\|_2^2 = \|x - f_\theta(z)\|_2^2. \quad (11)$$

For a fixed encoder and decoder architecture, we investigate the effect of varying the dimensionality of the latent space on the reconstruction error and the generalization to the test set, and evaluate both of them numerically for a subset of the GUH patients. After training the models, the input vector $x$ can be reconstructed by passing it to the inference model to obtain a $z$ vector, which is subsequently processed by the decoder in order to get the mean of the distribution $p(x|z)$. Due to the stochasticity in the mapping from input space to latent space in the VAE encoder, different VAE decoder reconstructions of the same data point slightly differ from each other.

### 3.4 Generative-classification models

One of the advantages of AAEs is that the standard architecture can be slightly modified in order to leverage the information of few labeled data points so that the models can be trained in a semi-supervised manner to perform a classification task on top of the generative task. In the original paper [15], different modifications of the standard AAE architecture are presented and evaluated for semi-supervised clustering and classification tasks, and it is shown that the model achieves good performance when using only a small fraction of labeled examples from the dataset. The most notable difference is the introduction of an extra latent variable $y \in \mathbb{R}^C$ which represents the class to which the input belongs over $C$ classes and is practically implemented as a sparse one-hot vector with a 1 entry at the position corresponding to the class. Figure 2 shows the modified encoder with the latent variables $y$ and $z$ that are output by the classification head and the style head, respectively. In the case of breathing, this $y$ variable could indicate the presence of irregularities or the patient to which
breathing pertains. The encoder therefore outputs the joint distribution \( q_\phi(y, z|x) \) that factorizes as
\[
q_\phi(y, z|x) = q_\phi^y(y|x)q_\phi^z(z|x),
\]
where \( q_\phi^y(y|x) \) is a categorical distribution with class probabilities \( \pi(x) : X \rightarrow \mathbb{R}^C \), or a deterministic mapping. Since the softmax non-linearity and the use of one-hot vectors as a target forces sparsity in \( \pi(x) \), and to facilitate calculations, we use the deterministic mapping \( y = \pi(x) \). The approximate posterior \( q_\phi(z|x) \) is either a distribution or a deterministic mapping, as in the standard AAE. Given that the final goal is to simultaneously classify and generate new samples given a specific input, we introduce a small modification to the semi-supervised AAE architecture that uses a single discriminator for both the classification and style heads, instead of two independent discriminators that separately regularize the distributions over \( z \) and \( y \) as in [15]. In this way, the aggregated approximated posterior is forced to match the \( C \)-dimensional mixture prior distribution as
\[
p(y) = \text{Cat}(y; c), \quad p(z) = \mathcal{N}(z; 0, I),
\]
where \( \alpha \) is a hyperparameter that controls the weight of the classification loss. We find that \( \alpha \) values of around 5-10 significantly enhance classification when the number of labels is limited, while higher values do not improve and even hinder performance. Practically, the term in Equation 14 is added to Equation 9 for optimization. Training the semi-supervised models requires careful hyperparameter tuning and model design. In particular, we find Batch Normalization [21] in the encoder and decoder and an unequal learning rate between the discriminator and the decoder to be crucial. Further details about the architecture of the models can be found in Appendix A.5.

3.4.3 Modeling breathing irregularities. We investigate the use of the semi-supervised framework to build generative models of signals that present irregularities, and in particular baseline shifts that represent gradual downwards or upwards shifts of the breathing signal relative to the baseline. First, two different experiments are performed in which the models are trained using an analytical dataset that contains simplified sinusoidal breathing signals. In S1, only the slope of the signals with constant amplitude and period is varied, and in S2 the period and amplitude are varied as well. The goal of these experiments is to showcase the ability of the model to represent the data in three low-dimensional Gaussian manifolds according to whether there is an upwards or downwards shift, or no shift at all (regular signals).

Second, we evaluate the performance of such models using the real breathing signals from the GUH and EMC datasets, and investigate the number of labeled samples needed to obtain accurate classification. Since the goal is to model the downwards or upwards trend of the signals, all the samples of each patient
Figure 2: Summary of the breathing modeling workflow. First, the original time series is preprocessed and projected using principal component analysis (PCA) into the main axis of movement (eigenvector with biggest eigenvalue), from which the input vectors $\mathbf{x}$ are obtained. Patient or population models are then obtained through the use of the (a) VAE, (b) AAE and (c) SAAE with one-dimensional convolutional encoder and decoder models. In the VAE and AAE, the encoder (or inference) model produces a low-dimensional latent variable $\mathbf{z}$ that ideally captures the factors of variation in the dataset, such as variations in period, amplitude and exhale position. In the SAAE the inference model generates a class label latent variable $\mathbf{y}$ besides vector $\mathbf{z}$. Labeled data can be leveraged during training in order to learn the classification task in a semi-supervised manner. During generation (red dashed square), the sampled latent variables are transformed into the input vector form. These new vectors $\hat{\mathbf{x}}$ are then transformed into a time series with the help of an auxiliary reconstruction neural network.
are rescaled to the interval [0,1] by dividing by the maximum among all patient’s samples (the minimum is always 0). For each data sample, a crude label is assigned according to the slope of the line fitted to its corresponding breathing time series: if the slope of a sample is above a certain threshold value, the breathing sample is labeled as upwards baseline shift. Likewise, if the (negative) slope is below the threshold, the data point is labeled as downwards baseline shift. We determine the threshold value to correspond to around the 7.5 upper and lower percentile of the distributions of slopes in the GUH dataset. All models are trained using the GUH dataset as the training set (with 10% as validation data) and tested on the EMC dataset.

3.5 Convolutional filters. We use one-dimensional convolutional layers for both the encoder and decoder models under the assumption that these provide the encoder and decoder with powerful feature extractors and reconstructors that exploit the order in time and local structure of the periods. A one-dimensional discrete kernel convolution operation (denoted as \(x \ast K\)) over an input \(x \in \mathbb{R}^{N_T \times 6}\) with \(N_T\) timesteps and 6 channels for the different time and position values, using a kernel \(K \in \mathbb{R}^{K \times 6}\), consists of sliding the kernel matrix through the different \(j\) timesteps and computing the product

\[
(x \ast K)(j) = \sum_{k=1}^{K} \sum_{h=1}^{6} K_{k,h} x_{j-k,h}
\]

(15)

To verify and quantify the advantages of using convolutional layers, we train encoder and decoder models purely based on fully-connected layers and compare them to the one-dimensional convolutional models in terms of performance in reconstructing and generating samples.

3.6 Sampling artificial signals. A graphical summary of the methodology is shown in Figure 2. New breathing signals can be sampled from the breathing models by independently sampling the latent parameters from their prior distributions. For the VAE and AAE patient models, the latent vectors \(z\) are sampled from the isotropic Gaussian distribution \(N(z; 0, I)\). For the SAAE, a recorded signal can be classified by the encoder to obtain a class \(y\), and new samples from that class can be generated by sampling the isotropic multivariate Gaussian for \(z\) and fixing the class vector. In principle, the SAAE can also be employed to randomly generate data from a whole population by also sampling \(y\) from the categorical distribution \(\text{Cat}(y, c)\), with class probabilities \(c\).

4 Results

4.1 Patient-specific models. Figure 3 shows how the different models perform in reconstructing the original input through a 5-dimensional latent space (\(N = 5\)). The comparison is shown in a time series form (after the linear interpolation) for a randomly selected sample from the training set, and the test samples corresponding to the worse SE (test samples with worst reconstruction).

For fixed encoder and decoder architectures, the middle and right plots in Figure 4 showcase the effect of varying the dimensionality of the latent space on the reconstruction performance. For two patient-specific VAE and AAE breathing models, the SE of training and test data are compared for a one-dimensional convolutional architecture (CVAE and CAAE) and an architecture purely based on fully connected layers (FVAE and FAAE). The loss values are rescaled to the interval [0,1] to facilitate comparison, with 1 being the maximum loss achieved at weight initialization.

The evolution of the loss during training is shown in the left plot in Figure 4, for the convolutional architectures and 2 patients. For the CVAE, we show the values of the ELBO (which adds the KL-divergence term on top of the reconstruction SE) for the training and validation set, while for the CAAE we display the evolution of the reconstruction loss. The loss is also rescaled between [0,1], and the effect of the dropout layers is not applied at evaluation time.

Appendix B.1 shows how data is arranged in the low-dimensional latent space. In particular, the distribution of the latent variables per latent dimension is displayed in Figure 9c for the AAE and Figure 9d for the VAE. Figure 9 includes grid samples from the 2-dimensional latent space for the AAE and VAE-based models trained with \(N = 2\), in order to observe patterns in the reduced space and how similar and different signals are organized and grouped apart or together.
Figure 3: (Top row) Reconstruction of breathing signals from AAE-based models and (bottom row) VAE-based models for (left) a sample from the training set, (middle) the worst performing sample in vector form, and (right) a fragment of the worst reconstruction with highest error. The discretized time series is passed to the encoder in order to obtain the latent variables $z$ from which the decoder reconstructs the input. The reconstructed signals are then linearly interpolated and transformed back into a time series form with the help of the reconstruction network.

Figure 4: (Left) Training and validation loss curves of the VAE and AAE for two different random patients. (Middle and right) Effect of the dimensionality of the latent space on the loss, for a fixed encoder and decoder and two different random patients. The reconstruction loss of the training and test data is displayed for the optimized convolutional (CVAE and CAAE) and feedforward (FVAE and FAAE) encoder and decoder architectures. Training is stopped when the validation loss starts to increase.
Figure 5: Performance of the population SAAE classification head on the test GUH data, when using 300 or 600 labeled samples per patient (around 10% of the dataset) and all the available labeled samples during the supervised training step. The remaining samples are used to train the models in an unsupervised manner. The abscissa displays the label assigned by the encoder, while the legend shows the color code for the true labels. Each of the bars shows the true label of the samples assigned to a certain class by the encoder.

4.2 Semi-supervised population model. Using the GUH dataset, the modified SAAE framework is trained to obtain a population model that classifies and assigns a breathing sample to the most similar patient and generates similar breathing samples from that patient. We evaluate the effect of the number of labeled examples, and the approximate number of samples needed to obtain good classification accuracy. Figure 5 shows the classification performance using 300 and 600 labeled data points per class when training the classification head (corresponding to approximately 10% of the dataset), as well as the performance when all the labeled samples are used during training in a supervised manner. The dimensionality of the latent space and the classification head is set to 15 ($C = 15, N = 15$). Additionally, Figure 10b in Appendix B.2 displays random samples from each of the 15 classes, showing different breathing signals randomly sampled for each of the 15 classes in the model.

4.3 Semi-supervised baseline shift models. The classification accuracy of the SAAE encoder for sinusoidals with constant period and amplitude and varying slope (S1), and sinusoidals with varying amplitude, period and slopes (S2) is shown at the top row of Figure 6. All the models are trained using a 15-dimensional latent space ($N = 15$) and 3 different classes corresponding to regular breathing, and upwards or downwards baseline shifts. These bar plots illustrate the added difficulty from S1 to S2, and the effect it has on the number of labeled samples needed in order to achieve 100% accuracy. The bottom row of Figure 6 shows the results from training the semi-supervised framework with real breathing signals. We find that including more labeled samples in the supervised classification stage does not increase the classification accuracy to 100%, unlike in the sinusoidal case. As expected, most of the misclassified samples correspond to samples whose slope is close to the threshold slope used during crude labeling. Signals sampled from the model for each type of baseline shifts can be found in Figure 10a in Appendix B.2.

4.4 Signal reconstruction. Two different models that transform the generated model samples into a time series form are evaluated: a patient-based model that is trained using the data of only a single patient from the dataset (PatBR), and a population model that uses a subset of samples from a population of patients that capture the different shapes of breathing signals (PopBR). The PatBR is trained using a single patient from the GUH dataset, while the PopBR uses 10% of the GUH dataset as training data, instead of all the available samples. This is due to the fact that the training data for this reconstruction task consists of fragments of the breathing time series (vectors with 120 position values) obtained from linear interpolation of the generated vectors, resulting in a few million training examples.

The middle and right plots in Figure 7 allows visual comparison of reconstructed breathing signals for the worst test sample (in the GUH) for both models. Different PatBR models are trained using the data from the patient with the largest and lowest period in the dataset, as well as one of the patients with an average period. From these PatBR models, the former (largest period) achieves the largest error (on signals of the patient with the lowest period) and its reconstruction is the ones shown in the middle...
Figure 6: Classification performance of the SAAE encoder for the artificial sinusoidal test data (top row), and GUH training and EMC test data (bottom row). The abscissa displays the label assigned by the encoder, and the legend shows the color code for the true labels. Thus, each bar indicates the true label of samples that are assigned to a certain class by the encoder. The models are trained to generate data from three clusters representing regular breathing, downwards baseline shifts and upwards baseline shifts. Depending on the model, a different number of labeled data points is used to train the classification head with supervision, while the remaining data is used in an unsupervised training phase.

Figure 7: (Left) Average absolute error (average L1-norm \(|w_{\text{real}} - w_{\text{rec}}|\) of time series vectors \(w\)) achieved by the PatBR and PopBR models in the reconstruction of breathing signals for the training patient, the worst-performing patient and the entire set of patients present in each of the GUH and EMC datasets. For the PatBR models, the average error over the set of patients does not include the data from the training patient. (Middle) Worst reconstruction of the fragment of an EMC signal for the PatBR model trained with data from the patient with maximum amplitude. (Right) Worst-performing reconstruction over all the EMC dataset for the PopBR model.
5 Discussion

5.1 Accuracy and effect of convolutions. All the standard VAE, standard AAE and SAAE architectures result in breathing models capable of capturing the variability of breathing in a patient or population through the use of few latent variables, as opposed to other approaches that use implicit adversarial models [8] [9]. The breathing models are easy to sample and the decoders generate realistic breathing samples, as seen in Figure 3 and Figure 9. The effect of including convolutional filters results in significant improvement in generalization, with a three-fold reduction of the reconstruction error of the test set compared to models based on fully-connected architectures. The main difference between the VAE and AAE resides in the latent space and how well they capture the training data: even though the generalization performance is very good and close for both models, the AAE results in aggregated posterior distributions for the latent space that are closer to the prior and reconstructs slightly better the training examples (see Figure 9 in Appendix B.1).

5.2 Time series reconstruction accuracy. Regarding the accuracy in the reconstruction of the time series, the population based reconstruction ANN PopBR consistently outperforms the single patient PatBR models and opens the door to using a single model to reconstruct breathing signals for any patient. PatBR models fail to reconstruct signals from other patients, especially when they are evaluated on breathing signals whose period significantly differs from that of the samples used for training, as seen in the left plot of Figure 7 where the maximum errors for the patients with maximum and minimum breathing period correspond to the signals from each other. In general, the generalization error of the PopBR model in reconstructing signals is very low and it provides accurate reconstructions for patients whose breathing signal was recorded in a different location and machine. The error could in principle be further decreased by training a specific PatBr for each specific patient, based on Figure 7, at the expense of slightly longer computation time.

5.3 Semi-supervised models. The fact that a single model captures many different classes of breathing and can selectively sample is a novelty with respect to previous architectures that specialize in one of such classes [5] [9]. With respect to the baseline shift model, the significantly larger number of samples needed in the supervised phase (2500 instead of 500) demonstrates the added complexity of real signals with respect to artificial sinusoidal dataset that does not capture period and amplitude variations within signals. Even though the accuracy does not increase to 100%, sampling each of the classes always results in breathing samples that are coherent with the appropriate regular, upwards and downwards trends. We believe that the main reason why the classification error does not decrease further resides in the way that samples were labeled, causing signals whose slope is in the limit of the decision boundary, as well as signals that present both upwards and downwards trends, to be misclassified. This is confirmed by the fact that breathing samples with larger slopes were always correctly assigned to the corresponding baseline shift.

The number of samples per class needed to train a population model is considerably lower than for the baseline model, and the classification error is greatly reduced when using 600 samples per patient. This type of population model is suitable for a population with a reduced number of patients, and modeling a large population would ideally require unsupervised clustering of similar breathing patterns. In principle, the proposed model could be applied for such unsupervised classification-generative task, but more research is needed in this direction.

5.4 Latent space structure & dimension. One of the main possible sources of error in the breathing models is the mismatch between the aggregated posterior and the prior distribution from which the samples are generated, as well as the lack of support over regions in the latent space \( Z \). Regarding the former, comparing Figure 9d and Figure 9c in Appendix B.1 it can be concluded that the latent
variables are distributed in a way that resembles the prior distribution, even though the AAE results in closer agreement. On the other hand, an AAE framework with a deterministic encoder may result in latent spaces that present gaps or regions without support when the dimensionality of the latent space is too large. This mainly depends on the size of the dataset and the fact that this is the only source of stochasticity in the encodings (as opposed to the VAE), indicating that high dimensional latent spaces may hinder performance in deterministic AAE frameworks. For latent spaces with larger number of dimensions, the encoder in the VAE provides Gaussian distributions with larger standard deviation, thus covering the totality of the space.

For the set of all possible models, the reconstruction performance is in theory independent of the latent dimension. For deterministic autoencoders, on could imagine that very powerful models - deep encoders and decoders - would perfectly reconstruct the input using as few as one latent dimension, but this does not happen in practice. The same reasoning can be applied to the AAE with a deterministic encoder, as well as for VAE-based models as long as the stochasticity in the latent encodings is taken into account. Thus, very powerful VAE architectures would in principle achieve good reconstruction regardless of the latent dimension: the contribution of the KL-divergence to the ELBO scales linearly with the latent dimensionality, so an increase in the ELBO caused by an increase of the latent space dimensionality can in theory be compensated by increasing the variance of the approximated posterior \( q_\phi(z|x) \) (lower KL-divergence per latent dimension). In practice, architectures with very powerful decoders tend to ignore the information encoded in \( z \) [23] [24], and adding more dimensions to the latent space may have a more immediate effect on performance than increasing the capacity of the model (adding more layers or neurons), in concordance with Figure 4.

Finally, assuming that the data lays in a low-dimensional manifold, there is a certain dimensionality for the latent space beyond which adding more dimensions will add little information or even be counterproductive due to the stochasticity in the encodings. Another reason behind the limited number of latent variables is related to collapse of the approximated posterior, where some units remain inactive during training and equal to the prior distribution [27]. For the specific case of breathing and given the presented encoder and decoder convolutional architectures, this limit seems to be around 10 latent variables.

5.5 Usefulness of breathing models. The models presented in this paper can be applied to a wide range of tasks involving generation and classification. First, the patient-specific models can be leveraged to capture the variability in breathing of a patient and consequently generate patient samples. Some applications are the simulation of breathing during a treatment in order to design robust treatments beforehand (such as obtaining proton therapy treatments robust against breathing movement that result in desired clinical outcomes), or the estimation of the likelihood of presenting a certain type of breathing.

The semi-supervised AAE framework can in principle be applied to dataset augmentation or breathing classification. The former is related to artificially generating samples when the available data for a patient is scarce. One example of such application is a new recorded breathing signal from a patient using the population model and generating samples from the corresponding class. Another example is breathing classification referring to determining the type of breathing irregularities when we are presented with a new breathing signal. In such case, it is possible to generate additional samples that present the identified irregularity. An application related to classification is the diagnostic of breathing abnormalities, or even underlying physiological conditions. One of the advantages of training the proposed framework in a semi-supervised way is the possibility to build such models requiring only the labels of a subset of the data, which could be assigned manually.

5.6 Applicability to other biomedical signals. The models presented in this paper can in principle be applied to any other kind of biomedical data that shows a repetitive or periodic structure, much like a breathing signal is composed of well-defined periods despite the fact that the length of these periods and their amplitude and location varies. The added advantage of our generative approach with respect to other models in the literature that do not explicitly model the density \( p_\theta(x) \) such as [5], [6] or [7] is the possibility to map the data samples to specific regions (or classes) in the latent space.

To our knowledge, generative models can be constructed for: i) bioelectrical signals generated by muscle cells and nerves such as electroglottograph (EGG), electrocardiogram (ECG) or electroencephalogram (EEG), ii) biomagnetic signals related to a specific physiological activity that are usually associated with
an accompanying electric field such as magnetoencephalography (MEG) or magnetocardiography (MCG), or iii) other biomechanical signals that describe mechanical functions of biological systems as motion, dynamics or blood pressure (BP) signals.

5.7 Limitations. One of the main limitations when training patient-specific models is the size of the dataset. Deep learning methods require a significant amount of different examples to achieve good generalization. The GUH dataset is formed by long breathing signals (in some cases multiple signals per patient) from which between 1200 and 5000 samples can be obtained for each patient. This is not generally the case for the data recorded in clinics on a regular basis, usually consisting of short breathing signals of few minutes, as is the case for the majority of the EMC dataset. This highlights the need for population models in the specific case of breathing.

Another drawback is the uninformative prior \( p(y) \) in the semi-supervised model, which is needed in order to ensure that the encoder uses all the available classes and that the \( y \) vector resembles a one-hot vector. Using such uninformative prior in cases where there is a considerable imbalance between classes may cause misclassification of some samples in order to match the uniform probability distribution over classes. While this can be solved by assigning a higher weight to the reconstruction term during training, it may sacrifice smoothness in the latent space. Therefore other categorical priors containing the estimated proportion of the different classes in the population could yield better performance.

5.8 Computational cost. An important advantage of the presented methodology is the fact that it achieves feasible compute times. We reduce training times by using Graphics Processing Units (GPUs), which are needed to train the presented convolutional architectures due to the requirements of the latest version of the Tensorflow package [25]. We perform most of the training using an NVIDIA\textsuperscript{®} Tesla\textsuperscript{®} K80, and the training times vary around 10 minutes for the patient-specific models, 30 minutes for the reconstruction PopBR and PatBR models, and 20 minutes for the semi-supervised AAE models, while generation is almost instantaneous.

6 Conclusion

We present a methodology to develop patient-specific breathing models using a discretized breathing signals based on VAE and AAE architectures. The models compress the data into a reduced dimension with few independent and uncorrelated parameters with known probability distributions that represent correlated movements in breathing, and sampling such distributions results in realistic signals that capture baseline shifts and oscillations in period and amplitude. We show that latent spaces with dimensionality of around 10 are able to capture most of the variation of the data and achieve reconstruction of the original samples.

The sampled output vector of the generative models can be transformed into a time series form with the help of a reconstruction model. We demonstrate that a reconstruction model trained with the data of a single patient (PatBR) does not achieve good generalization when evaluated on other patients, and it is outperformed by a population model (PopBR) trained with a subset of the data of a population of patients. This population model is trained only once and can be applied to new patients with great accuracy.

Finally, we introduce a modified framework based on the AAE that allows to build models that perform simultaneous classification and generation of different types of breathing. We show that the models can be trained end-to-end in a semi-supervised manner by including the labels of only a subset of the data, which varies around 10% of the dataset depending on the task. The main advantage of the proposed architecture is the fact that we can separate the different classes in the data into different low-dimensional manifolds that can be sampled from known probability distributions. Even though we base our study on mechanical breathing signals, the framework shows potential applicability to simulation and diagnostic purposes using any other biomedical signal with a quasi-periodic structure.
7 Acknowledgements

This work is supported by KWF Kanker Bestrijding, and part of the KWF research project PAREL with number 11711. The authors are also thankful to Dr. Dennis Schaart for useful discussing regarding the manuscript, as well as Dr. Mischa Hoogeman and Dr. Steven Habraken for the insightful discussions and providing one of the datasets.

8 Code availability

The code used to train the models in the AAE, VAE and SAAE, as well as to train the PopBR and PatBR reconstruction networks, is available at: https://github.com/opaserr/breathr.

References

[1] Neal, M. L., & Kerckhoffs, R. (2010). Current progress in patient-specific modeling. Briefings in bioinformatics, 11(1), 111-126.

[2] McSharry, P. E., Clifford, G. D., Tarassenko, L., & Smith, L. A. A dynamical model for generating synthetic electrocardiogram signals. IEEE Transactions on Biomedical Engineering, vol. 50, no. 3, pp. 289-294, March 2003. https://doi.org/10.1109/TBME.2003.808805.

[3] George, R., Vedam, S. S., Chung, T. D., Ramakrishnan, V., & Keall, P. J. (2005). The application of the sinusoidal model to lung cancer patient respiratory motion. Medical physics, 32(9), 2850–2861. https://doi.org/10.1118/1.2001220.

[4] Yildirim, O., San Tun, R., & Acharya, U. R. (2018). An efficient compression of ECG signals using deep convolutional autoencoders. Cognitive Systems Research, 52, 198-211. https://doi.org/10.1016/j.cosy.2018.07.004.

[5] Zhu, F., Ye, F., Fu, Y., Liu, Q., & Shen, B. (2019). Electrocardiogram generation with a bidirectional LSTM-CNN generative adversarial network. Scientific reports, 9(1), 1-11.

[6] Delaney, A. M., Brophy, E., & Ward, T. E. (2019). Synthesis of Realistic ECG using Generative Adversarial Networks. arXiv preprint arXiv:1909.09150.

[7] Hernandez-Matamoros, A., Fujita, H., & Perez-Meana, H. (2020). A novel approach to create synthetic biomedical signals using BiRNN. Information Sciences. https://doi.org/10.1016/j.ins.2020.06.019.

[8] Golany, T., & Radinsky, K. (2019, July). PGANs: Personalized generative adversarial networks for ECG synthesis to improve patient-specific deep ECG classification. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 33, pp. 557-564) https://doi.org/10.1609/aaai.v33i01.3301557.

[9] Wulan, N., Wang, W., Sun, P., Wang, K., Xia, Y., & Zhang, H. (2020). Generating Electrocardiogram Signals by Deep Learning. Neurocomputing. https://doi.org/10.1016/j.neucom.2020.04.076.

[10] Xiao, C., Choi, E., & Sun, J. (2018). Opportunities and challenges in developing deep learning models using electronic health records data: a systematic review. Journal of the American Medical Informatics Association, 25(10), 1419-1428. https://doi.org/10.1093/jamia/ocy068.

[11] Hong, S., Zhou, Y., Shang, J., Xiao, C., & Sun, J. (2020). Opportunities and challenges of deep learning methods for electrocardiogram data: A systematic review. Computers in Biology and Medicine, 103801.

[12] Kingma, D. P., & Welling, M. (2013). Auto-encoding variational bayes. arXiv preprint arXiv:1312.6114.
[13] Kingma, D. P., & Welling, M. (2019). An introduction to variational autoencoders. arXiv preprint arXiv:1906.02691.

[14] Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., & Lerchner, A. (2016). beta-vae: Learning basic visual concepts with a constrained variational framework.

[15] Makhzani, A., Shlens, J., Jaitly, N., Goodfellow, I., & Frey, B. (2015). Adversarial autoencoders. arXiv preprint arXiv:1511.05644.

[16] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. & Bengio, Y. (2014). Generative adversarial nets. In Advances in neural information processing systems (pp. 2672-2680).

[17] Mescheder, L., Nowozin, S., & Geiger, A. (2017). Adversarial variational bayes: Unifying variational autoencoders and generative adversarial networks. arXiv preprint arXiv:1701.04722.

[18] Coste-Manière, E., Olender, D., Kilby, W., & Schulz, R. A. (2005). Robotic whole body stereotactic radiosurgery: clinical advantages of the Cyberknife integrated system. The international journal of medical robotics + computer assisted surgery : MRCAS, 1(2), 28–39. https://doi.org/10.1002/rcs.39

[19] Ernst, F. (2011). Compensating for quasi-periodic motion in robotic radiosurgery. Springer Science & Business Media. https://doi.org/10.1007/978-1-4614-1912-9

[20] Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

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

[22] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958. http://jmlr.org/papers/v15/srivastava14a.html

[23] Zhao, S., Song, J., & Ermon, S. (2017). Infovae: Information maximizing variational autoencoders. arXiv preprint arXiv:1706.02262.

[24] Chen, X., Kingma, D. P., Salimans, T., Duan, Y., Dhariwal, P., Schulman, J., Sutskever, I., & Abbeel, P. (2016). Variational lossy autoencoder. arXiv preprint arXiv:1611.02731.

[25] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M. & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In 12th USENIX symposium on operating systems design and implementation (OSDI 16) (pp. 265-283).

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

[27] Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., & Winther, O. (2016). Ladder variational autoencoders. In Advances in neural information processing systems (pp. 3738-3746).
A Implementation details

A.1 Deriving the ELBO. Even though there are different ways to obtain the ELBO, the most common derivation is based on Jensen’s inequality. For a concave function such as the natural logarithm the Jensen inequality states that
\[ \log E[x] \geq E[\log x]. \]

Starting from the marginal likelihood of the probabilistic model, the expression of the ELBO can be obtained as

\[ \log (p_\theta(x)) = \log \int_z p_\theta(x, z) dz \]
\[ = \log \int_z p_\theta(x, z) \frac{q_\phi(z|x)}{q_\phi(z|x)} dz \]
\[ = \log E_{z \sim q_\phi(z|x)} \left[ \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \]
\[ \geq E_{z \sim q_\phi(z|x)} \left[ \log \left( \frac{p_\theta(x, z)}{q_\phi(z|x)} \right) \right] \]
\[ = E_{z \sim q_\phi(z|x)} \left[ \log \frac{p_\theta(x,z)}{q_\phi(z|x)} \right] - D_{KL}(q_\phi(z|x)||p(z)). \]

A.2 Dissecting the ELBO. The output of the probabilistic decoder is the likelihood conditional distribution \( p_\theta(x|z) \). This distribution is represented as a multivariate Gaussian probability distribution with identity covariance matrix \( p_\theta(x|z) = \mathcal{N}(x; f_\theta(z), I) \), where the function \( f_\theta(z) : \mathcal{Z} \to \mathbb{R}^M \) is parametrized with an ANN and represents the mean. The log-likelihood is formulated as

\[ \log(p_\theta(x|z)) = \log \left( \frac{1}{\sqrt{(2\pi)^M |I|}} \exp \left( -\frac{1}{2} (x - f_\theta(z))^T I^{-1} (x - f_\theta(z)) \right) \right) = C - \frac{1}{2} \| x - f_\theta(z) \|^2, \]

where \( C \) is a constant. The result has the same form as the SE, which is computed for the model output \( \hat{x} \), true output \( x \) as

\[ SE = \| x - \hat{x} \|^2. \]

Thus, minimizing the log-likelihood with respect to the parameters \( \theta \) (which is done by approximating the expectation \( E_{z \sim q_\phi(z|x)} \log(p_\theta(x|z)) \) by taking Monte Carlo samples) yields the same result as minimizing the SE. On the other hand, the KL-divergence is defined as

\[ D_{KL}(p(x)||q(x)) = \int p(x) \log \left( \frac{p(x)}{q(x)} \right) dx = E_{x \sim p(x)} \log \left( \frac{p(x)}{q(x)} \right). \]

When \( p \) and \( q \) are both Gaussian distributions, the KL-divergence can be computed in closed form. In this case the prior is \( p(z) = \mathcal{N}(z; 0, I) \) and the encoder distribution is \( q_\phi(z|x) = \mathcal{N}(z; \mu, \text{diag} \sigma^2) \). For an N-dimensional latent space, the KL-divergence can be analytically computed as:

\[ D_{KL}(q_\phi(z|x)||p(z)) = \frac{1}{2} \left( - \sum_i N \log \sigma_i^2 + 1 + \sum_i \sigma_i^2 + \sum_i \mu_i^2 \right) \]

A.3 VAE architecture. The architecture of the VAE models is shown in Figure 8. We find that using BatchNormalization [21] and Dropout [22] between layers significantly improves convergence and results in significantly better generalization. The encoder contains one-dimensional maximum pooling layers and the decoder uses dilation rates bigger than 1, which seem to positively affect reconstruction performance.
Regarding $\beta_n$, a value of 0.02 yields optimum balance between a compact Gaussian latent space and good reconstruction performance, with lower values slightly favoring more accurate reconstructions but latent spaces with larger standard deviations that do not match the prior. We use a batch size of 256 samples and the Adam optimizer for training [20], with original learning rate $10^{-4}$ decreased to $5 \cdot 10^{-5}$.

### A.4 Standard AAE architecture.

The architecture of the different models composing the AAE is shown in Figure 8. For this framework, the order of the Batch Normalization and activation layers greatly affects convergence and stability during training, with Batch Normalization placed in between the activation and Dropout yielding the best results. Using Leaky ReLU activation functions with slope 0.1 in the discriminator also seems to help to stabilize training, in concordance with [26]. The models are trained using a batch size of 256 samples and the Adam optimizer with unequal learning rates: $10^{-4}$ in the reconstruction phase and $2 \cdot 10^{-4}$ for the discriminator. Due to the fact that the SE is approximately 4 times lower than the cross-entropy loss used for the discriminator, the SE is equalized by multiplying it by 4 during training.

### A.5 Semi-supervised and unsupervised AAE architecture.

Figure 8 shows the architecture of the encoder, decoder and discriminator models for the semi-supervised modified AAE architecture. We find that Batch Normalization between layers in the encoder and decoder significantly boosts performance and helps stabilize training, as well as using unequal learning rates for the Adam optimizer: $10^{-4}$ in the reconstruction and supervised classification phase and $2 \cdot 10^{-4}$ for the discriminator. The models are trained using a batch size of 256 samples. As with the standard AAE architecture, the cross-entropy loss is approximately 4 times higher than the SE, and so the latter is equalized during training.

### A.6 Reconstruction network.

The architectures for the PatBr and PopBR models are identical and are shown in Figure 8. The learning rate is set to $10^{-4}$ with a decay rate of $10^{-6}$ per epoch, whereas the batch size is set to 256 samples per batch.

### B Additional results

#### B.1 Latent space of patient-specific models.

To observe the structure of the latent space produced by the inference models, we train the AAE and VAE frameworks using a 2-dimensional latent space. Figure 9a and Figure 9b show the a grid of samples from such latent space for each of the models trained using the data from a random patient. The square grid is defined as 25 equally spaced points between $[-1.5, 1.5]$ in each of the two axis. According to the imposed Gaussian distribution on the latent space, the samples in the center are more likely to be observed than the ones at the corners. Different parts of the latent space cluster signals with similar traits, such as the same type of irregularities or similar amplitudes and exhale positions.

To visualize any possible mismatch between the Gaussian prior and the aggregated posterior in the latent space, we plot the distribution of the $z$ latent variables for all points in the dataset (the approximated aggregated posterior distribution). Figure 9c and Figure 9d show the distribution over the latent space for the trained 5-dimensional AAE and VAE, respectively. Both Figures show aggregated posterior distributions that closely resemble the imposed Gaussian prior, while the AAE shows closer agreement.

#### B.2 Sampling the semi-supervised models.

The models trained using the modified semi-supervised AAE framework can be sampled to generate breathing signals that present a certain type of irregularity or resemble breathing from a certain patient. First, a class $y$ is selected, and then the Gaussian submanifold representing breathing of that particular class is sampled according to the prior distribution $p(z)$. Figure 10a displays samples for each of the three classes in the baseline shift model trained using 2500 labeled samples per class, while Figure 10b shows samples from each patient in the population based model that is trained using 600 labeled examples per class.
Figure 8: Architecture of the different networks used in the AAE and VAE algorithms. (a) Convolutional encoder architecture with 4 one-dimensional convolutional layers and 2 fully-connected layers. A 1-D max-pooling layer follows each convolution, and Batch Normalization and Dropout with probability 0.1 are applied after each pooling layer. (b) Convolutional decoder architecture, with 2 fully-connected layers followed by 4 up-sampling dilated one-dimensional convolutional layers. Batch Normalization and Dropout with probability 0.3 follow each of the convolutions. (c) Color code for the layers used in the different models. (d) Discriminator architecture for the AAE, containing 4 fully-connected hidden layers followed by a sigmoid unit. (e) Discriminator for the SAAE. (f) Reconstruction network transforming the interpolated time series into realistic shapes.
Figure 9: (a, b) Grid samples from a 2-dimensional latent space for the generative models based on the AAE and VAE architectures. The grid consists of 25 equally spaced points covering the interval [-1.5, 1.5] for both axis. 
(c, d) Distribution over the 5-dimensional latent space of the latent variables corresponding to all the data points for a random patient.
(a) Random samples from the baseline shift model. Rows 1, 2 and 3 visibly show signals with regular, downward and upwards baseline shifts.

(b) Random samples from the population model. Each of the rows represents a patient from the dataset.

Figure 10: Randomly generated signals for each class $y$ in the baseline shifts and the population model. Each row represents (a) a type of baseline shift — regular (C1), downwards baseline shifts (C2) and upwards baseline shifts (C3) — or (b) a type of breathing in a population of patients. For each class, different $z$ values are independently sampled from the isotropic Gaussian distribution $\mathcal{N}(z; 0, I)$. Each row contains data from different fractions for the same patient, where amplitudes can sometimes notably differ (P8) and periods are usually similar for each of the patients (P11).