Reichman University
Efi Arazi School of Computer Science
M.Sc. program - Research Track

Variational Information Bottleneck with Gaussian Process Priors for Semi-Supervised Time Series Classification

Itamar Efrati

M.Sc. dissertation, submitted in partial fulfillment of the requirements
for the M.Sc. degree, research track, School of Computer Science
Reichman University

May 2025

This work was carried out under the supervision of Dr. Shai Fine, head of the Data Science Institute, Reichman University.

Abstract

Time series classification problems are prevalent across various domains, often characterized by intra-series relationships within features, and inter-series relationships between the same features over time. Developing a generalized model capable of capturing these intricate properties poses a considerable challenge. When focusing on real world data, not only that gathering quality information might be difficult, but in many cases also labeling the data might be expencive and require data expertise for labaling the data which might be very expensive. In many cases this results in a very large dataset with limited number of labeled data wich is a common senario for semi supervised learning.

In this thesis, we introduce an innovative approach termed Gaussian Process Variational Information Bottleneck (GP-VIB). This model is designed as an end-to-end system, with a primary focus on acquiring a concise representation of the initial sequence. It aims to retain essential information vital for accurate classification. This stands in contrast to the GP-VAE model, which learns a latent space primarily geared towards information preservation for reconstruction, and necessitates a two-step process for classification. Through our experiments, we illustrate that the proposed GP-VIB model outperforms existing methods on renowned benchmark datasets like the multivariate Time Series Classification Archive (UEA). Moreover, we will show that our proposed model is also capable to handle irregularly sampled time series (ISTS) data, including real-world medical datasets.

In addition, we develop a semi-supervised version of GP-VIB which is able to use the unlabeled data to improve the performance of the classification task of the labeled data. We show that the semi supervised GP-VIB outperform its supervised version. In addition we show that our propose method outperform current results over real-world dataset for predicting the health states of patients with chronic pain based on cellphone usage data collected by the Data Science Institution.

Contents

Introduction

Time series data is characterized by inter-series relations between inferred features and intra-series relations within every such feature. The challenge of finding and understanding these features and their relations over time is motivated by applications in finance, healthcare, signal processing and other fields, where understanding and categorizing temporal patterns is crucial for Time Series Classification (TSC).

Common approaches to TSC involve leveraging diverse techniques such as distance-based methods, feature engineering-based methods, and hybrid methods. More advanced methods include the use of deep learning models, recurrent neural networks (RNNs), and convolutional neural networks (CNNs) tailored to handle the sequential nature of time series data.

An alternative approach for classification places a central focus on constructing a meaningful representation by transforming the original data into a distinct space conducive to the classification process. In classical machine learning, Principal Component Analysis (PCA) employs a linear transformation for this purpose, while more contemporary approaches employ non-linear transformations, such as deep Auto-Encoders (AE), and later on, Variational Auto-Encoders (VAE) [60]. Unlike the deterministic nature of the AE, VAE are stochastic models that learn the distribution of the latent space, which, by capturing latent space characteristics allow them to demonstrate increased robustness and superior performance, essential for reconstructing input data and generating new samples.

As for time series data, VAE, with it's mean field assumption, may not be well-suited for time series data due to the challenge of capturing temporal dependencies effectively within the assumed factorized latent variable distributions. A related model, more suitable for time series data, is the Gaussian Process Variational Auto-Encoder (GP-VAE) [32], which, instead of the mean field assumption, utilizes deep variational inference with a Gaussian process prior, enabling the acquisition of a structural variational posterior that adeptly captures intricate temporal dynamics in time series. Originally introduced as a method for imputing missing values in time series data, the GP-VAE model can also be applied to model complete data distributions. Leveraging the approximated posterior, latent representations can be generated and subsequently employed as input for a classifier.

However, questions arise regarding the information retained in the GP-VAE's latent space, what type of data might be excluded, and how to ensure that the compressed data version preserves relevant information for the classification task. It is crucial to

ascertain that the model does not omit crucial information for classification and to address potential disparities between data relevant for reconstruction and that pertinent to classification. Thorough evaluation and consideration of these aspects are essential to guarantee the efficacy of the compressed data in supporting the classification task.

A major challenge in time series classification is the limited availability of labeled data, as annotating time series can be costly and time consuming. To address this, **semi-supervised learning (SSL)** methods aim to leverage both labeled and unlabeled data to improve classification performance. Traditional SSL techniques include self-training and graph-based methods, while modern deep learning approaches employ consistency regularization and pseudo-labeling to extract useful representations from unlabeled instances. In the context of time series data, SSL has been explored through contrastive learning and temporal consistency constraints, offering promising directions for enhancing classification performance in low-data regimes.

We introduce the **G**aussian **P**rocess **V**ariational **I**nformation **B**ottleneck (GP-VIB), a novel deep learning probabilistic model tailored for TSC. By seamlessly integrating the Variational Information Bottleneck (VIB) method [1] with a GP prior, our model excels in acquiring a discriminative latent space that effectively captures the essential temporal dynamics crucial for classification tasks. To enhance our model's capacity for feature extraction and parameter learning, we incorporate InceptionTime blocks [53] into the encoder architecture. This inclusion enables our GP-VIB model to efficiently process time series data, discern intricate temporal patterns, and derive comprehensive representations. Unlike the GP-VAE model, which entails two distinct steps for classification, our proposed GP-VIB model offers an end-to-end solution, streamlining the classification process for improved efficiency and performance. Furthermore, our model is on par with state-of-the-art (SOTA) models on benchmark datasets for TSC.

Inspired by research at the Data Science Institute in collaboration with Sheba Hospital, where patient cellular usage data was used to predict health state levels of chronic pain patients, we developed a semi-supervised version of GP-VIB. This variant leverages both labeled and unlabeled data to enhance classification performance, outperforming its supervised counterpart in various data availability scenarios while demonstrating comparable performance on the real-world chronic pain patients dataset.

The rest of the work is organized as follows. We begin in Chapter 2 by presenting essential background information and related work on the methods employed in our study. Next in Chapter 3, we introduce the GP-VIB model in greater detail. In Chapter 4 we describe a series of experiments designed to evaluate the utility of GP-VIB, including comparisons with GP-VAE on the Healing MNIST dataset [62] and assessments against SOTA models on the UEA archive [3] for multivariate time series classification. Additionally in Chapter 5, we extend our model to the semi-supervised setting and and in Chapter 6 we present a series of experiments we conducted to assess its effectiveness on Healing MNIST and a real-world dataset for predicting the health states of patients with chronic pain based on cellphone usage data. Finally in Chapter 7, we summarize our findings and discuss potential directions for future research.

Background

2.1 Essentials

In this thesis, we introduce our novel model, SS-GP-VIB, which combines variational information bottleneck with Variational Auto-Encoder, and Gaussian Process priors, for semi-supervised time series classification. In the subsequent sections, we will dive into each component individually.

2.1.1 Notation and Definitions

Observed Data and Likelihood

The set of all observed data points is denoted as $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, where n is the total number of observations and x_i is a single observe sample for $i \in \{1, 2, \ldots, n\}$.

The **likelihood of observing x**, with θ represents the model parameters, is given by:

$$p_\theta(\mathbf{x}) = p_\theta(x_1, x_2, \ldots, x_n)$$

Latent Variables and Bayes' Rule

Let z represent a latent variable associated with a single observation. The set of latent variables corresponding to all observations is denoted as $\mathbf{z} = \{z_1, z_2, \ldots, z_n\}$, where each z_i corresponds to the latent variable associated with the observation x_i. Their relationship can be modeled by **Bayes' rule** as follows:

$$\underbrace{p_\theta(z \mid x)}_{posterior} = \frac{\overbrace{p_\theta(x \mid z)}^{\text{likelihood}} \overbrace{p_\theta(z)}^{prior}}{\underbrace{p_\theta(x)}_{\text{evidence}}}$$

where:

- $p_\theta(x \mid z)$ is the **likelihood of the observed data** x**, given the latent variables** z.

- $p_\theta(z)$ is the **prior** distribution of the latent variables, which represents our beliefs about the latent variables before observing the data.
- $p_\theta(x)$ is the **evidence** of the observed data. The evidence, also known as the marginal likelihood, represents the probability of observing x. It is termed "marginal likelihood" because it is obtained by marginalizing (integrating) over the latent variable z. The marginal likelihood is given by the equation:

$$p_\theta(x) = \int p_\theta(x \mid z) p_\theta(z)\, dz$$

- $p_\theta(z \mid x)$ is the **posterior** distribution of the latent variables z, given the observed data x. It represents the updated belief about the latent variables after observing the data.

Log-Likelihood

For a single observed sample x and its corresponding latent variable z, the **joint likelihood** is the product of the likelihood of observing x given z and the prior distribution over z:

$$p_\theta(x,z) = p_\theta(x \mid z) p_\theta(z)$$

Taking the logarithm of this joint likelihood gives the **log-likelihood**:

$$\log p_\theta(x,z) = \log \left(p_\theta(x \mid z) p_\theta(z) \right)$$

Using the logarithmic property $\log(ab) = \log(a) + \log(b)$, we can express the log-likelihood as:

$$\log p_\theta(x,z) = \log p_\theta(x \mid z) + \log p_\theta(z)$$

Thus, for a single sample, the log-likelihood is the sum of the log of the likelihood of observing x given z, and the log of the prior distribution over z.

Kullback-Leibler Divergence

The Kullback-Leibler (KL) divergence, also known as the relative entropy, is a measure of how one probability distribution diverges from a second, expected probability distribution. It is defined as:

$$\mathrm{KL}(p \parallel q) = \mathbb{E}_p \left[\log \frac{p(x)}{q(x)} \right],$$

The KL divergence is always non-negative by definition. To demonstrate this, we proceed with the following derivation.

First, consider the expression for the KL divergence:

$$-KL(p \parallel q) = -\mathbb{E}_p \left[\log \frac{p(x)}{q(x)} \right] = \mathbb{E}_p \left[\log \frac{q(x)}{p(x)} \right].$$

Next, we apply Jensen's inequality [1]

$$\mathbb{E}_p \left[\log \frac{q(x)}{p(x)} \right] \leq \log \mathbb{E}_p \left[\frac{q(x)}{p(x)} \right].$$

Since $p(x)$ is a probability distribution and $\mathbb{E}_p[1] = 1$, we get:

$$\log \mathbb{E}_p \left[\frac{q(x)}{p(x)} \right] = \log 1 = 0.$$

Thus, we have:

$$-D(p \parallel q) \leq 0 \rightarrow D(p \parallel q) \geq 0.$$

Therefore, the KL divergence $\mathrm{KL}(p \parallel q)$ is always non-negative.

2.1.2 Foundations of Variational Inference

In the realm of probabilistic modeling, estimating parameters and inferring latent structures are pivotal tasks. This section lays the groundwork for understanding Variational Inference (VI) and its application in Variational Autoencoders (VAEs). We begin by discussing Maximum Likelihood Estimation (MLE), a fundamental method for parameter estimation. However, the presence of latent variables often renders direct likelihood maximization intractable. To address this, we introduce the Evidence Lower Bound (ELBO), which provides a tractable surrogate for the log-evidence. This motivates the transition to VI, a powerful framework for approximating intractable posteriors, ultimately leading to the development of VAEs.

Maximum Likelihood Estimation

In statistics MLE is a method for estimating the parameters θ that maximize the likelihood function of the observe data $\mathbf{x}$. If the likelihood function is differentiable, the parameters that maximize the likelihood can be found by taking the derivative of this function and solving for its maximum.

Let's consider the following example. Let p be a normal distribution and $\theta = (\mu, \sigma^2)$ are it's parameters, where μ is the mean and σ^2 is the variance. Given data $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, we would like to find the parameters θ that maximize the likelihood of the observe data $\mathbf{x}$. By assuming that the data points are independent and identically

[1] Jensen's inequality states that for a concave function f, $f(\mathbb{E}_p[y(x)]) \geq \mathbb{E}_p[f(y(x))]$

distributed (i.i.d), the likelihood function can be written as the product of the individual likelihoods for each data point:

$$p_\theta(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i),$$

where $p_\theta(x_i)$ is the probability density function (PDF) of the normal distribution:

$$p_\theta(x_i) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(x_i-\mu)^2}{2\sigma^2}\right).$$

The log-likelihood function, which simplifies computation and improves numerical stability, is given by:

$$\log p_\theta(\mathbf{x}) = \sum_{i=1}^{n} \log p_\theta(x_i).$$

Substituting the PDF into the log-likelihood:

$$\log p_\theta(\mathbf{x}) = -\frac{n}{2}\log(2\pi\sigma^2) - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2.$$

To find the MLE of the parameters, we compute the derivatives of the log-likelihood with respect to μ and σ^2.

The derivative of the log-likelihood with respect to μ is:

$$\frac{\partial}{\partial\mu}\log p_\theta(\mathbf{x}) = \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i-\mu).$$

Setting this derivative equal to zero and solving for μ:

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i.$$

Thus, the MLE for μ is the sample mean. The derivative of the log-likelihood with respect to σ^2 is:

$$\frac{\partial}{\partial\sigma^2}\log p_\theta(\mathbf{x}) = -\frac{n}{2\sigma^2} + \frac{1}{2\sigma^4}\sum_{i=1}^{n}(x_i-\mu)^2.$$

Setting this derivative equal to zero and solving for σ^2:

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}(x_i-\mu)^2.$$

Thus, the MLE for σ^2 is the sample variance.

When latent variables are present in a model, the standard MLE framework, which maximizes the log-likelihood $\log p_\theta(\mathbf{x})$, needs to be extended. In the case of latent variables $\mathbf{z}$, we are interested in the **marginal log-likelihood** $\log p_\theta(\mathbf{x})$ (the evidence), which is the log of the marginal probability of the observed data obtained by integrating out the latent variables, i.e., $p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{x}, \mathbf{z})\, dz$. The evidence, instead of the log-likelihood of the observed data alone, becomes the target for parameter estimation.

Evidence Lower Bound

Maximizing the evidence is not always straightforward, particularly when the data is generated from a more complex distribution. To address this issue, we can derive a lower bound to the evidence. The derivation follows these steps:

$$\log p(x) = \mathbb{E}_q\left[\log p(x)\right] \tag{2.1}$$

$$= \mathbb{E}_q\left[\log \frac{p(x,z)}{p(z|x)}\right] \tag{2.2}$$

$$= \mathbb{E}_q\left[\log \frac{p(x,z)}{p(z|x)}\frac{q(z)}{q(z)}\right] \tag{2.3}$$

$$= \mathbb{E}_q\left[\log \frac{p(x,z)}{q(z)}\frac{q(z)}{p(z|x)}\right] \tag{2.4}$$

$$= \underbrace{\mathbb{E}_q\left[\log \frac{p(x,z)}{q(z)}\right]}_{\text{ELBO}} + \underbrace{\mathbb{E}_q\left[\log \frac{q(z)}{p(z|x)}\right]}_{\text{KL}} \tag{2.5}$$

$$\mathrm{KL}(q(z) \parallel p(z \mid x)) \geq 0 \rightarrow \log p(x) \geq \mathrm{ELBO} \tag{2.6}$$

where q is some distribution over the latent variables z. Once we reach this point, there are two potential directions we can take:

1. If the posterior is tractable or approximate we can set $q(z) = p(z \mid x)$, which results in the following:

$$\mathrm{KL} = \mathbb{E}_{p(z|x)}\left[\log \frac{p(z \mid x)}{p(z \mid x)}\right] = 0 \rightarrow \log p(x) = \mathbb{E}_{p(z|x)}\left[\log \frac{p(x,z)}{p_\theta(z \mid x)}\right] = \mathrm{ELBO} \tag{2.7}$$

   This leads to the **Expectation-Maximization** (EM) algorithm, where:

   - **E-step**, we compute the expectation of the marginal posterior for each value in the latent space, given the observed data and fixed model parameters.

$$\forall i\; q_\theta(z_i) = p_\theta(z_i|x_i)$$

   - **M-step**, We maximize the expected complete log-likelihood of the data, using the marginal posterior we computed in E-step

$$\theta^* = \arg\max_\theta \mathbb{E}_{p_\theta(z|x)}[\log p(x,z)]$$

2. If the posterior distribution $p(z|x)$ is complex or intractable, we can use **Variational Inference**

Variational Inference

VI is a method for approximating intractable integrals in Bayesian inference, especially posterior $p(z \mid x)$, by introducing a parameterized family of distributions $q_\theta(z)$ and an optimization objective that provides a tractable approximation. This objective is typically based on a bound to the intractable integral and a divergence measure chosen to guide the optimization process. The most common divergence is the KL divergence:

$$\mathrm{KL}(q_\theta(z) \| p(z \mid x)) = \mathbb{E}_{q_\theta(z)} \left[\log q_\theta(z) - \log p(z \mid x) \right].$$

Since the posterior $p(z \mid x)$ is generally unknown, this KL divergence cannot be directly computed. A common bound used to tackle this problem is the ELBO, given by:

$$\begin{aligned} \mathscr{L}(\theta) &= \mathbb{E}_{q_\theta(z)}[\log p(x,z) - \log q_\theta(z)] && (2.8) \\ &= \mathbb{E}_{q_\theta(z)}[\log p(x|z) + \log p(z) - \log q_\theta(z)] && (2.9) \\ &= \mathbb{E}_{q_\theta(z)}[\log p(x|z)] - \mathrm{KL}(q_\theta(z)) \parallel p(z)) && (2.10) \end{aligned}$$

To ensure tractability, the variational distribution $q_\theta(z)$ is often chosen from a family of distributions that allow efficient sampling and optimization. A common assumption is the mean-field approximation, where $q_\theta(z)$ is factorized as:

$$q_\theta(z) = \prod_i q_{\theta_i}(z_i).$$

This assumption simplifies computation by removing dependencies between latent variables but may limit the expressiveness of the variational approximation.

A classical approach for optimizing the ELBO is coordinate ascent variational inference (CAVI) [7], where each latent variable distribution is updated iteratively while keeping the others fixed. Although CAVI is straightforward and often converges to a local optimum, it has significant limitations. It requires analytical updates, which are not always available, and it can be slow in high-dimensional settings due to its sequential nature.

Due to the limitations of coordinate ascent, more flexible optimization approaches, such as gradient-based methods, are desirable. However, optimizing the ELBO via gradient descent requires computing the gradient of an expectation:

$$\nabla_\theta \mathbb{E}_{q_\theta(z)}[\log p(x|z)]$$

which is intractable in most cases. The main difficulty arises because the expectation requires integrating over all possible values of z, and this integral is typically not available in closed form. When z is high-dimensional or when $q_\theta(z)$ has a complex dependency on θ, computing this integral exactly becomes impractical.

A natural approach is to approximate the expectation using Monte Carlo sampling. However, this introduces an additional challenge: we now need to differentiate a stochastic function, which is not straightforward. The dependency of the samples on θ complicates differentiation, and naïve approaches can lead to high variance estimates, making optimization inefficient.

These challenges highlight the need for specialized techniques to approximate the expectation while maintaining tractable gradients.

Variational Autoencoder

Variational Autoencoders (VAEs) [60] extend variational inference by leveraging deep learning models to approximate the posterior distribution and optimize the ELBO. The core idea is to learn a latent representation z that captures meaningful features of the data while allowing efficient sampling and generation.

The standard VAE framework introduces a probabilistic encoder-decoder structure. Given an input x, the encoder models the approximate posterior $q_\theta(z \mid x)$, which is typically assumed to be a Gaussian distribution:

$$q_\theta(z \mid x) = \mathcal{N}(z \mid \mu_\theta(x), \Sigma_\theta(x)),$$

where $\mu_\theta(x)$ and $\Sigma_\theta(x)$ are parameterized by a neural network. The decoder then reconstructs x by sampling from the latent space and modeling the likelihood $p_\phi(x \mid z)$, which is also parameterized by a neural network.

To train a VAE, we maximize the ELBO:

$$\mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\theta(z|x)} \left[\log p_\phi(x \mid z)\right] - \mathrm{KL}(q_\theta(z \mid x) \| p(z)).$$

The first term ensures that the latent representation preserves information relevant to reconstructing x, while the second term regularizes the latent distribution to be close to a chosen prior $p(z)$, often set as a standard Gaussian $\mathcal{N}(0, I)$.

A major challenge in training VAEs arises when computing the gradient of the expectation in the ELBO:

$$\nabla_\theta \mathbb{E}_{q_\theta(z|x)} \left[\log p_\phi(x \mid z)\right].$$

As discussed in the previous section, differentiating an expectation is generally intractable due to the dependence of $q_\theta(z \mid x)$ on θ. A naïve Monte Carlo approach would lead to high-variance gradient estimates, making training unstable.

To address this issue, VAEs employ the reparameterization trick [60], which allows gradients to be computed efficiently. Instead of directly sampling $z \sim q_\theta(z \mid x)$, we rewrite the sampling process using a deterministic transformation:

$$z = \mu_\theta(x) + \sigma_\theta(x) \odot \varepsilon, \quad \varepsilon \sim \mathcal{N}(0, I).$$

Here, $\odot$ represents element-wise multiplication. By expressing z as a differentiable function of θ, we can now propagate gradients through the sampling process, enabling efficient optimization via stochastic gradient descent.

In practice, VAEs are trained using deep neural networks for both the encoder and decoder. The encoder learns meaningful latent representations, while the decoder reconstructs data from the latent space. The model is typically optimized using the reparameterized ELBO with backpropagation.

By combining variational inference with deep learning, VAEs provide a scalable framework for unsupervised representation learning and generative modeling. The reparameterization trick plays a crucial role in enabling efficient gradient-based optimization, making VAEs a practical and widely used approach in deep generative modeling.

2.1.3 Gaussian Process

Fundamental Definitions

Multivariate Gaussian Distribution. A random vector $\mathbf{X} \in \mathbb{R}^n$ follows a multivariate normal distribution with mean vector $\mu \in \mathbb{R}^n$ and covariance matrix $\Sigma \in \mathbb{R}^{n\times n}$, denoted by

$$\mathbf{X} \sim \mathcal{N}(\mu, \Sigma),$$

if its probability density function (pdf) is given by

$$p(\mathbf{X}) = \frac{1}{(2\pi)^{n/2}|\Sigma|^{1/2}} \exp\left(-\frac{1}{2}(\mathbf{X}-\mu)^\top \Sigma^{-1}(\mathbf{X}-\mu)\right),$$

where $|\Sigma|$ denotes the determinant of Σ, and Σ is symmetric and positive semi-definite.

Marginal Distributions of a Multivariate Gaussian. Consider partitioning $\mathbf{X}$, μ, and Σ as follows:

$$\mathbf{X} = \begin{bmatrix} \mathbf{X}_a \\ \mathbf{X}_b \end{bmatrix}, \quad \mu = \begin{bmatrix} \mu_a \\ \mu_b \end{bmatrix}, \quad \Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}.$$

Then the marginal distributions are given by

$$\mathbf{X}_a \sim \mathcal{N}(\mu_a, \Sigma_{aa}), \quad \mathbf{X}_b \sim \mathcal{N}(\mu_b, \Sigma_{bb}).$$

Conditional Distribution of a Multivariate Gaussian. Given the partitioned random vector $\mathbf{X}$ and observing $\mathbf{X}_b = \mathbf{b}$, the conditional distribution of $\mathbf{X}_a$ is also Gaussian:

$$\mathbf{X}_a \mid \mathbf{X}_b = \mathbf{b} \sim \mathcal{N}(\mu_{a|b}, \Sigma_{a|b}),$$

where

$$\mu_{a|b} = \mu_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{b} - \mu_b),$$
$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba}.$$

Stochastic Process. A stochastic process is a collection of random variables $\{f(x) : x \in \mathscr{X}\}$ indexed by a set $\mathscr{X}$, where each x corresponds to an input (such as time, space, or another domain), and $f(x)$ represents a random variable.

Formally, a stochastic process defines a distribution over functions:

$$f : \mathscr{X} \to \mathbb{R}.$$

Gaussian Process (GP). A GP is a stochastic process where any finite collection of function values follows a multivariate Gaussian distribution. That is, a function $f(x)$ is said to follow a Gaussian Process if for any finite set of inputs $\mathbf{X} = [x_1, x_2, \ldots, x_n]^\top$, the corresponding outputs $\mathbf{f} = [f(x_1), f(x_2), \ldots, f(x_n)]^\top$ satisfy:

$$\mathbf{f} \sim \mathrm{N}\left(\mu(\mathbf{X}), K(\mathbf{X}, \mathbf{X})\right),$$

where:

- $\mu(\mathbf{X}) = [\mu(x_1), \mu(x_2), \ldots, \mu(x_n)]^\top$ is the mean vector, determined by the mean function $\mu : \mathscr{X} \to \mathbb{R}$,
- $K(\mathbf{X}, \mathbf{X})$ is the $n \times n$ covariance matrix, where each entry is given by a positive semi-definite kernel function $k : \mathscr{X} \times \mathscr{X} \to \mathbb{R}$:

$$[K(\mathbf{X}, \mathbf{X})]_{ij} = k(x_i, x_j).$$

We write this compactly as:

$$f(x) \sim GP\left(\mu(x), k(x, x')\right).$$

GP is fundamentally a Bayesian inference procedure. Given observed data, we update our prior belief about the underlying function to obtain a posterior distribution, which we can use to make predictions at new, unseen inputs.

Problem Setup

Suppose we observe a data set of input-output pairs n:

$$\mathscr{D} = \{(x_i, y_i)\}_{i=1}^n,$$

where $x_i \in \mathscr{X}$ and $y_i \in \mathbb{R}$. We model the outputs as noisy observations of an underlying function f:

$$y_i = f(x_i) + \varepsilon_i, \quad \varepsilon_i \sim \mathrm{N}(0, \sigma_n^2).$$

We denote:

- $\mathbf{X} = [x_1, x_2, \ldots, x_n]^\top$ — the matrix of observed inputs.
- $\mathbf{y} = [y_1, y_2, \ldots, y_n]^\top$ — the vector of observed outputs.

Prior

Before observing any data, our prior belief is that the function f follows a Gaussian process: $f(x) \sim GP(\mu(x), k(x,x'))$. This implies that the vector of function values at any finite set of inputs $\mathbf{X}$ is multivariate Gaussian:

$$\mathbf{f} = [f(x_1), \ldots, f(x_n)]^\top \sim \mathrm{N}(\mu(\mathbf{X}), K(\mathbf{X},\mathbf{X})).$$

Thus, the prior is fully determined by the mean function $\mu(x)$, and the covariance (kernel) function $k(x,x')$.

Posterior

The goal of inference is to compute the posterior distribution over the function values at new, unobserved test inputs $\mathbf{X}_* = [x_1^*, x_2^*, \ldots, x_m^*]^\top$, given the observed data $(\mathbf{X}, \mathbf{y})$.

Bayesian inference tells us that the posterior is the conditional distribution:

$$p(\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y}),$$

where $\mathbf{f}_* = [f(x_1^*), f(x_2^*), \ldots, f(x_m^*)]^\top$.

Joint Distribution

From the GP prior, the joint distribution of the observed outputs $\mathbf{y}$ and the function values at the test points $\mathbf{f}_*$ is Gaussian:

$$\begin{bmatrix} \mathbf{y} \\ \mathbf{f}_* \end{bmatrix} \sim \mathrm{N}\left(\begin{bmatrix} \mu(\mathbf{X}) \\ \mu(\mathbf{X}_*) \end{bmatrix}, \begin{bmatrix} K(\mathbf{X},\mathbf{X}) + \sigma_n^2 \mathbf{I} & K(\mathbf{X},\mathbf{X}_*) \\ K(\mathbf{X}_*,\mathbf{X}) & K(\mathbf{X}_*,\mathbf{X}_*) \end{bmatrix} \right).$$

Posterior Distribution

Using the formula for the conditional distribution of a multivariate Gaussian, the posterior over $\mathbf{f}_*$ given $\mathbf{y}, \mathbf{X}_*, \mathbf{X}$ is:

$$\mathbf{f}_* \mid \mathbf{X}_*, \mathbf{X}, \mathbf{y} \sim \mathrm{N}(\mu_*, \Sigma_*),$$

where:

$$\mu_* = \mu(\mathbf{X}_*) + K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1}(\mathbf{y} - \mu(\mathbf{X})),$$
$$\Sigma_* = K(\mathbf{X}_*,\mathbf{X}_*) - K(\mathbf{X}_*,\mathbf{X})[K(\mathbf{X},\mathbf{X}) + \sigma_n^2 \mathbf{I}]^{-1} K(\mathbf{X},\mathbf{X}_*).$$

The Role of the Kernel

The kernel function $k(x,x')$ is a critical component of the Gaussian Process prior. It defines the covariance between the function values at any two inputs x and x':

$$\mathrm{Cov}[f(x), f(x')] = k(x,x').$$

Through the kernel, we encode our prior beliefs about the properties of the function we aim to model, such as:

- **Smoothness:** How rapidly the function is expected to vary.
- **Periodicity:** Whether the function repeats over some interval.
- **Linearity:** If the function behaves approximately linearly.

The choice of kernel has a direct influence on both the prior and the posterior distributions. It determines the types of functions that are considered plausible before seeing any data and how the model generalizes after conditioning on observations.

Importantly, kernels can be combined to create more expressive models:

- **Addition of kernels:** $k(x,x') = k_1(x,x') + k_2(x,x')$ allows the model to capture the sum of different behaviors (e.g., smooth trend + periodicity).
- **Multiplication of kernels:** $k(x,x') = k_1(x,x') \cdot k_2(x,x')$ allows properties to interact multiplicatively (e.g., periodic structure with decaying amplitude).

Thus, the kernel is not only a mathematical object but also a crucial tool for incorporating domain knowledge into the model and shaping the space of functions we consider.

2.1.4 Gaussian Process Variational Autoencodr

Multivariate time series with missing values are quite common in real-life settings. Classical time series imputation methods often fail to capture the complex, non-linear interactions between different channels, typically treating each feature independently and ignoring shared temporal dynamics. Moreover, many standard time series models are not designed to handle arbitrary patterns of missing data, which are common in real-world datasets. GPs are a powerful tool for modeling temporal data due to their ability to provide smooth interpolations and calibrated uncertainty estimates. However, GPs suffer from poor scalability with respect to the number of observations, with a computational complexity of $\mathrm{O}(N^3)$, making them impractical for long or high-dimensional time series. Furthermore, GPs require carefully designed kernel functions to capture complex temporal patterns and cross-channel dependencies, which becomes increasingly challenging as the dimensionality of the data grows and the missingness patterns become more irregular.

To address these limitations, Fortuin et al. proposed a VAE with a GP prior and a structured variational posterior, termed GP-VAE [32]. This model can learn a latent representation of input time series with missing data that captures complex dynamics between different channels. In addition, the model can reconstruct an imputed version of the input time series. GP-VAE uses the structured inference algorithm [4] to learn a variational posterior for each feature in the latent space, that is,

$$q(z_{1:T,j}|x^o{}_{1:T}) = \mathcal{N}(m_j, \Lambda_j^{-1}),$$

where j is the index of the feature in the latent space and x^o is the input time series with only observed values. The prior of Z is represented as a GP such that

$$z(j) \sim \mathrm{GP}(m_z(\cdot), k_z(\cdot,\cdot)),$$

where k_z is the Cauchy kernel, which is appropriate for modeling data that varies at multiple time scales:

$$K_{Cau}(\tau, \tau') = \sigma^2 \left(1 + \frac{(\tau - \tau')^2}{l^2}\right)^{-1}. \tag{2.11}$$

In order to learn only from the observed data, they devised an ELBO such that its reconstruction error term sums only over the observed values, reminiscent of the HI-VAE method [82]. The model's ELBO follows:

$$\sum_{t=1}^{T} \mathbb{E}_{q(z_t|x_{1:T})} \left[\log p\left(x_t^o|z_t\right)\right] - \beta \cdot \mathrm{KL}\left[q\left(z_{1:T}|x_{1:T}\right)||p\left(z_{1:T}\right)\right]. \tag{2.12}$$

Structured Variational Inference

In their paper, Kingma and Welling [60] set the variational distribution to be a diagonal covariance (independent) multivariate Gaussian, i.e., a mean-field approximation. Moving to a structured variational inference, the variational distribution is represented by a full covariance multivariate Gaussian distribution. This results in a reparameterization gradient that involves a dense matrix multiplication that scales as $O(T^2)$, where T is the number of time steps. Bamler and Mandt [4] suggested an algorithm for the reparameterization gradient in $O(T)$. The main idea is to infer the parameters of the full covariance matrix through its precision matrix. Using the Cholesky decomposition, the precision matrix Λ can be written as the product of a bidiagonal matrix and its transpose:

$$\Lambda := B^T B, \quad \text{with} \quad B = \begin{cases} b_{tt'}, & \text{if } t' \in \{t, t+1\} \\ 0, & \text{otherwise.} \end{cases}$$

This parameterization ensures that Λ is positive definite, symmetric, and tridiagonal. While the precision matrix is sparse, its inverse (the covariance matrix) can be dense, enabling the use of a broader and richer set of variational distributions than those obtained under the mean-field approximation. Moreover, learning the B matrix requires estimating only $2T-1$ parameters, scaling as $O(T)$, similarly to the computational effort of the mean-field approach.

2.1.5 InceptionTime

While deep learning has led to major breakthroughs in fields like natural language processing and computer vision, progress in TSC remained limited. Inception [101] is a convolutional neural network architecture originally designed for image classification, where its main innovation was the use of parallel convolutions with multiple kernel sizes within the same block, allowing the network to capture features at different spatial scales. This architecture was later improved with the addition of residual connections in Inception-v4 [102], further boosting its performance. Recognizing the potential of this design, InceptionTime [53] adapted the Inception architecture for time series data by replacing the two-dimensional convolutions with one-dimensional convolutions, increasing the kernel sizes to better fit time series patterns, and introducing bottleneck layers within each Inception block to reduce the input dimensionality and control the model's complexity.

Inception Module The core building block of InceptionTime is the *Inception module*, which applies several parallel operations to the same input time series:

- A bottleneck layer with m filters of length 1 to reduce the input dimensionality.
- Several parallel one-dimensional convolutions with different kernel lengths $l \in \{10, 20, 40\}$ applied to the bottleneck output.
- A MaxPooling operation with a bottleneck applied afterward.
- The outputs of all these parallel operations are concatenated along the feature dimension.

This structure allows the model to capture patterns at multiple temporal scales while keeping the number of parameters manageable.

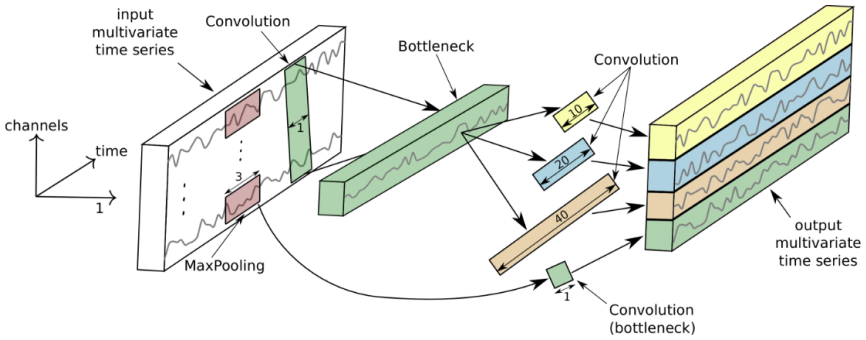


Figure 2.1: Single InceptionTime block. Source [53].

Single InceptionTime Model A *single InceptionTime model* is a deep convolutional neural network consisting of:

- A stack of d residual blocks (typically $d = 3$).
- Each residual block contains three consecutive Inception modules.
- A residual connection that adds the block's input to its output to stabilize training.
- A Global Average Pooling (GAP) layer that aggregates the feature maps over the time dimension.
- A fully connected softmax layer that outputs class probabilities for C classes.

Formally, for an input time series x_i, the output of the single model can be expressed as $\sigma_c(x_i, \theta)$ where σ_c is the softmax output for class c, and θ are the learned parameters of the model.

Ensemble InceptionTime leverages an ensemble of $n = 5$ identical Inception-based models, each initialized with different random weights. This approach reduces the variance of predictions caused by random initialization and the stochastic nature of training. The final prediction $\hat{y}_{i,c}$ for instance x_i and class c is the average of the softmax outputs from each model:

$$\hat{y}_{i,c} = \frac{1}{n} \sum_{j=1}^{n} \sigma_c(x_i, \theta_j), \quad \forall c \in \{1, \ldots, C\}$$

where θ_j are the parameters of the j-th model in the ensemble.

2.1.6 Deep Variational Information Bottleneck

The Information Bottleneck (IB) method is an Information Theoretic technique, introduced by Tishby et al. [106]. The main idea behind the IB method is to find a compressed representation of the input data that maximizes the mutual information with the relevant output (task) while minimizing the mutual information with the irrelevant parts of the input. More formally, given a random variable X representing the input data and a random variable Y representing the output, the IB optimization problem can be expressed as follows.

$$\max I(Z,Y) \,\text{s.t.}\, I(X,Z) \leq I_c \tag{2.13}$$

where I_c is an information constraint. Equivalently, by introducing a Lagrange multiplier β, the IB objective function can be stated as follows:

$$R_{IB} = I(Z,Y) - \beta \cdot I(X,Z) \tag{2.14}$$

where:

$$I(Z,Y) = \int p(z,y) \log \frac{p(y,z)}{p(y)p(z)} \, dy\, dz = \int p(z,y) \log \frac{p(y|z)}{p(y)} \, dy\, dz \tag{2.15}$$

$$I(X,Z) = \int p(x,z) \log \frac{p(x,z)}{p(x)p(z)} \, dz\, dx = \int p(x,z) \log p(z|x) \, dz\, dx - \int p(z) \log p(z) dz \tag{2.16}$$

Here, Z is the learned lattent representation, and β is a trade-off parameter. $I(Z,Y)$, represents the mutual information between the representation and the output, while the term $I(X,Z)$, represents the mutual information between the input data and the lattent representation, weighted by β.

The three random variables X, Z and Y form a Markov chain: $Y \leftrightarrow X \leftrightarrow Z$. By this assumption, we find that Y and Z are d-separated given X, i.e., conditionally independent, such that $p(Z|X,Y) = p(Z|X)$. This leads to the following factorization of the joint distribution:

$$p(X,Y,Z) = p(Z|X)\, p(Y|X)\, p(X) \tag{2.17}$$

Calculating both mutual information measurements is intractable. Alemi et al. [1] suggested using VI to introduce a lower bound to the IB objective Eq. (2.14). They showed both an upper and a lower bound for mutual information using VI. Specifically, Eq. (2.16) requires calculating $p(z) = \int p(z|x)\, p(x)\, dx$, which is intractable because it involves the calculation of $p(x)$. Hence, $r(z)$ is introduced as a variational approximation to $p(z)$. Since:

$$KL[p(Z)\,||r(Z)] \geq 0 \rightarrow \int p(z) \log p(z) dz \geq \int p(z) \log r(z) dz \tag{2.18}$$

we get the following upper bound:

$$I(X,Z) \leq \int p(x,z) \log p(z|x)\, dx\, dz - \int p(z) \log r(z) dz = KL[p(Z|X)\,||r(Z)] \tag{2.19}$$

As for Eq. (2.15), the calculation requires calculating $p(y|z)$. Applying the Markovian assumption, we get the following.

$$p(y|z) = \int p(x,y|z)\, dx = \int \frac{p(y|x)\, p(z|x)\, p(x)}{p(z)} dx \tag{2.20}$$

Eq. (2.20) is intractable since the above requires calculating $p(x)$. By introducing the variational approximation $q(y|z)$, we get that

$$KL[p(Y|Z)||q(Y|Z)] \geq 0 \rightarrow \int p(y|z) \log p(y|z) dy \geq \int p(y|z) \log q(y|z) dy \tag{2.21}$$

which leads to the following lower bound:

$$I(Z,Y) \geq \int p(y,z) \log \frac{q(y|z)}{p(y)}\, dy\, dz = \int p(y,z) \log q(y|z)\, dy\, dz + H(Y) \tag{2.22}$$

$H(Y)$, the entropy of Y, is a positive constant and therefore it can be ignored in the optimization problem. Focusing on the first term of EQ 2.22, by writing the marginalization form of $p(y,z)$ and by applying the Markovian assumption, it follows that:

$$p(y,z) = \int p(y,z,x) dx = \int p(x) p(y|x) p(z|x) dx$$

This leads to the following lower bound:

$$I(Z,Y) \geq \int p(x) p(y|x) p(z|x) \log q(y|z)\, dx\, dy\, dz \tag{2.23}$$

Putting it all together and considering both EQ 2.19 and EQ 2.23, we get the following lower bound to the IB objective:

$$I(Z,Y) - \beta \cdot I(X,Z) \geq \int p(x) p(y|x) p(z|x) \log q(y|z)\, dx\, dy\, dz - \beta \cdot KL[p(Z|X)\,||r(Z)] \tag{2.24}$$

In practice, we can approximate $p(x,y) = p(y|x)p(x)$ using the empirical data distribution $p(x,y) = \frac{1}{N}\sum_{n=1}^{N} \delta_{x_n}(x)\delta_{y_n}(y)$ and hence we can write the VIB objective function as follows:

$$J_{VIB} = \frac{1}{N}\sum_{n=1}^{N}\left[\int p(z|x_n)\log q(y_n|z)dz - \beta \cdot KL[p(z|x_n)\,||\,r(z)]\right] \tag{2.25}$$

2.2 Related work

2.2.1 Time Series Charactaristics

Time series data presents unique challenges that set it apart from other data types, such as images or text. These challenges arise from the inherent properties of time series, the varying sources from which they are derived, and the wide range of tasks they are used for. Some of the key challenges in time series data include:

- **Temporal Dependency**: Time series data is inherently sequential, meaning that each data point depends on previous ones. Mathematically, a time series can be represented as:
$$x_t = f(x_{t-1}, x_{t-2}, \ldots, x_{t-k}) + \varepsilon_t$$
where x_t is the value at time t, k represents the order of dependency, and ε_t is a noise term.

- **Non-Stationarity**: Many time series exhibit non-stationarity, meaning that their statistical properties, such as mean and variance, change over time. A time series is stationary if:
$$\mathbb{E}[x_t] = \mu, \quad \text{Var}(x_t) = \sigma^2, \quad \text{Cov}(x_t, x_{t+\tau}) = \gamma(\tau)$$
remain constant over time. If these properties vary, the series is non-stationary.

- **High Dimensionality and Correlations**: Multivariate time series involve multiple interdependent variables, where dependencies exist both within each variable over time (**intra-series correlations**) and between different variables (**inter-series correlations**). A multivariate time series can be represented as
$$\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_T\}$$
where each $\mathbf{x}_t \in \mathbb{R}^d$ is a d-dimensional vector representing multiple features at the time step t.

  The dependencies can be captured using covariance matrices:

  - **Intra-series correlation** (temporal dependency within each variable):
$$\text{Cov}(x_t^{(i)}, x_{t+\tau}^{(i)}) = \gamma_i(\tau), \quad \forall i \in \{1, \ldots, d\}$$
where $\gamma_i(\tau)$ measures the autocorrelation for feature i over lag τ.

– **Inter-series correlation** (relationships between different variables):

$$\text{Cov}(x_t^{(i)}, x_t^{(j)}) = \rho_{ij}, \quad \forall i \neq j$$

where ρ_{ij} quantifies the correlation between variables i and j at the same time step.

- **Irregularity (Varying Sample Rates and Sparsity)**: In multivariate time series, different variables may be sampled at different rates, leading to missing values and a sparse data representation. Given a set of d univariate time series, the observation times for each variable i may differ:

$$t_1^{(i)}, t_2^{(i)}, \ldots, t_{N_i}^{(i)}, \quad i = 1, \ldots, d$$

where N_i is the number of observations for the i-th feature. Since N_i varies across features, the resulting time series matrix is sparse:

$$\mathbf{X} \in \mathbb{R}^{d \times T}$$

Irregularity can require interpolation or imputation and may complicate modeling and analysis.

- **Seasonality, Trends, and Residual Components**: A time series can often be decomposed into three main components:

$$x_t = T_t + S_t + R_t$$

– *Trend* (T_t): The long-term movement in the data, ignoring short-term fluctuations, can be modeled using a low-degree polynomial:

$$T_t = \beta_0 + \beta_1 t + \beta_2 t^2 + \ldots$$

where β_i are coefficients, and t is the time index.

– *Seasonality* (S_t): Periodic fluctuations that repeat at regular time intervals, which can be represented as a sinusoidal function:

$$S_t = A\cos(2\pi f t + \phi)$$

where A is the amplitude, f is the frequency, and ϕ is the phase shift.

– *Residual Component* (R_t): The variation in the data that cannot be explained by the trend or seasonality, often modeled as a stochastic process:

$$R_t \sim \mathcal{N}(0, \sigma^2)$$

indicating that R_t follows a normal distribution with mean 0 and variance σ^2.

2.2.2 Time Series Tasks

A time series $\mathbf{x} = \{x_1, x_2, \ldots, x_T\}$ is a sequence of data points ordered in time, where T represents the length of the sequence, and each x_t corresponds to an observation at time t. Time series analysis involves various tasks, each with its own set of techniques and challenges. The main tasks in time series analysis include:

1. **Time Series Classification**: Time series classification involves assigning a label to a sequence of observations over time. The task is to assign a label $y \in \mathscr{Y}$, where $\mathscr{Y}$ is the set of all possible labels. The goal is to learn a function $f : \mathbf{x} \to \hat{y}$ that minimizes the classification error, where $\hat{y}$ is the predicted label. The loss function commonly used for classification is **cross-entropy**:

$$\mathscr{L} = -\sum_{y \in \mathscr{Y}} y \log(\hat{y})$$

   where $\hat{y}$ is the predicted probability distribution for the label y.

2. **Time Series Regression**: In time series regression, the goal is to predict a continuous value based on a sequence of data points. The task is to predict a continuous target $y \in \mathbb{R}$. The objective is to learn a function $f : \mathbf{x} \to \hat{y}$ that minimizes the regression error, where $\hat{y}$ is the predicted value. This is typically done using **Mean Squared Error (MSE)** as the loss function:

$$\mathscr{L} = \frac{1}{T} \sum_{t=1}^{T} (y_t - \hat{y}_t)^2$$

3. **Time Series Forecasting**: Forecasting is the task of predicting future values in a time series based on past observations. Given historical data, the goal is to predict future values $\mathbf{y}_{T+1}, \mathbf{y}_{T+2}, \ldots$. This can be framed as a problem of predicting k-steps ahead, where the predicted value is:

$$\hat{y}_{T+k} = f(\mathbf{x})$$

   where k is the number of steps ahead.

4. **Anomaly Detection**: Anomaly detection in time series data focuses on identifying unusual patterns that deviate from expected behavior. The goal is to detect time points or subsequences that significantly deviate from the normal patterns. This is typically represented as a function $\hat{A}_t$ that outputs 1 if an anomaly is detected at time t, and 0 otherwise:

$$\hat{A}_t = \begin{cases} 1, & \text{if anomaly at time } t \\ 0, & \text{otherwise} \end{cases}$$

5. **Time Series Clustering**: Time series clustering involves grouping similar time series data points or subsequences. The goal is to partition a set of time series into K clusters $C_1, C_2, \ldots, C_K$, such that the time series in each cluster are more similar to each other than to those in other clusters.

2.2.3 Time Series Classification

TSC can be approached using both classical machine learning methods and deep learning techniques. Classical methods typically rely on feature engineering, distance-based metrics, or tree-based models to classify time series. These approaches have been widely studied and remain competitive, especially for small datasets. However, deep learning has emerged as a powerful alternative, offering automatic feature extraction and end-to-end learning capabilities.

Classical Machine Learning Methods

Classical machine learning methods can be further categorized into three main groups: direct classification methods, feature engineering-based methods, and hybrid methods.

Within the realm of direct classification methods, there exist two notable subgroups **distance-based** methods and **interval-based** methods. Distance-based methods involve quantifying dissimilarities between time series and utilizing these measures for classification. Historically, distance measures were integrated with the K-Nearest-Neighbors (K-NN) algorithm, with the 1-NN Dynamic Time Warping (DTW) standing out as a noteworthy approach [90]. A recent advancement in this domain is the ProximityForest ensemble [73], which employs a combination of 11 different distance functions and Proximity Tree classifiers.

In contrast, **interval-based** methods entail dividing time series into random intervals and subjecting them to processing by a random forest, where each tree employs distinct splits of the input data. An extension of this concept is the Randomized Supervised Time Series Forest [8], a method that markedly enhances previous methods by introducing randomness in the positioning of intervals within time series.

Feature engineering-based methods involve creating complex features from time series data and feeding them into traditional classifiers like linear models or random forests. Various assumptions about the data have led to different types of feature engineering methods. A notable example is TSFRESH [15], which extracts 800 features from time series, followed by feature selection before classification. A more advanced approach is FreshPRINCE [76] that utilizes a rotation forest classifier over the features, without performing feature selection.

Another approach involves **kernel-based** methods that generate a large number of kernels to create features for simpler classifiers. A well-known model is ROCKET [20], which generates thousands of convolutional kernels, applies pooling operations, and employs these outputs as features in a Ridge regression classifier. Further extensions of ROCKET that combine additional types of kernels are the Multi-ROCKET [103] and Hydra-MultiROCKET (Hydra-MR) [21].

Shapelets are discriminative subsequences that can indicate a class within time series datasets. The Random Dilated Shapelet Transform [42] randomly selects numerous shapelets from the training data and uses features derived from these shapelets to train a linear RIDGE classifier.

An alternative approach is **dictionary-based**. It treats phase-independent subsequences as words using techniques like Symbolic Fourier Approximation. These methods record

the frequency of repeating patterns to create features. Algorithms like WEASEL-D [92] and TDE [79] fall under this category.

Hybrid methods combine various aforementioned approaches. An notable exemple is HIVE-COTE v2.0 (HC2) [78], an ensemble method that integrates multiple techniques. This approach stands as the state-of-the-art solution across numerous benchmark datasets.

Deep learning has played a crucial role in time series classification (TSC) since the early 2010s, when advancements in neural network architectures, computational power, and large-scale datasets enabled end-to-end learning from raw time series data. Traditional machine learning approaches relied on handcrafted feature extraction, limiting their ability to generalize across diverse datasets. The emergence of deep models, particularly Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), demonstrated that neural networks could automatically extract meaningful temporal patterns, reducing the need for manual feature engineering. This shift was driven by the success of deep learning in computer vision and natural language processing, inspiring researchers to explore its potential for time series analysis.

Convolutional Neural Networks

Convolutional Neural Networks (CNNs) have been widely used for TSC due to their ability to learn hierarchical representations from raw data. Unlike traditional machine learning methods, which rely on handcrafted features, CNN-based approaches extract relevant patterns automatically, improving classification performance.

The first CNN models for TSC focused on adapting convolutional architectures to handle sequential data. Multi-Channel Deep Convolutional Neural Networks (MC-DCNN) [123] applied independent convolutions to each input channel before merging them via fully connected layers, while MC-CNN [84] processed all input channels simultaneously to capture temporal and spatial dependencies. Fully Convolutional Networks (FCN) [113] improved upon these by removing fully connected layers and incorporating Global Average Pooling (GAP) for flexible input handling and better interpretability. Residual Networks (ResNet) [44] further enhanced CNN-based TSC by introducing residual connections to mitigate vanishing gradients, leading to deeper and more effective models.

Beyond standard convolutional architectures, several modifications have been introduced to enhance CNN-based TSC. Dilated Convolutions (DCNNs) [67] expanded the receptive field without increasing parameters, improving the ability to model long-range dependencies. Disjoint-CNNs [93] separated convolution operations into temporal and spatial components for more efficient feature extraction. Another approach involved transforming time series into image-like representations, where methods such as Gramian Angular Fields (GAF) and Markov Transition Fields (MTF) [112] allowed CNNs to learn spatial relationships within time series data. While these transformations improved pattern recognition, they often led to increased computational costs and potential information loss.

To further enhance feature extraction, multi-scale convolutional architectures emerged. Models like Multi-Scale CNNs (MCNN) [18] and Time LeNet [64] applied convolu-

tions at different scales, inspired by computer vision techniques. The most notable development was InceptionTime [53], which extended the Inception architecture by using bottleneck layers and multi-scale convolutions to capture diverse temporal patterns efficiently. InceptionTime's success led to multiple extensions, including EEG-Inception [100], InceptionFCN [109], and LITE [54], which introduced efficiency-focused modifications such as depthwise separable convolutions. These advancements have solidified CNNs as a dominant deep learning approach for TSC, balancing accuracy, interpretability, and computational efficiency.

Recurrent Neural Networks

Recurrent Neural Networks (RNNs) are designed to model sequential dependencies by incorporating internal memory, making them well-suited for TSC. Early RNN-based approaches classified time series by analyzing their dynamic behavior using sequence-to-sequence architectures, where sub-series were first classified, and an argmax operation determined the final label [51]. To improve parallelization and capture long-term dependencies, hierarchical RNN architectures were introduced, where multiple RNN layers processed different aspects of the sequence before merging their outputs [23, 31, 45]. However, standard RNNs suffer from vanishing and exploding gradients due to their iterative training process via backpropagation through time [87]. To address these issues, Long Short-Term Memory (LSTM) networks [47] and Gated Recurrent Units (GRU) [16] were developed, incorporating gating mechanisms to regulate information flow and improve memory retention over long sequences. LSTM-based models, such as Sequence-to-Sequence with Attention, have been successfully applied to TSC, where an encoder-decoder framework extracts meaningful representations from time series data before classification [104]. Similarly, GRU-based sequence autoencoders have been proposed to process variable-length time series, leveraging unsupervised pretraining on large datasets to enhance accuracy [74].

To further improve TSC performance, hybrid models combining CNNs and RNNs have been explored, leveraging CNNs for spatial feature extraction and RNNs for modeling temporal dependencies. Architectures like MLSTM-FCN [58], TapNet [120], and SMATE [124] integrate convolutional and recurrent components in dual-network architectures, where the extracted CNN and RNN features are concatenated before classification. However, these models treat spatial and temporal representations independently, leading to limited interaction between feature extraction pathways. To address this, architectures such as GCRNN [69] and CNN-LSTM [81] employ layer-wise integration of CNNs and RNNs, enabling better feature fusion. Despite their potential, RNN-based models for TSC face challenges, including difficulties in parallelization, high computational costs, and struggles with long-range dependencies [87, 63, 104, 88]. As a result, while RNNs remain an important tool in deep learning for time series analysis, their use in TSC is often complemented by CNNs and attention-based mechanisms to enhance efficiency and scalability.

Attention-based methods

Attention-based methods have emerged as a powerful alternative to CNNs and RNNs for TSC. Although CNNs excel at capturing local temporal and spatial dependencies, they struggle with long-range dependencies, and RNNs, despite their sequential modeling capabilities, are computationally expensive and difficult to parallelize. Attention mechanisms provide a broader receptive field, allowing models to focus on the most informative features while suppressing irrelevant ones. Inspired by their success in natural language processing, attention models have been adapted to time series analysis, significantly improving classification performance. Self-attention, originally developed for sequence modeling, has been applied effectively to TSC in models such as CA-SFCN [43] and LAXCAT [49], which integrate temporal and variable attention mechanisms to capture complex dependencies within multivariate time series. Other models, like WHEN [111], incorporate wavelet-based and DTW-based attention to enhance feature extraction in nonstationary and distorted time series. Additionally, Squeeze-and-Excitation networks [50] improve feature selection by dynamically recalibrating channel-wise representations, making them highly effective when combined with CNNs and attention mechanisms.

The success of multi-head self-attention in transformers has led to its adoption for TSC, enabling models to efficiently capture both global and local dependencies. Early transformer-based models, such as Simply Attend and Diagnose [97], demonstrated the effectiveness of self-attention in clinical time series classification by incorporating positional encodings and dense interpolation techniques. More recent approaches, including Gated Transformer Networks [98], enhance attention mechanisms by introducing gating mechanisms to merge multiple attention layers adaptively. Flexible multi-head linear attention flexible multi-head linear attention [122] further improves transformers for TSC by integrating deformable convolutional blocks and knowledge distillation to enhance locality awareness. ConvTran [33], which was consider a state-of-the-art for multivariate TSC, refines positional encoding methods to better represent time series structures, combining novel absolute and relative position encodings within a transformer framework. These advancements highlight the growing impact of attention-based models in TSC, providing both interpretability and superior performance in capturing long-range dependencies.

Irregularly Sampled Time Series

Irregularly sampled time series (ISTS) refer to time series data where observations occur at uneven time intervals rather than fixed, regular intervals. This setting is common in many real-world applications, such as healthcare, where patient vitals are recorded at varying times, finance, where stock trades occur asynchronously, and environmental monitoring, where sensor readings depend on external factors.

A common approach to handling ISTS is to convert it into a regularly spaced format, where a predefined set of discrete time steps is used, and missing values appear in time steps with no recorded observations. This transformation introduces the challenge of missing data, which must be addressed to enable effective downstream analysis.

The missing data-based approach assumes that the time series consists of non-overlapping uniform time intervals, and periods without recorded values are treated as missing data points. There are two primary methods to handle ISTS under this framework: a **two-step approach**, where missing values are first imputed and then the completed time series is used for classification, and an **end-to-end approach**, where the model directly processes the irregularly sampled data while handling missing values as part of the classification task.

In the two-step approach, traditional imputation techniques have been widely used, such as matrix completion and statistical interpolation methods ([117], [39]), followed by standard time series classification models ([38], [119], [59]). More advanced imputation methods leverage deep learning, including GP-VAE ([32]), which models missing values probabilistically using a Gaussian Process-based Variational Autoencoder, BRITS ([9]), which applies a bidirectional LSTM trained with self-supervision to predict missing values, and GRUI-GAN ([70]), which uses a GAN-based architecture where a generator learns to fill missing values while preserving the temporal structure of the data.

While the two-step approach is widely used, it assumes that imputation and classification are independent, which is often unrealistic. Instead, end-to-end models jointly learn to handle missing data while optimizing for the classification task. Several deep learning architectures have been proposed for this purpose, including GRU-D ([10]), which extends GRU by modeling missing values as a combination of the last observed value and the mean value through a decay mechanism; Phased-LSTM ([83]), which introduces time gating mechanisms to process inputs sampled at irregular intervals; Interpolation Networks ([94]), which use a semi-parametric interpolation network followed by a standard deep learning classifier; and Set Functions for Time series (SeFT) ([48]), which treats time series as sets of unordered observations, using an attention-based embedding mechanism before classification.

The two-step approach remains a reasonable baseline, but end-to-end models are increasingly preferred as they avoid error propagation from imputation to classification, leverage missing patterns as informative signals rather than artifacts, and adapt naturally to real-world ISTS datasets, where missingness often conveys meaningful structure.

Advancements and Challenges in Deep Learning for Time Series Classification

Deep learning has significantly advanced TSC by introducing models capable of capturing complex temporal dependencies. CNNs effectively extract local patterns, while attention-based models, particularly transformers, enhance long-range dependency modeling. Additionally, other approaches such as Graph Neural Networks (GNNs) enable the modeling of structural relationships in multivariate time series, while transfer learning facilitates knowledge adaptation from related domains. Data augmentation techniques, including jittering, window slicing, and magnitude warping, further improve generalization by artificially expanding training data. However, the reliance on large labeled datasets remains a challenge. Self-supervised learning mitigates this issue by leveraging unlabeled data to learn meaningful representations, complementing semi-supervised frameworks.

2.2.4 Self-Supervised and Semi-Supervised Learning for Time Series Classification

Semi-supervised learning (SSL) is a paradigm that leverages both labeled and unlabeled data to improve model performance. Formally, given a dataset $\mathscr{D} = \mathscr{D}_l \cup \mathscr{D}_u$, where $\mathscr{D}_l = \{(x_i, y_i)\}_{i=1}^{N_l}$ represents the labeled instances and $\mathscr{D}_u = \{x_j\}_{j=1}^{N_u}$ represents the unlabeled instances, SSL aims to learn a function $f_\theta : \mathscr{X} \to \mathscr{Y}$ that generalizes well to both labeled and unlabeled data.

In TSC, obtaining labeled data can be expensive and time consuming, making SSL a practical alternative to fully supervised approaches. The inherent structure of time series data introduces additional challenges, including temporal dependencies, varying sequence lengths, and potential irregular sampling. These complexities make traditional supervised learning approaches less effective in low-label regimes. By incorporating unlabeled data into the learning process, SSL techniques improve generalization by leveraging the underlying structure of $\mathscr{D}_u$ to guide the decision boundary learned from $\mathscr{D}_l$.

Self-supervised learning is an unsupervised representation learning framework in which models extract meaningful features from unlabeled data through **pretext tasks**. The learned representations are then fine-tuned for downstream tasks such as classification, anomaly detection, and forecasting. Self-supervised learning is particularly beneficial in time series analysis, where meaningful patterns are often embedded in long-term temporal dependencies.

Contrastive learning is one of the most widely used self-supervised learning techniques. It encourages models to learn representations by maximizing the similarity between different augmented versions of the same sample (positive pairs) while pushing apart representations of different samples (negative pairs). The InfoNCE loss (Noise Contrastive Estimation) is commonly used for contrastive learning:

$$\mathscr{L}_{\text{NCE}} = -\mathbb{E}\left[\log \frac{\exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_i^+)/\tau\right)}{\sum_j \exp\left(\text{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau\right)}\right] \tag{2.26}$$

where $\mathbf{z}_i$ and $\mathbf{z}_i^+$ are representations of the same instance under different augmentations, $\mathbf{z}_j$ are negative samples, and τ is a temperature scaling parameter.

Several works have successfully adapted contrastive learning to time series. TS-TCC [27] applies contrastive loss across temporal and contextual views of time series, ensuring that representations remain invariant under transformations. Similarly, SimTSC [72] demonstrates that contrastive pretraining significantly improves the generalization of time series classifiers.

Self-prediction is another form of self-supervised learning that focuses on reconstructing missing or masked data. Transformer-based models such as BENDER [61], TST [118], and TATNet [13] leverage masked time series modeling, where certain parts of the input are masked, and the model learns to reconstruct them. This encourages the model to develop meaningful representations of the temporal structure. Additionally, Series2Vec [34] introduces domain-specific pretext tasks, predicting temporal-spectral

representations rather than relying on handcrafted augmentations. Such methods have proven particularly effective in large-scale datasets like EEG and speech processing.

Semi-supervised learning integrates labeled and unlabeled data within a single training phase, unlike self-supervised learning, which relies on a two-step process of pretext task training followed by fine-tuning. The two most common techniques in semi-supervised learning are pseudo-labeling and self-training, both of which iteratively refine model predictions by leveraging unlabeled data alongside labeled examples to enhance classification performance

Pseudo-labeling, introduced by Dong-Hyun Lee [65], is a simple yet effective SSL technique where the model assigns labels to unlabeled samples based on its highest-confidence predictions. Given an unlabeled sample, the model selects the most probable class, and these pseudo-labels are used in subsequent training iterations. By encouraging decision boundaries to align with low-density regions in the data distribution, pseudo-labeling improves classification performance. For TSC, pseudo-labeling has been adapted to better capture temporal dependencies. SemiTime [30] enforces consistency between past and future segments, ensuring that pseudo-labels remain stable across time.

Self-training builds upon pseudo-labeling by iteratively refining pseudo-labels. The model is first trained on labeled data, then generates pseudo-labels for the unlabeled set, selecting the most confident predictions to augment the labeled dataset. SSTSC [116] introduced one of the earliest applications of semi-supervised learning in time series, demonstrating that self-training can improve classifiers with limited labeled data.

Beyond a single classifier, self-training can be expanded to multiple models, where different classifiers pseudo-label data for each other, as explored in **Self-Training** [2]. Further extensions include co-training and multi-view learning, where models trained on different feature subsets or transformations provide complementary pseudo-labels. TS-TFC [71] applies this approach to time series classification by propagating labels across both temporal and frequency-domain views.

Additionally, advanced self-training techniques such as reinforced self-training and domain adaptation have been investigated to further improve robustness [2]. TS-TCC [27] integrates self-training with contrastive learning, ensuring that pseudo-labels remain discriminative across different time segments, while SimTSC [72] extends contrastive semi-supervised learning for time series by leveraging SimCLR-based [12] representation learning. More recent approaches have incorporated domain-specific constraints to enhance semi-supervised learning in time series tasks. For example, Temporal-Frequency Co-training (TS-TFC) [71] employs co-training in different views of the time domain and the frequency domain, improving classification performance through complementary information.

Deep-SSL-TSC [40] explored how methods originally developed for image classification, such as MixMatch [6], Virtual Adversarial Training (VAT) [80], and Mean Teacher [105], can be transferred to time series classification, demonstrating the need for domain-specific augmentations. SemiTime [30], which leverages temporal consistency constraints to ensure that pseudo-labels remain meaningful over time. The method enforces consistency between temporally adjacent segments, effectively leveraging unlabeled data to improve classification performance.

Variational Information Bottleneck with Gaussian Process Priors

We propose a novel model named **Variational Information Bottleneck with Gaussian Process Priors** that can handle time seires data for classification tasks. In this section, we define the prior distribution $r(Z)$, the encoder $p(Z|X)$, the decoder $q(y|Z)$, and specify the GP-VIB objective, similarly to Eq. (2.25). Since we are dealing with TSC, our decoder functions as a classifier.

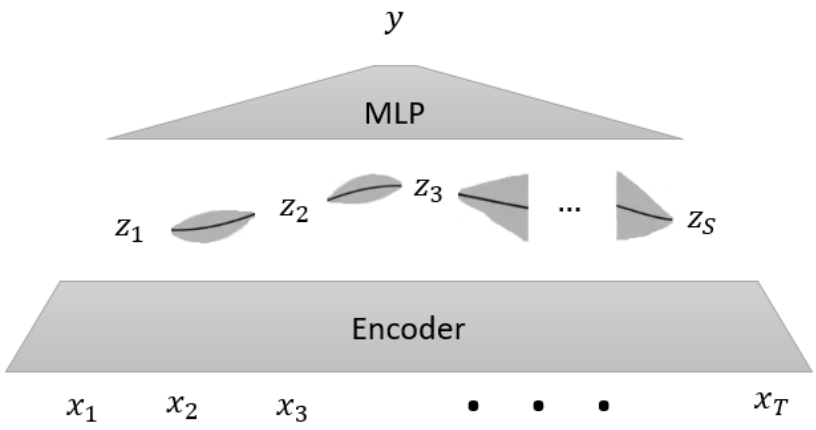


Figure 3.1: Overview of the GP-VIB architecture. The encoder is comprised of InceptionTime blocks and a compression block. The latent sequence, Z, is the input to the MLP decoder, which in turn predicts y.

3.1 Problem Setting

Let $X \in \mathbb{R}^{T \times d}$ denote a multivariate time series of length T, where each instance in the sequence is a vector of size d features such that $x_t = [x_{t1}, \ldots, x_{td}] \in \mathbb{R}^d$. For each time series, there is a single label $y \in Y$, where Y is a finite set of categories.

Hence, our modeling approach suggests constructing a dense (latent) time seires representation $Z \in \mathbb{R}^{S \times d'}$ of the input time series X that preserves the relevant information

for the classification task, i.e. predicting y. Each time series in the latent space consists of S data points, where each instance in the sequence is a vector of size d' such that $z_s = [z_{s1}, \ldots, z_{sd'}] \in \mathbb{R}^{d'}$.

3.2 A Graphical Model Perspective

The IB principle states that random variables Y, X, Z form a Markov chain. In the current setting, it should be noted that X and Z are time series of different lengths. This leads to the graphical model depicted in Fig. 3.2.

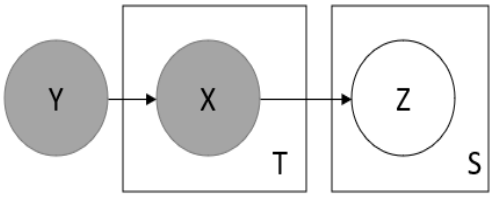


Figure 3.2: A graphical model illustration of the GP-VIB model. Y, X, Z form a Markov chain, while X and Z are time series of lengths T and S, reps.

The graphical model perspective suggests that Z_n and y_n are d-separated given X_n namely y_n and Z_n are independent. Hence, the GP-VIB model architecture consists of two parts:

- The encoder (inference model): given the input X_n the encoder models the posterior distribution $p(Z_n|X_n)$.
- The clasifaier (prediction model): given the latent representation the decoder models $p(y_n|Z_n)$.

Similar to their role in the VIB model, both the encoder and the clasifaier are implemented using deep neural networks.

3.3 Model Architecture

3.3.1 The Prior Distribution of Z

The definition of the prior $r(Z)$ is an important step in variational inference. We would like to choose a prior that reflects the time dynamics and relationships between features over time. Following [32], we set the prior to be a GP, specifically a multivariate Gaussian with mean 0 and a covariance matrix constructed using a Cauchy kernel. We further assume that every feature in the latent space is a GP, i.e.

$$z_{1:T,j} \sim GP\left(m_z(\cdot), k_z(\cdot,\cdot)\right),$$

where all features in the latent space share the same prior.

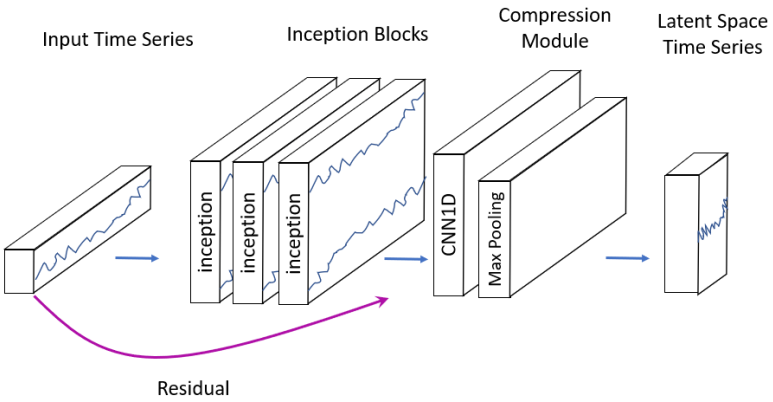


Figure 3.3: GP-VIB encoder overview. The encoder is comprised of 3 InceptionTime modules followed by a residual module. Then the compression block compresses the time series into the desired latent space size.

3.3.2 The Encoder

The encoder models the posterior distribution, $p(Z_n|X_n)$, where X_n is a time series. We use a neural network to approximate the parameters of the variational distribution. The encoder consists of three InceptionTime blocks (Fig. 2.1) with a residual layer followed by a compression block, which compresses the time series to the desired latent space dimension.

To set the number of channels in the time series, we use a 1-dimensional CNN layer, where the number of output filters determines the number of channels in the latent time series. To set the length of the latent time series, we use an adaptive max pooling layer.

Next, we apply the structured variational inference algorithm and extract the mean and covariance matrix of the latent representation. This leads to the following posterior:

$$p(Z_n|X_n) = p(Z_n|f_\phi(X_n)),$$

where $f_\phi(\cdot)$ is a neural network with parameters ϕ.

3.3.3 The Decoder

The decoder, $q(y_n|Z_n)$, is a multinomial distribution, conditioned on the latent series Z_n, such that:

$$q(y_n|Z_n) = q(y_n|g_\theta(Z_n)),$$

where $g_\theta(\cdot)$ is a nonlinear function with parameters θ that outputs the logits of the multinomial distribution.

Thus, the log-likelihood expression $\log q(y_n|g_\theta(Z_n))$ corresponds to the cross-entropy loss function. As for $g_\theta(\cdot)$, we use a "Flattened" decoder, which takes the latent time series and flattens it into a single vector. This vector serves as an input to a Multi-Layer Perceptron (MLP), which outputs the logits of the label.

3.4 IB Objective Lower Bound

By rewriting the VIB objective we obtain the GP-VIB objective:

$$J_{\text{GP-VIB}} = \frac{1}{N}\sum_{n=1}^{N} \mathbb{E}_{p\left(Z_n \mid f_\phi(X_n)\right)} \log q\left(y_n \mid g_\theta(Z_n)\right) - \beta\, KL\left(p\left(Z_n \mid f_\phi(X_n)\right) \middle\| r(Z_n)\right). \tag{3.1}$$

Supervised Experiments

In this section, we present a comprehensive evaluation of our approach through three experimental settings. First, we compare different VAE and VIB architectures to assess their representation learning capabilities. Next, we benchmark our model against state-of-the-art methods for multivariate TSC, demonstrating its effectiveness in structured temporal data. Finally, we evaluate our model on ISTS data, motivated by our observations during the initial experiments, where the model exhibited robustness to missing data.

4.0.1 Datasets

Healing MNIST

Electronic medical records (EMR), which are ISTS by nature, are high dimension and highly sparse data. It is quite rare to find publicly available EMR data, due to privacy concerns. Krishnan et al. [62] created a dataset called Healing MNIST, which is designed to reflect attributes of ISTS that somewhat mimics medical data properties. The dataset contains sequences of the same digit that rotate randomly between frames. The analogy to healthcare is drawn by the fact that every frame (image of digit) is a collection of measurements (every pixel is a measure). This presumably describes a patient's health state. The frames contains around 60% missing pixels and the rotation between two consecutive frames is normally distributed.

UEA Time Series Classification Archive

The UEA Time Series Classification Archive is a critical resource for multivariate time series classification. It includes datasets from diverse real-world applications such as motion capture, medical diagnostics, and industrial processes, presenting complex and dynamic challenges. These datasets require models to capture dependencies across multiple time series variables and handle missing data effectively. By providing a rich and varied collection, the UEA archive serves as a benchmark for evaluating models that need to learn intricate temporal relationships within multivariate sequences.

MIMIC-III Mortality Prediction

MIMIC-III [56] is a widely-used, freely accessible dataset of distinct ICU stays of patients. The dataset contains around 21,000 stays with demographic data about the patient collected at admission time and up to 16 time series measurements of vital signs and lab results. Out of the tasks defined over this dataset, we choose to focus on the task of mortality prediction, namely predicting whether a patient will die during their hospital stay, using only data from the first 48 hours of the ICU stay.

Physionet 2012 Mortality Prediction Challenge

The 2012 Physionet challenge dataset [95] contains 12,000 ICU stays of 48 hours. For each stay there is a demographic data about the patients collected at admission time, and up to 37 time series measurements of vital signs and lab results. Not all measurements were sampled at the same rate and some of the measurements were sampled only when needed. There are several labels provided with this dataset. We chose to predict whether a patient will die during the hospital stay (mortality).

4.0.2 Comparison of VIB and VAE Architectures

GP-VAE has been proposed as an imputation method for time series data. Hence, we used the Healing MNIST dataset [62], which is designed to reflect attributes of irregularly sampled time series that are common in the healthcare domain. We followed the exact experimental setup used in [32]. For comparison, we used VAE [60], HI-VAE [82], GP-VAE [32], a logistic regression model, VIB [1], and GP-VIB. We then compared three performance metrics: Accuracy, AUCROC, and AUPRC. To evaluate the VAE models, we implemented them using the architectures proposed in [32], applying a logistic regression model to the imputed data for classification. Additionally, we tested the logistic regression model directly on the incomplete data. For the GP-VIB architecture, we implemented the encoder and prior used in GP-VAE. To ensure a fair comparison between our model and the VAE models, we used a single linear output layer for the decoder.

Table 4.1: Performance over HMnist dataset.

Model	Accuracy	AUROC	AUPRC
Logistic Regression	66.6 ± 0.0	92.2 ± 0.0	63.5 ± 0.0
VAE	70.0 ± 2.7	94.2 ± 0.9	74.2 ± 0.3
HI-VAE	76.6 ± 0.1	96.6 ± 0.0	82.8 ± 0.1
GP-VAE	77.9 ± 0.2	96.5 ± 0.1	84.1 ± 0.3
VIB	83.0 ± 0.0	98.1 ± 0.0	90.0 ± 0.3
GP-VIB	$\mathbf{88.2 \pm 0.6}$	$\mathbf{99.2 \pm 0.0}$	$\mathbf{95.2 \pm 0.2}$

For each architecture, we trained three models with different seeds and reported the mean and standard deviation for each metric. The results are presented in Table 4.1, where evaluation metrics are scaled to 100 for readability, and the best results are highlighted in bold.

GP-VIB consistently outperforms VAE-based approaches across multiple metrics, highlighting its effectiveness in handling missing values in TSC. While GP-VIB and GP-VAE share architectural similarities, they serve different objectives: GP-VAE focuses on learning a latent space conducive to data reconstruction, while GP-VIB emphasizes a discriminative latent space tailored for downstream tasks. In TSC scenarios with missing data, GP-VAE follows a two-step approach, imputing missing values before classification. In contrast, GP-VIB directly learns the dynamics of input time series with missing values, compressing them into a representation that preserves discriminative information. This end-to-end approach enables GP-VIB to consistently achieve superior performance across all evaluated metrics.

4.0.3 TSC results

We evaluated our model using the UEA [3] benchmark, selecting 26 out of 30 datasets after excluding those with unequal lengths or missing values. Following Middlehurst et al. [77], we trained GP-VIB 30 times per dataset with different seeds. To assess statistical significance, we applied the Friedman test [22], followed by a Wilcoxon signed-rank test with Holm's alpha (5%) correction [5, 37].

To visualize model comparisons, we used critical difference diagrams [22], ranking classifiers by average performance across datasets. The critical difference, computed via the Nemenyi test, defines significance thresholds, with thick horizontal lines indicating groups of models with no significant difference.

Training GP-VIB posed challenges, particularly for datasets with limited training samples, a common difficulty in deep learning. To mitigate this, we adjusted the latent space size based on dataset size, ranging from 2 for small datasets (50 samples) to 10 for larger ones.

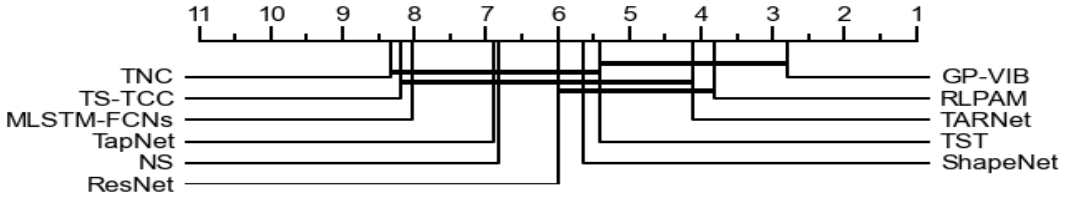


Figure 4.1: Accuracy Critical Difference diagrams on UEA archive, for GP-VIB vs. Top Deep Learning models.

For this set of experiments, we compared between GP-VIB and the best models presented is [14]. In addition, we also used the results presented on the github[1] for missing SOTA models. For comparison, we added the following deep learning models: ShapeNet [66], time series Transformer (TST) [118], ResNet, MLSTM-FCNs [58], Negative samples (NS) [35], TNC [107], TS-TCC [28], TAPNet [120], RLPAM [36] and TARNet [13]. As for machine learning models, we added DrCIF and Arsenal, which are components in the HC2 hybrid ensemble model [78] ,and the ROCKET model [20].

[1] https://github.com/time series-machine-learning/tsml-eval/tree/main/results/classification/Multivariate

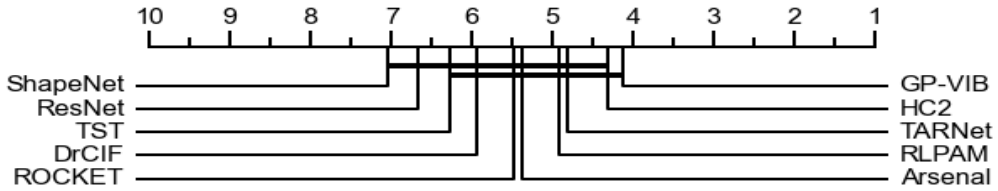


Figure 4.2: Accuracy Critical Difference diagrams on UEA archive, for GP-VIB vs. combined TSC models.

Fig. 4.1 shows that GP-VIB achieves the highest average accuracy rank of the critical different diagrams among the deep learning models. Fig. 4.2 shows that among the top models, including machine learning models, GP-VIB is still at the top of the critical difference diagram with the best average rank among the competitors. In both diagrams, GP-VIB is comparable with the top SOTA models as it is not significantly different from them.

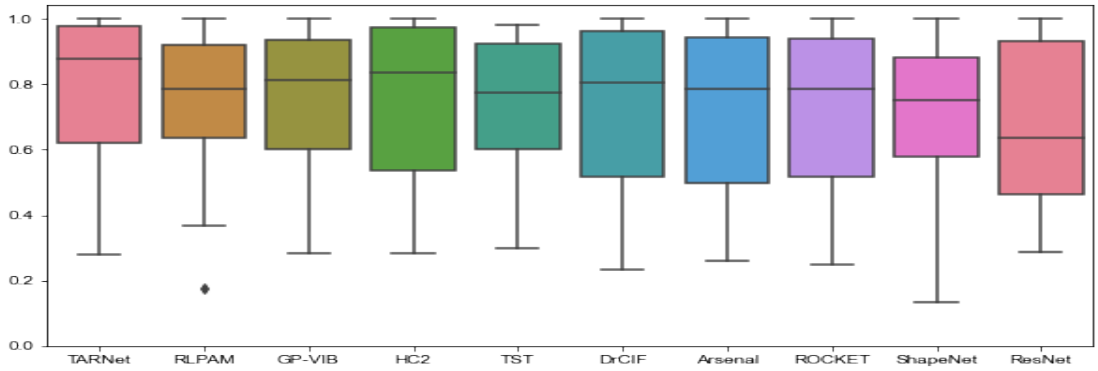
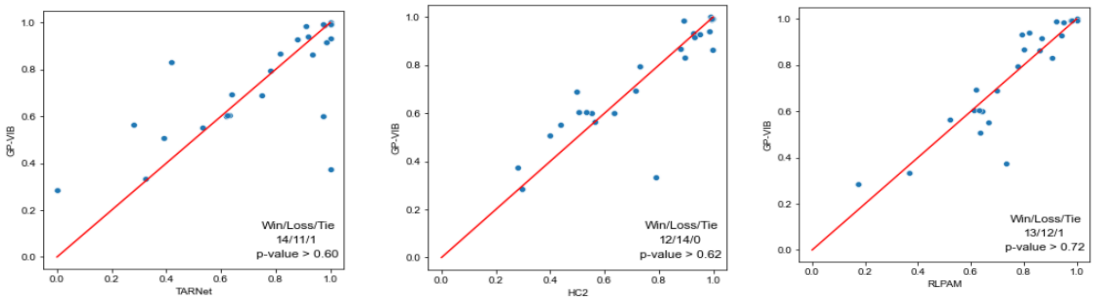


Figure 4.3: UEA Test accuracy box-plot, sorted by mean accuracy per model.

Although GP-VIB achieves the top rank, Fig. 4.3 shows that the mean accuracy of GP-VIB is third to TARNet and RLPAM. The pairwise plots in Fig. 4.4 accept the null hypothesis, with a high p-value, confirming that our model presents performance comparable to the 3 top models.



(a) GP-VIB vs. TARNet (b) GP-VIB vs. HC2 (c) GP-VIB vs. RLPAM

Figure 4.4: Pairwise test accuracy comparison of GP-VIB with HC2, RLPAM, and TARNet over the UEA archive.

4.0.4 Irregularly Sampled Time Series data Results

After achieving strong performance on the HMNIST dataset, we aimed to assess our model's robustness to missing data. In our previous experiments, GP-VIB demonstrated stability and high predictive accuracy even when trained on incomplete or noisy time series, suggesting its potential for handling real-world datasets with irregular sampling. To further evaluate this capability, we tested GP-VIB on benchmark datasets specifically designed for ISTS.

We compared GP-VIB's performance with the results reported by Horn et al. [48], evaluating multiple methods on the MIMIC-III Mortality Prediction and the Physionet 2012 Mortality Prediction Challenge. To ensure a fair comparison, we closely followed their experimental setup. Similar to Horn et al. [48], we trained GP-VIB to optimize AUPRC, as these datasets are imbalanced. We trained three models with different seeds and reported the mean and standard deviation for each metric. Table 4.2 presents the comparison for the Physionet 2012 Mortality Prediction Challenge, and the results for the MIMIC-III Mortality Prediction task.

Table 4.2: Mortality Prediction Performance on Physionet 2012 and MIMIC-III

Physionet 2012				**MIMIC-III**			
Model	Accuracy	AUPRC	AUROC	Model	Accuracy	AUPRC	AUROC
GRU-Simple	82.2 ± 0.2	42.2 ± 0.6	80.8 ± 1.1	GRU-Simple	78.1 ± 0.2	43.6 ± 0.4	82.8 ± 0.0
GRU-D	80.0 ± 2.9	53.7 ± 0.9	86.3 ± 0.3	GRU-D	77.0 ± 1.5	$\mathbf{52.0 \pm 0.8}$	$\mathbf{85.7 \pm 0.2}$
SEFT	75.3 ± 3.5	52.4 ± 1.1	85.1 ± 0.4	SEFT	$\mathbf{79.0 \pm 2.2}$	46.3 ± 0.5	83.9 ± 0.4
IP-Nets	79.4 ± 0.3	51.0 ± 0.6	86.0 ± 0.2	IP-Nets	78.3 ± 0.7	48.3 ± 0.4	83.2 ± 0.5
Phased-LSTM	76.8 ± 5.2	38.7 ± 1.5	79.0 ± 1.0	Phased-LSTM	73.8 ± 3.3	37.1 ± 0.5	80.3 ± 0.4
Transformer	$\mathbf{83.7 \pm 3.5}$	52.8 ± 2.2	$\mathbf{86.3 \pm 0.8}$	Transformer	77.4 ± 5.6	42.6 ± 1.0	82.1 ± 0.3
GP-VIB	76.7 ± 1.6	$\mathbf{55.0 \pm 0.3}$	85.9 ± 0.5	GP-VIB	78.7 ± 1.6	47.8 ± 0.3	84.1 ± 0.3

For the Physionet 2012 Mortality Prediction task, the GP-VIB model outperforms all competitor methods in the AUPRC metric, which was the primary objective across all models. For MIMIC-III Mortality Prediction, GP-VIB ranks third in AUPRC, following GRU-D and IP-Nets. In other evaluation metrics, the model's ranking varies, but it performs relatively better on the MIMIC-III dataset.

Notably, the model rankings differ between MIMIC-III and Physionet 2012. As observed by [48], the splits of the MIMIC-III benchmark tend to produce validation performance that is, on average, about 6% higher than test performance. We observed a similar trend in our experiments. Since model selection is based on validation performance, this discrepancy between validation and test sets may explain the ranking shifts across datasets.

Overall, the GP-VIB model achieved results that are close to state-of-the-art across all metrics, despite not being explicitly designed for handling missing data. Instead, the model relied solely on the information bottleneck mechanism to manage irregular sampling patterns.

Semi Supervised Variational Information Bottleneck with Gaussian Process Priors

We extend the GP-VIB model to a semi-supervised setting for time series data. Specifically, we utilize the prior distribution $r(Z)$ and the encoder $p(Z|X)$ from Chapter **??**. In this section, we introduce a decoder to process unlabeled data and a classifier for labeled instances. By incorporating a decoder, our model learns to reconstruct data from the latent space. If the reconstructed features are correlated with those relevant to the classification task, they enhance the latent representation, allowing us to leverage unlabeled samples more effectively. Finally, we modify the original GP-VIB formulation to integrate both labeled and unlabeled data, deriving the semi-supervised objective.

5.1 Problem Setting

Let $X \in \mathbb{R}^{T \times d}$ denote a multivariate time series of length T, where each instance in the sequence is a vector of size d features, such that

$$x_t = [x_{t1}, \ldots, x_{td}] \in \mathbb{R}^d, \quad \forall t \in \{1, \ldots, T\}.$$

In the semi-supervised setting, we assume a dataset consisting of both labeled and unlabeled time series instances. Specifically, let $\mathscr{D}_l = \{(X^{(i)}, y^{(i)})\}_{i=1}^{N_l}$ denote the labeled set, where each time series $X^{(i)}$ has an associated label $y^{(i)} \in Y$, a finite set of categories. Additionally, let $\mathscr{D}_u = \{X^{(j)}\}_{j=1}^{N_u}$ denote the unlabeled set, where the labels are unknown. The total dataset consists of $N = N_l + N_u$ instances.

Hence, our modeling approach suggests constructing a dense (latent) time series representation $Z \in \mathbb{R}^{S \times d'}$ of the input time series X that preserves the relevant information for the classification task while effectively leveraging both labeled and unlabeled data. Each time series in the latent space consists of S data points, where each instance in the sequence is a vector of size d', such that

$$z_s = [z_{s1}, \ldots, z_{sd'}] \in \mathbb{R}^{d'}, \quad \forall s \in \{1, \ldots, S\}.$$

The representation Z is learned such that it optimally captures the structure of X while aligning with the labels in $\mathscr{D}_l$ and facilitating meaningful clustering or self-supervised objectives for the unlabeled data in $\mathscr{D}_u$.

5.2 A Graphical Model Perspective

The IB principle states that random variables Y, X, Z form a Markov chain. In the current setting, it should be noted that X and Z are time series of different lengths. This leads to the graphical model depicted in Fig. 5.1.

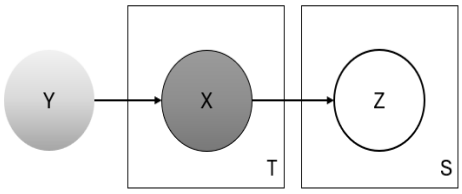


Figure 5.1: A graphical model illustration of the SS-GP-VIB model. Y, X, Z form a Markov chain, while X and Z are time series of lengths T and S, reps, and Y is partially observed.

The graphical model perspective suggests that Z_n and y_n are d-separated given X_n namely y_n and Z_n are independent when y_n observed. Hence, the SS-GP-VIB model architecture consists of two parts:

- The encoder (inference model): given the input X_n the encoder models the posterior distribution $p(Z_n|X_n)$.
- The clasifaier (prediction model): given the latent representation the classifier models $p(y_n|Z_n)$ when y_n is observed.
- The decoder (reconstruction model): given the latent representation the decoder models $p(X_n|Z_n)$ when y_n is not observed.

The encoder, the decoder, and the clasifaier are implemented using deep neural networks.

5.3 The Classifier

The classifier $q(y_n|Z_n)$ is modeled as a multinomial distribution, conditioned on the latent representation Z_n, such that:

$$q(y_n|Z_n) = q(y_n|g_\theta(Z_n))$$

where $g_\theta(\cdot)$ is a nonlinear function with parameters θ that outputs the logits of the multinomial distribution. The log-likelihood of the classifier is expressed as:

$$\log q\left(y_n | g_\theta\left(z_{1:T}\right)\right)$$

which corresponds to the cross-entropy loss function for classification. The function $g_\theta(\cdot)$ takes the latent time series Z_n and flattens it into a single vector, which is then passed to a Multi-Layer Perceptron. The output of the Multi-Layer Perceptron represents the logits of the multinomial distribution, from which the probabilities of the classes are computed.

5.4 The Decoder

The decoder in this semi-supervised framework reconstructs the original time series X_n from the latent representation Z_n, and is modeled as $p(X_n|Z_n)$. The function $h_\phi(Z_n)$, parameterized by ϕ, maps the latent variable $Z_n \in \mathbb{R}^{S\times d'}$ to the reconstructed time series $\hat{X}_n \in \mathbb{R}^{T\times d}$, where T is the length of the time series and d is the feature dimension. The decoder is typically composed of a series of MLPs that progressively transform the latent space representation back to the original data space. The objective is to minimize the reconstruction error, which is quantified by the negative log-likelihood:

$$\mathscr{L}_{\text{rec}} = -\mathbb{E}_{q(Z_n|X_n)}\left[\log p(X_n|Z_n)\right]$$

This loss function drives the model to reconstruct X_n as accurately as possible, ensuring that the latent representation Z_n captures the essential temporal and feature-based structures of the original time series.

By minimizing $\mathscr{L}_{\text{rec}}$, the decoder encourages the learning of a latent space that preserves the important information from X_n, even in the absence of labeled data. This allows the model to effectively utilize unlabeled data, regularizing the latent space in a way that improves both reconstruction and the ability to generalize to other downstream tasks.

5.5 Semi Supervised IB Objective Lower Bound

In the semi-supervised framework, the IB objective of the GP-VIB model is modified to account for both labeled and unlabeled data. The objective function is as follows:

$$\begin{aligned} L = &\frac{1}{N_l}\sum_{n=1}^{N_l}\left[\mathbb{E}_{p\left(Z_n|f_\psi(X_n)\right)}\log q\left(y_n|g_\theta\left(Z_n\right)\right) - \beta KL\left[p\left(Z_n|f_\psi\left(X_n\right)\right)||r\left(Z_n\right)\right]\right] \\ &+\frac{1}{N_u}\sum_{n=1}^{N_u}\left[\gamma\mathbb{E}_{p(Z_n|f_\psi(X_n))}\log p(X_n|h_\phi(Z_n)) \text{ - } \delta KL\left[p\left(Z_n|f_\psi\left(X_n\right)\right)||r\left(Z_n\right)\right]\right] \end{aligned} \tag{5.1}$$

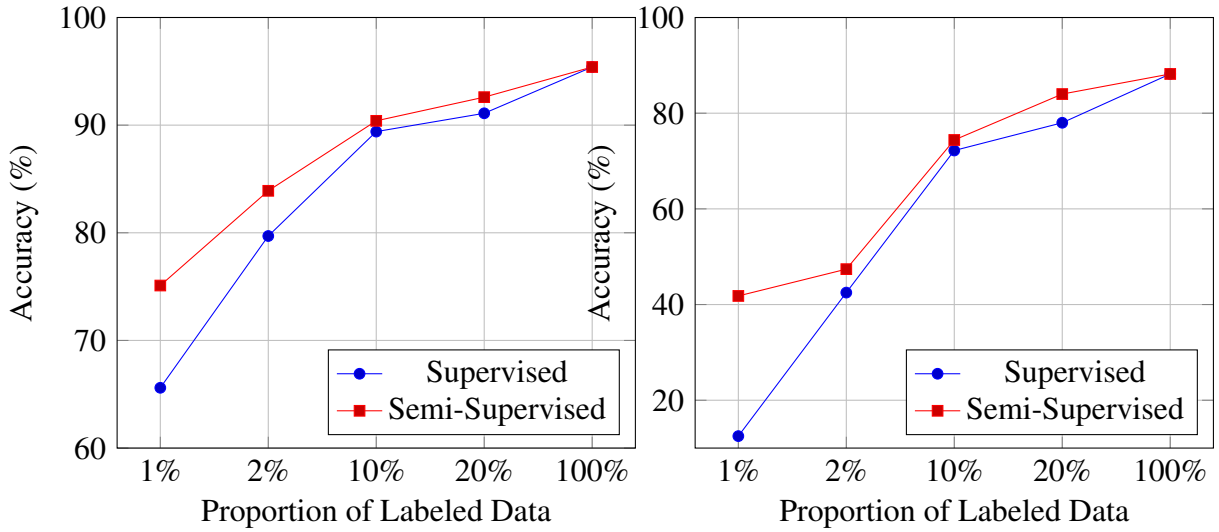
Here, N_l and N_u represent the number of labeled and unlabeled instances, respectively. The first term represents the supervised learning part, where the model maximizes the likelihood of the labels given the latent representations Z_n. The second term is the reconstruction loss for the unlabeled data, where the decoder reconstructs the input time series X_n given the latent representation Z_n, encouraging the model to learn useful features from the unlabeled data. For both terms we also added the KL regularization term, which enforces the variational distribution $p(Z_n|X_n)$ to be close to the prior $r(Z_n)$.

Semi Supervised Experiments

In this section, we present a comprehensive evaluation of our semi supervised approach through two experimental settings. First, we compare different the GP-VIB model against the SS-GP-VIB model on different proportions of labeled data over the HM-nist dataset to assess their representation learning capabilities. In addition, as our model showed robustness to missing data, we include also an experiment over different proportions of labeled data over missing data, Next, we evaluated our model over the Chronic Pain from Cellphone Usage Data dataset.

6.1 Comparison of GP-VIB and SS-GP-VIB

To ensure comparability across different data settings, we used the same architecture for all experiments. For each proportion of labeled data and the semi-supervision flag, we conducted a hyperparameter search and selected the best-performing model. The best model for each category was then trained across five different random seeds.



(a) Fully Observed Data

(b) Missing Data

Figure 6.1: Accuracy comparison for Fully Observed and Missing Data settings.

As shown in Figure 6.1, accuracy declines as the proportion of labeled data decreases. However, semi-supervised learning mitigates this decline, particularly at extremely low

data regimes (1% and 2% labeled data). In these cases, the gap between supervised and semi-supervised models is substantial, especially under the missing data setting.

Table 6.1: Performance metrics for different dataset sizes over HMNIST, under Fully Observed and Missing Data settings. Values are mean ± standard deviation, scaled by 100.

Proportion of	**Fully Observed Data**			**Missing Data**		
Labeled Data	Accuracy (%)	AUROC (%)	AUPRC (%)	Accuracy (%)	AUROC (%)	AUPRC (%)
Supervised (100%)	95.4 ± 0.1	99.8 ± 0.03	98.9 ± 0.1	88.2 ± 0.6	99.2 ± 0.0	95.2 ± 0.2
Semi-supervised (20%)	92.6 ± 0.6	99.6 ± 0.07	97.6 ± 0.4	84.0 ± 0.5	98.4 ± 0.1	91.1 ± 0.4
Supervised (20%)	91.1 ± 1.0	99.5 ± 0.1	96.9 ± 0.4	78.0 ± 0.6	97.1 ± 0.1	85.2 ± 0.5
Semi-supervised (10%)	90.4 ± 1.0	99.4 ± 0.07	96.5 ± 0.3	74.4 ± 2.3	96.3 ± 0.6	81.1 ± 2.5
Supervised (10%)	89.4 ± 0.6	99.2 ± 0.06	95.7 ± 0.3	72.2 ± 2.5	95.7 ± 0.6	78.7 ± 2.4
Semi-supervised (2%)	83.9 ± 0.6	98.2 ± 0.1	90.8 ± 0.5	47.4 ± 1.8	85.7 ± 0.7	46.7 ± 1.9
Supervised (2%)	79.7 ± 1.1	97.4 ± 0.3	86.8 ± 1.2	42.5 ± 1.3	83.7 ± 0.8	40.9 ± 1.1
Semi-supervised (1%)	75.1 ± 1.7	96.2 ± 0.5	82.3 ± 2.0	41.8 ± 2.1	81.9 ± 1.4	40.9 ± 2.7
Supervised (1%)	65.6 ± 2.5	93.4 ± 0.8	70.0 ± 3.1	12.5 ± 3.1	57.2 ± 5.4	15.4 ± 1.4

For 1% labeled data, semi-supervised learning improves accuracy by **9.5 percentage points** in the fully observed setting and **29.3 percentage points** in the missing data setting. Similarly, for 2% labeled data, it improves accuracy by **4.2 percentage points** in the fully observed setting and **4.9 percentage points** in the missing data setting. These results highlight the advantages of incorporating unlabeled data, particularly in settings where labeled data is scarce. At higher labeled data proportions (e.g., 10% and 20%), the performance gap between supervised and semi-supervised models narrows, suggesting that the benefit of semi-supervised learning diminishes as more labeled data becomes available. However, even in these settings, the semi-supervised method consistently outperforms its fully supervised counterpart. These observations indicate the potential of semi-supervised learning to enhance performance, particularly in low-data scenarios.

6.2 Predicting Health States of Patients with Chronic Pain from Cellphone Usage Data

This experiment was carried out as part of a broader study con- ducted by the Data Science Institute at Reichman University, which was responsible for establishing the dataset, including data collection, labeling, and preprocessing [99]. The SS-GP-VIB model was evaluated against traditional machine learning benchmarks.

6.2.1 Dataset Description

Chronic pain exerts an enormous burden on society, healthcare systems, and individuals, affecting more than 30% of people worldwide [17]. It significantly disrupts patient routines and reduces their quality of life. Lately, machine learning solutions and data-driven approaches have been used in chronic pain research in attempts to ease the burden [91]. For this purpose, a clinical study was carried out a six-month observational at a local pain clinic involving participants exclusively diagnosed with chronic pain. The

experimental setup involved continuously gathering and analyzing cellular usage data (logs of app usage and cellphone events) from a cohort of 91 participants.

In addition, patients were asked to fill out two types of health-related quality of life (HRQL) questionnaires to monitor their health states. With the guidance of the research assistant, they manually filled out the SF-36 questionnaire [114] three times during the study - in the beginning, middle, and end. On their own time and ideally every week, they were asked to digitally fill out the EQ-5D-5L questionnaire [89], which is shorter and clinically validated for mobile devices [57].

The information provided by the patients in their EQ-5D reports served as the basis for the supervised prediction task. Based on the relevant value set [96], the five dimensions of the EQ-5D questionnaire (mobility, self-care, usual activities, pain/discomfort, and anxiety/depression) can be convert into a continuous health index. Then, a binary classification was performed on the joint HRQL measure. Using the accepted cutoff of 0, indicating a health state as bad as death [24], The states can be divide into worse-than-death health states (0) and otherwise (1). In this binary setting, we aimed to determine whether a patient's health state was so severe that they considered it to be worse than death.

The dataset consists of both labeled and unlabeled data. Each instance represents a week in a participant's life, at the end of which the participant completed the EQ-5D test, yielding a binary label. Since participants were involved in the clinical trial for six months, the dataset contains a substantial number of unlabeled weeks. Each week comprises 181 hours, spanning seven full days and an additional 13 hours. Every hour corresponds to the total time a participant spent using applications from different categories within that interval. These application categories were consolidated into 17 groups, resulting in each instance having a size of 181×17. The total number of labeled instances is provided in Table 9.1, along with an additional 436 unlabeled instances. Due to the nature of application usage, where not all categories are used continuously, the resulting dataset is both high-dimensional and sparse, with a limited number of labeled samples.

For detailed information on the experimental setup and data preprocessing methods, please refer to Appendix 9 and Appendix 10, respectively.

6.2.2 SS-GP-VIB Results

Following the recommendations of classical machine learning evaluation [99], we conducted separate evaluations for male and female participants. We tested the same pipeline described in Appendix 10.0.1. Additionally, we included an evaluation using the raw data without applying the preprocessing pipeline to the sparse dataset. For each setting, we evaluated the GP-VIB model using only the labeled data and the SS-GP-VIB model utilizing the full dataset. In each case, we performed a hyperparameter search, followed by three different runs with varying random seeds based on the best-found configuration. Table 6.2 presents the evaluation results across the different settings. Additionally, we included results from a Naive baseline model that mimics common clinical practice by predicting each label based on the participant's most recent known state, with initial predictions determined by the overall label distribution.

Table 6.2: Best Classification Results per Gender and Pre-clustering Transformation

Setting				F1-Score			Class 0		Class 1		AUC
Gender	Pipeline	SSL	K	Acc.	M. avg.	W. avg.	Prec.	Recall	Prec.	Recall	ROC
Male	Raw	False	-	86.4±3.4	78.4±5.5	85.8±3.5	89.2±3.6	94.2±5.5	77.8±19.2	59.0±16.0	77.3±5.6
	Raw	True	-	84.2±1.0	74.5±1.0	83.3±0.7	87.2±0.8	93.5±2.2	69.5±5.7	51.3±4.4	76.0±4.1
	Standard	False	8	91.0±2.0	86.6±2.7	90.9±1.9	93.6±1.9	94.9±3.3	82.0±8.8	76.9±7.7	87.5±4.4
	Standard	True	24	**92.1±2.0**	**88.1±2.6**	**92.0±1.8**	93.7±1.8	96.4±3.3	87.2±11.8	76.9±7.7	**92.1±3.3**
	PCT	False	6	83.6±5.4	66.3±15.6	79.8±8.5	84.2±6.4	**97.8±2.2**	82.8±16.7	33.3±31.1	73.8±6.9
	PCT	True	16	89.3±5.2	85.1±5.9	89.5±4.6	94.3±3.0	92.0±8.8	78.3±20.2	79.5±11.8	85.2±5.6
	DCT	False	16	85.3±6.8	73.2±16.1	83.2±9.2	86.6±6.0	96.4±1.3	72.6±20.5	46.2±27.7	81.5±12.6
	DCT	True	6	90.4±4.3	85.0±5.9	90.0±4.1	91.8±2.9	96.4±6.3	**88.9±19.2**	69.2±13.3	81.8±5.3
	PCT+DCT	False	16	88.1±5.1	83.9±6.2	88.5±4.7	**94.6±1.5**	89.9±5.5	70.7±13.0	**82.1±4.4**	89.3±4.1
	PCT+DCT	True	16	90.4±4.3	84.9±7.1	90.0±4.6	91.8±3.3	96.4±3.3	84.9±13.1	69.2±13.3	82.7±5.4
	Naive	False	-	86.4	81.3	86.8	93.2	89.1	66.7	76.9	50.0
Female	Raw	False	-	68.4±8.9	52.1±20.3	59.1±16.2	68.1±8.7	**97.2±4.9**	60.0±52.9	20.2±32.0	49.9±21.2
	Raw	True	-	74.2±10.2	67.4±15.8	71.1±13.3	74.4±10.4	92.2±1.2	73.3±10.8	44.0±28.9	67.9±14.1
	Standard	False	8	79.6±0.8	78.4±0.7	79.6±0.7	84.7±0.2	82.3±1.2	71.6±1.4	75.0±0.0	78.7±3.4
	Standard	True	8	79.6±1.5	78.4±1.5	79.6±1.5	84.7±0.4	82.3±2.5	71.7±2.9	75.0±0.0	78.4±1.5
	PCT	False	24	82.2±0.8	81.4±0.7	82.4±0.7	**88.5±0.2**	82.3±1.2	73.4±1.3	**82.1±0.0**	**81.9±2.6**
	PCT	True	64	80.9±1.5	79.5±2.1	80.8±1.7	85.3±4.0	84.4±4.4	74.5±3.7	75.0±9.4	79.1±3.9
	DCT	False	16	75.6±5.4	72.6±7.7	74.8±6.5	78.9±7.1	84.4±5.4	69.8±4.8	60.7±18.9	73.3±4.7
	DCT	True	16	**83.6±1.5**	**81.7±1.4**	**83.2±1.4**	83.3±0.7	92.2±3.3	**84.4±5.5**	69.0±2.1	80.6±3.5
	PCT+DCT	False	128	81.8±1.5	80.3±1.6	81.7±1.5	84.2±1.1	87.2±2.1	77.3±3.0	72.6±2.1	78.1±4.3
	PCT+DCT	True	8	82.2±5.0	81.3±4.9	82.3±4.8	87.4±1.2	83.7±8.6	75.5±10.9	79.8±2.1	81.2±2.9
	Naive	False	-	78.7	77.5	78.8	84.4	80.9	70.0	75.0	50.0

Acc. = Accuracy; M. avg. = Macro average; W. avg. = Weighted average; Prec. = Precision. In **bold** we mark the best results and in underline we mark the second best result.

Male participants For male subjects, the best performance was obtained with the Standard pipeline using SSL, reaching 92.1% accuracy, 88.1% macro-average F1, and 92.1% AUC-ROC. This substantially outperformed the naive model (86.4% accuracy, 81.3% macro-average F1) and exceeded the supervised Standard pipeline, which achieved 91.0% accuracy with lower macro-average F1 (86.6%) and AUC-ROC (87.5%). Comparing SSL with supervised settings across other pipelines shows consistently positive effects: Raw data showed a slight decrease in performance with SSL (84.2% vs 86.4% accuracy), while PCT demonstrated significant improvement (89.3% vs 83.6% accuracy), DCT showed substantial gains (90.4% vs 85.3% accuracy), and PCT+DCT improved modestly (90.4% vs 88.1% accuracy). Overall, SSL provided benefits across most preprocessing methods for males, with the Standard pipeline achieving the highest absolute performance, indicating that semi-supervised learning enhances feature representation effectively across different transformation approaches.

Female participants For female subjects, the best results were achieved with the DCT pipeline using SSL, which reached 83.6% accuracy, 81.7% macro-average F1, and 80.6% AUC-ROC. This surpassed the naive model (78.7% accuracy, 77.5% macro-average F1) and significantly outperformed the supervised DCT pipeline (75.6% accuracy, 72.6% macro-average F1, 73.3% AUC-ROC), representing an 8.0% accuracy improvement. Comparing SSL to supervised settings across other pipelines reveals mixed effects: Raw data showed improvement with SSL (74.2% vs 68.4% accuracy), Standard pipeline remained stable (79.6% for both), PCT showed a slight decline (80.9% vs 82.2% accuracy), while PCT+DCT showed minimal improvement (82.2% vs 81.8% accuracy). The DCT pipeline demonstrated the most dramatic SSL benefit, with substantial improvements across all metrics.

Comparing with Two-Steps Classification We compare our best results against the best results from the Two-Step pipeline reported in [99], as shown in Table 10.1.

Table 6.3: Best Classification Results per Gender

	Setting			F1-Score			Class 0		Class 1		AUC
Gender	Pipeline	K	Model	Acc.	M. avg.	W. avg.	Prec.	Recall	Prec.	Recall	ROC
Male	PCT	32	kNN	89.8	83.2	89.1	88.9	61.5	90.0	97.8	79.7
	Standard	24	SS-GP-VIB	92.1	88.1	91.9	93.7	96.4	87.2	76.9	92.1
Female	PCT	32	RF	82.7	80.7	82.4	79.2	70.4	84.3	89.6	80.0
	DCT	16	SS-GP-VIB	83.6	81.7	83.2	83.3	92.2	84.4	69.0	80.6

Acc. = Accuracy; M. avg. = Macro average; W. avg. = Weighted average; Prec. = Precision.

Predicting the health states of patients with chronic pain from cellphone usage data is a challenging task, particularly due to the sparsity of the time series and the limited availability of labeled instances compared to unlabeled data. Our GP-VIB model, together with its SSL extension, effectively addressed this challenge. The SSL variant consistently outperformed classical machine learning baselines and the Two-Step pipeline across both genders, demonstrating significant improvements in prediction accuracy and robustness. For male participants, the Standard pipeline with SSL achieved the highest performance (92.1% accuracy, 88.1% F1-score, 92.1% AUC-ROC), while for female participants, the DCT pipeline with SSL yielded the best results (83.6% accuracy, 81.7% F1-score, 80.6% AUC-ROC). Notably, SSL showed consistent benefits for males across all preprocessing methods, while for females, the benefits were more pronounced with DCT preprocessing. These findings highlight the advantage of leveraging both labeled and unlabeled data, demonstrating that semi-supervised learning with GP-VIB provides robust improvements over supervised approaches in the context of chronic pain health state prediction, with the effectiveness varying by gender and preprocessing method.

Discussion and Conclusions

In this thesis, we introduce GP-VIB and its semi-supervised extension, a novel model specifically designed for time series classification. GP-VIB implements a Variational Information Bottleneck architecture, enhanced with a GP representation of the latent space. Our model achieves performance on par with SOTA models on benchmark TSC datasets, highlighting its potential for advancing the field.

While VAEs typically learn a latent space optimized for reconstructing input data, VIB is explicitly designed to extract a discriminative latent space that captures the most relevant information for classification. By incorporating a GP prior and employing structured inference to approximate a variational posterior, our approach enables the latent space to preserve time dependent dynamics, which is crucial for time series data.

The primary objective of any VIB model is to construct a latent space that maximizes mutual information with the target labels while minimizing mutual information with the input. It is important to emphasize that, by definition, VIB is not inherently a classifier. However, our experiments demonstrate that GP-VIB exhibits strong classification performance despite being derived from an information-theoretic objective rather than a direct classification framework.[1]

A common challenge in real-world applications is the scarcity of labeled data relative to the total collected dataset. To address this, semi-supervised methods are crucial. Motivated by the goal of predicting health states in patients with chronic pain based on cellphone usage data, we developed a semi-supervised variant of GP-VIB. While our model demonstrated robustness and promising results on the HMNIST dataset, further refinements are necessary to enhance its performance in more complex real-world scenarios.

This study opens several avenues for future research. Initially, we hypothesized that adding a decoder for the unlabeled data would enrich the latent space with features relevant for reconstruction. While some of these features may correlate with classification, many do not, potentially limiting performance gains. Future work could explore combining GP-VIB with contrastive learning techniques, which aim to create a more discriminative latent space aligned with classification tasks. Our preliminary trials did not yield significant improvements, but alternative pretext tasks from contrastive learning remain unexplored and could better integrate with GP-VIB's framework.

[1] The goal of VIB is to construct a latent space Z that maximizes the mutual information with the target labels Y, while minimizing the mutual information of Z and the input X.

References

[1] Alexander A Alemi et al. “Deep Variational Information Bottleneck”. In: (2016).

[2] Massih-Reza Amini, Nicolas Usunier, and Cyril Goutte. “A Survey on Self-Training Methods for Semi-Supervised Learning”. In: *Journal of Machine Learning Research*. 2025.

[3] Anthony Bagnall et al. “The UEA multivariate time series classification archive, 2018”. In: *arXiv preprint arXiv:1811.00075* (2018).

[4] Robert Bamler and Stephan Mandt. “Structured black box variational inference for latent time series models”. In: *arXiv preprint arXiv:1707.01069* (2017).

[5] Alessio Benavoli, Giorgio Corani, and Francesca Mangili. “Should we really use post-hoc tests based on mean-ranks?” In: *The Journal of Machine Learning Research* 17.1 (2016), pp. 152–161.

[6] David Berthelot et al. “Mixmatch: A holistic approach to semi-supervised learning”. In: *Advances in neural information processing systems* 32 (2019).

[7] David M. Blei, Alp Kucukelbir, and Jon D. McAuliffe. “Variational Inference: A Review for Statisticians”. In: *Journal of the American Statistical Association* 112.518 (Apr. 2017), pp. 859–877. ISSN: 1537-274X. DOI: 10.1080/01621459.2017.1285773. URL: http://dx.doi.org/10.1080/01621459.2017.1285773.

[8] Nestor Cabello et al. “Fast, accurate and interpretable time series classification through randomization”. In: *arXiv preprint arXiv:2105.14876* (2021).

[9] Wei Cao et al. “Brits: Bidirectional recurrent imputation for time series”. In: *Advances in neural information processing systems* 31 (2018).

[10] Zhengping Che et al. “Recurrent neural networks for multivariate time series with missing values”. In: *Scientific reports* 8.1 (2018), pp. 1–12.

[11] Siyuan Chen. “Modeling Aging Trajectories and Urban-rural Disparities in Chronic Pain among Middle-aged and Older Chinese: Evidence from the Generalized Estimating Equations (GEE) and Generalized Linear Mixed Models (GLMMs)”. PhD thesis. Ghent University, 2024.

[12] Ting Chen et al. “A simple framework for contrastive learning of visual representations”. In: *International conference on machine learning*. PmLR. 2020, pp. 1597–1607.

[13] Ranak Roy Chowdhury et al. “Tarnet: Task-aware reconstruction for time-series transformer”. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 212–220.

[14] Ranak Roy Chowdhury et al. “Tarnet: Task-aware reconstruction for time-series transformer”. In: *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 2022, pp. 212–220.

[15] Maximilian Christ et al. “Time series feature extraction on basis of scalable hypothesis tests (tsfresh–a python package)”. In: *Neurocomputing* 307 (2018), pp. 72–77.

[16] Junyoung Chung et al. “Empirical evaluation of gated recurrent neural networks on sequence modeling”. In: *arXiv preprint arXiv:1412.3555* (2014).

[17] Steven P Cohen, Lene Vase, and William M Hooten. “Chronic pain: an update on burden, best practices, and new advances”. In: *The Lancet* 397.10289 (2021), pp. 2082–2097.

[18] Zhicheng Cui, Wenlin Chen, and Yixin Chen. “Multi-scale convolutional neural networks for time series classification”. In: *arXiv preprint arXiv:1603.06995* (2016).

[19] Roy De Maesschalck, Delphine Jouan-Rimbaud, and Désiré L Massart. “The mahalanobis distance”. In: *Chemometrics and intelligent laboratory systems* 50.1 (2000), pp. 1–18.

[20] Angus Dempster, François Petitjean, and Geoffrey I Webb. “ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels”. In: *Data Mining and Knowledge Discovery* 34.5 (2020), pp. 1454–1495.

[21] Angus Dempster, Daniel F Schmidt, and Geoffrey I Webb. “Hydra: Competing convolutional kernels for fast and accurate time series classification”. In: *Data Mining and Knowledge Discovery* (2023), pp. 1–27.

[22] Janez Demšar. “Statistical comparisons of classifiers over multiple data sets”. In: *The Journal of Machine learning research* 7 (2006), pp. 1–30.

[23] Don Dennis et al. “Shallow rnn: accurate time-series classification on resource constrained devices”. In: *Advances in neural information processing systems* 32 (2019).

[24] Nancy Devlin, David Parkin, and Bas Janssen. *Methods for analysing and reporting EQ-5D data*. Springer Nature, 2020.

[25] Nancy Devlin et al. “An introduction to EQ-5D instruments and their applications”. In: *Methods for analysing and reporting EQ-5D data* (2020), pp. 1–22.

[26] Clare Dominick, Fiona Blyth, and Michael Nicholas. “Patterns of chronic pain in the New Zealand population”. In: *The New Zealand Medical Journal (Online)* 124.1337 (2011).

[27] Emadeldeen Eldele et al. “Time-Series Representation Learning via Temporal and Contextual Contrasting”. In: *International Joint Conference on Artificial Intelligence (IJCAI)*. 2021.

[28] Emadeldeen Eldele et al. “Time-series representation learning via temporal and contextual contrasting”. In: *arXiv preprint arXiv:2106.14112* (2021).

[29] Thomas E Elliott, Colleen M Renier, and Jeanette A Palcher. “Chronic pain, depression, and quality of life: correlations and predictive value of the SF-36”. In: *Pain medicine* 4.4 (2003), pp. 331–339.

[30] Haoyi Fan et al. “Semi-Supervised Time Series Classification by Temporal Relation Prediction”. In: *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*. 2021.

[31] Santiago Fernández, Alex Graves, and Jürgen Schmidhuber. “Sequence labelling in structured domains with hierarchical recurrent neural networks”. In: *Proceedings of the 20th International Joint Conference on Artificial Intelligence, IJCAI 2007*. 2007.

[32] Vincent Fortuin et al. “Gp-vae: Deep probabilistic time series imputation”. In: *International conference on artificial intelligence and statistics*. PMLR. 2020, pp. 1651–1661.

[33] Navid Mohammadi Foumani et al. “Improving position encoding of transformers for multivariate time series classification”. In: *Data mining and knowledge discovery* 38.1 (2024), pp. 22–48.

[34] Navid Mohammadi Foumani et al. “Series2vec: similarity-based self-supervised representation learning for time series classification”. In: *Data Mining and Knowledge Discovery* 38.4 (2024), pp. 2520–2544.

[35] Jean-Yves Franceschi, Aymeric Dieuleveut, and Martin Jaggi. “Unsupervised scalable representation learning for multivariate time series”. In: *Advances in neural information processing systems* 32 (2019).

[36] Ge Gao et al. “A reinforcement learning-informed pattern mining framework for multivariate time series classification”. In: *In the Proceeding of 31th International Joint Conference on Artificial Intelligence (IJCAI-22)*. 2022.

[37] Salvador Garcia and Francisco Herrera. “An Extension on" Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons.” In: *Journal of machine learning research* 9.12 (2008).

[38] Katherine Godfrey. “Simple linear regression in medical research”. In: *New England Journal of Medicine* 313.26 (1985), pp. 1629–1636.

[39] Gene H Golub and Christian Reinsch. “Singular value decomposition and least squares solutions”. In: *Linear algebra*. Springer, 1971, pp. 134–151.

[40] Jann Goschenhofer et al. “Deep Semi-supervised Learning for Time Series Classification”. In: *arXiv preprint*. 2022.

[41] The EuroQol Group. “EuroQol-a new facility for the measurement of health-related quality of life”. In: *Health policy* 16.3 (1990), pp. 199–208.

[42] Antoine Guillaume, Christel Vrain, and Wael Elloumi. “Random dilated shapelet transform: A new approach for time series shapelets”. In: *International Conference on Pattern Recognition and Artificial Intelligence*. Springer. 2022, pp. 653–664.

[43] Yifan Hao and Huiping Cao. “A new attention mechanism to classify multivariate time series”. In: *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*. 2020.

[44] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[45] Michiel Hermans and Benjamin Schrauwen. “Training and analysing deep recurrent neural networks”. In: *Advances in neural information processing systems* 26 (2013).

[46] Mónica Hernández Alava, Steve Pudney, and Allan Wailoo. “Estimating the relationship between EQ-5D-5L and EQ-5D-3L: results from a UK population study”. In: *Pharmacoeconomics* 41.2 (2023), pp. 199–207.

[47] Sepp Hochreiter and Jürgen Schmidhuber. “Long short-term memory”. In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[48] Max Horn et al. “Set functions for time series”. In: *International Conference on Machine Learning*. PMLR. 2020, pp. 4353–4363.

[49] Tsung-Yu Hsieh et al. “Explainable multivariate time series classification: a deep neural network which learns to attend to important variables as well as time intervals”. In: *Proceedings of the 14th ACM international conference on web search and data mining*. 2021, pp. 607–615.

[50] Jie Hu, Li Shen, and Gang Sun. “Squeeze-and-excitation networks”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 7132–7141.

[51] Michael Hüsken and Peter Stagge. “Recurrent neural networks for time series classification”. In: *Neurocomputing* 50 (2003), pp. 223–235.

[52] Aparecida Mari Iguti, Margareth Guimarães, and Marilisa Berti Azevedo Barros. “Health-related quality of life (SF-36) in back pain: a population-based study, Campinas, São Paulo State, Brazil”. In: *Cadernos de saude publica* 37 (2021), e00206019.

[53] Hassan Ismail Fawaz et al. “Inceptiontime: Finding alexnet for time series classification”. In: *Data Mining and Knowledge Discovery* 34.6 (2020), pp. 1936–1962.

[54] Ali Ismail-Fawaz et al. “LITE: Light Inception with boosTing tEchniques for Time Series Classification”. In: *2023 IEEE 10th International Conference on Data Science and Advanced Analytics (DSAA)*. 2023, pp. 1–10. DOI: 10.1109/DSAA60987.2023.10302569.

[55] Melinda Járomi et al. “Assessment of health-related quality of life and patient’s knowledge in chronic non-specific low back pain”. In: *BMC Public Health* 21 (2021), pp. 1–8.

[56] Alistair E.W. Johnson et al. “MIMIC-III, a freely accessible critical care database”. In: *Scientific Data* 3.1 (May 2016), p. 160035. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.35. URL: https://doi.org/10.1038/sdata.2016.35.

[57] Regina JM Kamstra et al. “Validation of the Mobile App Version of the EQ-5D-5L Quality of Life Questionnaire Against the Gold Standard Paper-Based Version: Randomized Crossover Study”. In: *JMIR Formative Research* 6.8 (2022), e37303.

[58] Fazle Karim et al. “Multivariate LSTM-FCNs for time series classification”. In: *Neural networks* 116 (2019), pp. 237–245.

[59] Mohammed Khalilia, Sounak Chakraborty, and Mihail Popescu. “Predicting disease risks from highly imbalanced data using random forest”. In: *BMC medical informatics and decision making* 11.1 (2011), pp. 1–13.

[60] Diederik P Kingma and Max Welling. “Auto-Encoding Variational Bayes”. In: *stat* 1050 (2014), p. 1.

[61] Demetres Kostas, Stephane Aroca-Ouellette, and Frank Rudzicz. “BENDR: Using transformers and a contrastive self-supervised learning task to learn from massive amounts of EEG data”. In: *Frontiers in Human Neuroscience* 15 (2021), p. 653659.

[62] Rahul G Krishnan, Uri Shalit, and David Sontag. “Deep Kalman Filters”. In: *arXiv e-prints* (2015), arXiv–1511.

[63] Martin Längkvist, Lars Karlsson, and Amy Loutfi. “A review of unsupervised feature learning and deep learning for time-series modeling”. In: *Pattern recognition letters* 42 (2014), pp. 11–24.

[64] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. “Data augmentation for time series classification using convolutional neural networks”. In: *ECML/PKDD workshop on advanced analytics and learning on temporal data*. 2016.

[65] Dong-Hyun Lee. “Pseudo-label: The Simple and Efficient Semi-Supervised Learning Method for Deep Neural Networks”. In: *ICML Workshop on Challenges in Representation Learning* (2013). URL: `https://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.695.7473&rep=rep1&type=pdf`.

[66] Guozhong Li et al. “Shapenet: A shapelet-neural network approach for multivariate time series classification”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 35. 9. 2021, pp. 8375–8383.

[67] Yuhong Li, Xiaofan Zhang, and Deming Chen. “Csrnet: Dilated convolutional neural networks for understanding the highly congested scenes”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 1091–1100.

[68] Huang-Li Lin et al. “Bodily pain and vitality are the key factors in the disability of chronic low back pain patients under Short Form 36 base study: a five-year cohort study”. In: *Health and Quality of Life Outcomes* 22.1 (2024), p. 88.

[69] Sangdi Lin and George C Runger. “GCRNN: Group-constrained convolutional recurrent neural network”. In: *IEEE transactions on neural networks and learning systems* 29.10 (2017), pp. 4709–4718.

[70] Zachary C Lipton and Subarna Tripathi. “Precise recovery of latent vectors from generative adversarial networks”. In: *arXiv preprint arXiv:1702.04782* (2017).

[71] Zhen Liu et al. “Temporal-Frequency Co-training for Time Series Semi-supervised Learning”. In: *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI)*. 2023.

[72] Ziyu Liu et al. “Self-Supervised Learning for Time Series: Contrastive or Generative?” In: *arXiv preprint*. 2023.

[73] Benjamin Lucas et al. “Proximity forest: an effective and scalable distance-based classifier for time series”. In: *Data Mining and Knowledge Discovery* 33.3 (2019), pp. 607–635.

[74] Pankaj Malhotra et al. “TimeNet: Pre-trained deep recurrent neural network for time series classification”. In: *arXiv preprint arXiv:1706.08838* (2017).

[75] Leland McInnes, John Healy, and James Melville. “Umap: Uniform manifold approximation and projection for dimension reduction”. In: *arXiv preprint arXiv:1802.03426* (2018).

[76] Matthew Middlehurst and Anthony Bagnall. “The freshprince: A simple transformation based pipeline time series classifier”. In: *International Conference on Pattern Recognition and Artificial Intelligence*. Springer. 2022, pp. 150–161.

[77] Matthew Middlehurst, Patrick Schäfer, and Anthony Bagnall. “Bake off redux: a review and experimental evaluation of recent time series classification algorithms”. In: *arXiv preprint arXiv:2304.13029* (2023).

[78] Matthew Middlehurst et al. “HIVE-COTE 2.0: a new meta ensemble for time series classification”. In: *Machine Learning* 110.11-12 (2021), pp. 3211–3243.

[79] Matthew Middlehurst et al. “The temporal dictionary ensemble (TDE) classifier for time series classification”. In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2020, Ghent, Belgium, September 14–18, 2020, Proceedings, Part I*. Springer. 2021, pp. 660–676.

[80] Takeru Miyato et al. “Virtual adversarial training: a regularization method for supervised and semi-supervised learning”. In: *IEEE transactions on pattern analysis and machine intelligence* 41.8 (2018), pp. 1979–1993.

[81] Ronald Mutegeki and Dong Seog Han. “A CNN-LSTM approach to human activity recognition”. In: *2020 international conference on artificial intelligence in information and communication (ICAIIC)*. IEEE. 2020, pp. 362–366.

[82] Alfredo Nazabal et al. “Handling incomplete heterogeneous data using vaes”. In: *Pattern Recognition* 107 (2020), p. 107501.

[83] Daniel Neil, Michael Pfeiffer, and Shih-Chii Liu. “Phased lstm: Accelerating recurrent network training for long or event-based sequences”. In: *Advances in neural information processing systems* 29 (2016).

[84] Minh Nhut Nguyen, Phyo Phyo San, and Shonali Krishnaswamy. “Deep convolutional neural networks on multichannel time series for human activity recognition.” In.

[85] MJ Oemar. *Basic information on how to use the EQ-5D-5L instrument*. Tech. rep. EQ-5D-5L User Guide Version 20, 2013.

[86] Tianxin Pan et al. “A comparison of PROPr and EQ-5D-5L value sets”. In: *Pharmacoeconomics* (2022), pp. 1–11.

[87] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “On the difficulty of training recurrent neural networks”. In: *International conference on machine learning*. Pmlr. 2013, pp. 1310–1318.

[88] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. “Understanding the exploding gradient problem”. In: *CoRR, abs/1211.5063* 2.417 (2012), p. 1.

[89] Rosalind Rabin and Frank de Charro. “EQ-SD: a measure of health status from the EuroQol Group”. In: *Annals of medicine* 33.5 (2001), pp. 337–343.

[90] Thanawin Rakthanmanon et al. “Addressing big data time series: Mining trillions of time series subsequences under dynamic time warping”. In: *ACM Transactions on Knowledge Discovery from Data (TKDD)* 7.3 (2013), pp. 1–31.

[91] Alex Novaes Santana, Charles Novaes de Santana, and Pedro Montoya. “Chronic pain diagnosis using machine learning, questionnaires, and QST: a sensitivity experiment”. In: *Diagnostics* 10.11 (2020), p. 958.

[92] Patrick Schäfer and Ulf Leser. “WEASEL 2.0–A Random Dilated Dictionary Transform for Fast, Accurate and Memory Constrained Time Series Classification”. In: *arXiv preprint arXiv:2301.10194* (2023).

[93] Joan Serra, Santiago Pascual, and Alexandros Karatzoglou. “Towards a universal neural network encoder for time series”. In: *Artificial Intelligence Research and Development*. IOS Press, 2018, pp. 120–129.

[94] Satya Narayan Shukla and Benjamin M Marlin. “Interpolation-prediction networks for irregularly sampled time series”. In: *arXiv preprint arXiv:1909.07782* (2019).

[95] Ikaro Silva et al. “Predicting in-hospital mortality of icu patients: The physionet/computing in cardiology challenge 2012”. In: *2012 Computing in Cardiology*. IEEE. 2012, pp. 245–248.

[96] John Smith. “Citation omitted to preserve blindness”. In: (2022).

[97] Huan Song et al. “Attend and diagnose: Clinical time series analysis using attention models”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[98] Huan Song et al. “Attend and diagnose: Clinical time series analysis using attention models”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. 1. 2018.

[99] Maya Stemmer et al. *Predicting Chronic Pain State from Cellphone Usage Data*. Manuscript submitted for publication. 2025.

[100] Jingyu Sun, Susumu Takeuchi, and Ikuo Yamasaki. “Prototypical inception network with cross branch attention for time series classification”. In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–7.

[101] Christian Szegedy et al. “Going deeper with convolutions”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[102] Christian Szegedy et al. “Inception-v4, inception-resnet and the impact of residual connections on learning”. In: *Proceedings of the AAAI conference on artificial intelligence*. Vol. 31. 1. 2017.

[103] Chang Wei Tan et al. “MultiRocket: multiple pooling operators and transformations for fast and effective time series classification”. In: *Data Mining and Knowledge Discovery* 36.5 (2022), pp. 1623–1646.

[104] Yujin Tang et al. “Sequence-to-sequence model with attention for time series classification”. In: *2016 IEEE 16th International Conference on Data Mining Workshops (ICDMW)*. IEEE. 2016, pp. 503–510.

[105] Antti Tarvainen and Harri Valpola. “Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results”. In: *Advances in neural information processing systems* 30 (2017).

[106] Naftali Tishby, Fernando C. Pereira, and William Bialek. “The information bottleneck method”. In: *Proc. of the 37-th Annual Allerton Conference on Communication, Control and Computing*. 1999, pp. 368–377. URL: https://arxiv.org/abs/physics/0004057.

[107] Sana Tonekaboni, Danny Eytan, and Anna Goldenberg. “Unsupervised representation learning for time series with temporal neighborhood coding”. In: *arXiv preprint arXiv:2106.00750* (2021).

[108] Nicola Torrance et al. “Analysing the SF-36 in population-based research. A comparison of methods of statistical approaches using chronic pain as an example”. In: *Journal of Evaluation in Clinical Practice* 15.2 (2009), pp. 328–334.

[109] Saidrasul Usmankhujaev et al. “Time series classification with inceptionfcn”. In: *Sensors* 22.1 (2021), p. 157.

[110] Kotryna Vereščiagina, Kazys Vytautas Ambrozaitis, and Bronius Špakauskas. “Health-related quality-of-life assessment in patients with low back pain using SF-36 questionnaire”. In: *Medicina* 43.8 (2007), p. 607.

[111] Jingyuan Wang et al. “WHEN: A Wavelet-DTW hybrid attention network for heterogeneous time series analysis”. In: *Proceedings of the 29th ACM SIGKDD conference on knowledge discovery and data mining*. 2023, pp. 2361–2373.

[112] Zhiguang Wang, Tim Oates, et al. “Encoding time series as images for visual inspection and classification using tiled convolutional neural networks”. In: *Workshops at the twenty-ninth AAAI conference on artificial intelligence*. Vol. 1. Austin, TX. 2015, pp. 1–7.

[113] Zhiguang Wang, Weizhong Yan, and Tim Oates. “Time series classification from scratch with deep neural networks: A strong baseline”. In: *2017 International joint conference on neural networks (IJCNN)*. IEEE. 2017, pp. 1578–1585.

[114] John E Ware et al. “SF-36 health survey”. In: *Manual and interpretation guide* 2 (1993).

[115] John E Ware Jr. “SF-36 health survey update”. In: *Spine* 25.24 (2000), pp. 3130–3139.

[116] Li Wei and Eamonn Keogh. “Semi-Supervised Time Series Classification”. In: *Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*. 2006.

[117] William Young, Gary Weckman, and W Holland. “A survey of methodologies for the treatment of missing values within datasets: Limitations and benefits”. In: *Theoretical Issues in Ergonomics Science* 12.1 (2011), pp. 15–43.

[118] George Zerveas et al. “A transformer-based framework for multivariate time series representation learning”. In: *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining*. 2021, pp. 2114–2124.

[119] Guojun Zhang. “A modified svm classifier based on rs in medical disease prediction”. In: *2009 Second International Symposium on Computational Intelligence and Design*. Vol. 1. IEEE. 2009, pp. 144–147.

[120] Xuchao Zhang et al. “Tapnet: Multivariate time series classification with attentional prototypical network”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 04. 2020, pp. 6845–6852.

[121] Yu Zhang and Qiang Yang. “A survey on multi-task learning”. In: *IEEE Transactions on Knowledge and Data Engineering* 34.12 (2021), pp. 5586–5609.

[122] Bowen Zhao et al. “Rethinking attention mechanism in time series classification”. In: *Information Sciences* 627 (2023), pp. 97–114.

[123] Y. Zheng et al. “Time series classification using multi-channels deep convolutional neural networks”. In: *International Conference on Web-Age Information Management*. Springer, 2014, pp. 298–310.

[124] Jingwei Zuo, Karine Zeitouni, and Yehia Taher. “Smate: Semi-supervised spatio-temporal representation learning on multivariate time series”. In: *2021 IEEE International Conference on Data Mining (ICDM)*. IEEE. 2021, pp. 1565–1570.

Appendix A: EQ-5 Clinical Study Settings

In this section, we briefly outline the clinical study, list the inclusion and exclusion criteria for the study, describe the dataset of passive cell phone logs, and provide descriptive statistics on the participants.

iFeel app: real-time mobile data

iFeel is an innovative digital health research platform enabling passive and active digital monitoring and providing continuous objective measurements for any disorder. iFeel harnesses mobile phone data to find associations between digital markers and HRQL obtained via periodically EQ-5D self-reports. The passively sensed data is collected continuously, 24/7. For this study, the data consisted of communication patterns (number of phone calls, duration of incoming and outgoing calls, communication app usage duration), device usage (power on/off, doze mode in/out, number of device screen opens and locks, Wi-Fi connections, Bluetooth connections, battery usage, network mobile on/off, airplane mode on/off), and app usage (name of app and seconds of active app usage). All the collected information had non-identifiable information and fully complied with the General Data Protection Regulation (GDRP). No app-specific data, content, texts, or other sensitive data was collected. Per the study's protocol, data were collected over six months from the intake data.

9.0.1 SF-36 for Measuring Chronic Pain

The SF-36 questionnaire is a useful monitoring tool for patients with chronic pain [29, 68]. It measures health-related quality of life (HRQL) using 36 questions within eight concepts: physical functioning, bodily pain, role limitations due to physical health problems, role limitations due to personal or emotional problems, emotional well-being, social functioning, energy/fatigue, and general health perceptions. Each domain is measured on a scale from 0 (meaning worst possible health) to 100 (meaning perfect health) [115].

The SF-36 questionnaire has been widely used for measuring HRQL in patients with chronic pain [55, 110]. As can be expected, patients suffering from chronic pain reported significantly lower SF-36 scores compared to the general population [110, 108].

Previous studies showed that mean bodily pain scores of patients with chronic pain range from approximately 35 in severe cases when the patients suffer from multiple pain sites [26] to 64 in other cases [55, 52]. The equivalent means of the control groups who did not suffer from chronic pain were over 80 [26, 52].

9.0.2 EQ-5D as a Patient-Reported Outcome

The EQ-5D questionnaire is a standardized, generic measure of HRQL [41]. It is widely used in clinical and economic appraisal and population health surveys. EQ-5D assesses health status in five dimensions: mobility, self-care, usual activities, pain/discomfort, and anxiety/depression. In the EQ-5D-3L version, each dimension has three levels: no problems, some problems, and extreme problems. In the EQ-5D-5L version, designed to improve the instrument's sensitivity and reduce ceiling effects, each dimension has five levels: no problems, slight problems, moderate problems, severe problems, and extreme problems [85, 46]. EQ-5D is designed for self-completion by respondents and can hence be referred to as a patient-reported outcome measure [25]. Patients can complete the questionnaire themselves to provide information about their current health status and how it changes over time. The patient indicates their health state by choosing the most appropriate statement in each dimension, and their decision is associated with a 1-digit number that expresses the level selected for that dimension. The digits for the five dimensions can be combined into a 5-digit number describing the patient's health state. In the EQ-5D-5L version, the best health state is represented by 11111, and the worst by 55555 [86].

The health state captured by the EQ-5D is often converted into a single index, summarizing the patient's health using a country-specific value set [85]. A value set consists of weights attached to each level in each dimension, and a designated formula converts the EQ-5D health state into a single value on a scale anchored at 1 (meaning full health) and 0 (meaning a state as bad as death). The scale allows negative values to be assigned to health states considered worse than dead. In the absence of a designated value set for a specific country, the analysis should be based on a value set for a similar country [24].

9.0.3 Recruitment and Participant Breakdown

This study included adults who regularly attended physician appointments at a local Pain Clinic. They were recruited by their physician during appointments or by research staff during treatment visits. The study and its goals were explained to obtain the patient's consent. Ninety-one patients consented to the study and were guided to download the iFeel app. Once downloaded, the app initiated collecting relevant passively sensed data from the mobile phone and EQ-5D questionnaires. Research assistants trained the participants on filling out the questionnaires. Each patient filled out their first questionnaire in the presence of the research assistants. Afterward, patients could fill out the questionnaires as much as they liked on their own time. Per the research protocol, they were asked to do so weekly and received weekly reminders from the research assistants. The research assistants also contacted participants during the follow-up period in case of missing active or passive data collection and assisted in technical or adherence issues.

In addition, participants manually filled out the SF-36 questionnaire in the presence of the research assistant three times during the study - in the beginning, middle, and end.

9.0.4 Statistical Analysis and Train-Test Split

The patient sample included participants aged 18-80, with an average age of 60. Female participants accounted for 52.7% of the participants (48/91), and males accounted for 47.3% (43/91). As illustrated in Figure 9.1, men and women reported similar USA index levels in their EQ-5D questionnaires during the study. The same was true for the pain domain. Two Generalized Mixed Linear Models (GLMMs) [11] confirmed that there were indeed no statistically significant gender differences in the reported USA index levels (P=0.376) and pain levels (P=0.487).

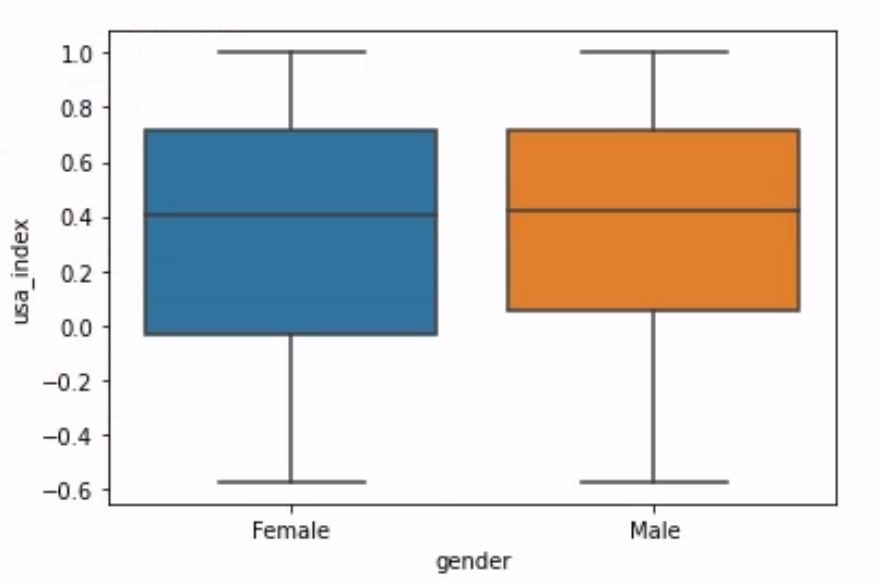


Figure 9.1: Box Plots of EQ-5D Health per Gender

On some occasions, patients filled out more than one EQ-5D questionnaire per day. In such cases, we kept the latest one. Still, each participant had multiple observations - health states reported on different dates and their corresponding cellular usage data. To avoid data leakage, all labels of the same patient were assigned to either the train or the test set.

To avoid bias, we aligned male-female and class "0" out of total proportions between the sets. Multiple Z-tests for independent samples showed no difference in gender or class "0" between the train and test sets (P>0.1). Table 9.1 presents the number of labels for males and females in each class and each set.

A total of 188 SF-36 questionnaires were filled out during the study, 86 at the beginning, 43 in the middle, and 59 at the end. Regarding the pain domain, patients reported extremely low outcomes (indicating severe pain) with an average of 29.6. A third GLMM confirmed that there were no statistically significant differences in the reported pain levels between men and women (P=0.788) throughout the study. However, patients reported better health (less pain) in the middle and the end of the study compared to the

beginning (P=0.002 and 0=0.001, respectively). Figure 9.2 shows the upward trend of the mean SF-36 pain levels within the three study phases.

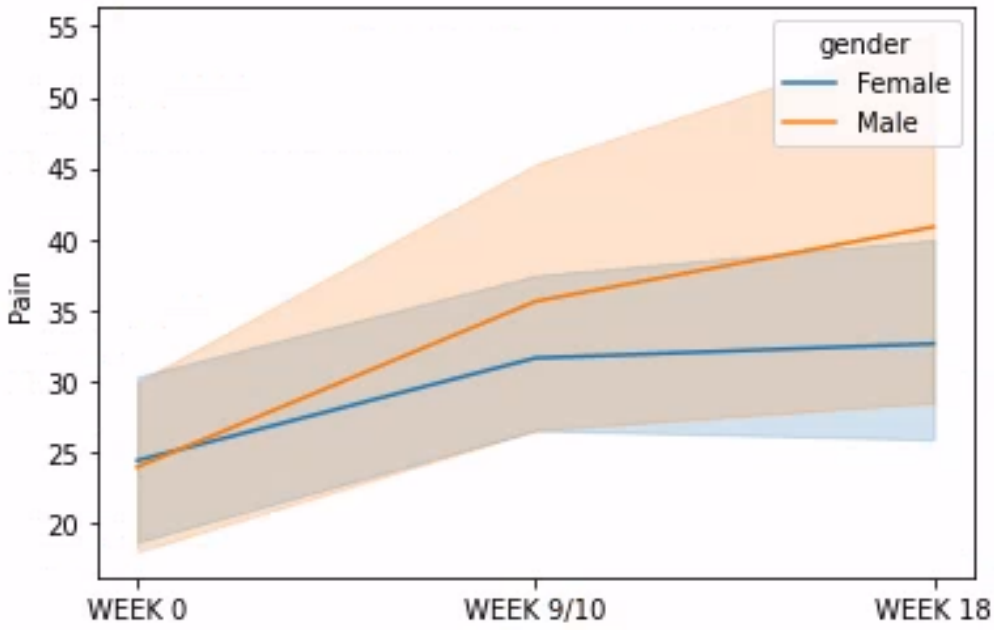


Figure 9.2: Mean SF-36 Pain by Experiment Phase

To assess the correlation between the two questionnaires, specifically in their ability to measure pain, we investigated the reported outcomes of patients who filled out both questionnaires within the same week. Overall, 113 patients filled out the EQ-5D and the SF-36 questionnaires within the same week: 84 on week 0 when the study started, 19 on week 9/10 in the middle of the study, and 10 on week 18, the final week of the study. The two measures use different scales - SF-36 is scaled between 0 (worst health) and 100 (best health), whereas the EQ-5D-5L is scaled between 1 (best health) and 5 (worst health). Hence, we calculated the Spearman correlation between the SF-36 bodily pain concept and the *reversed* ED-5D-5L pain/discomfort domain, which was 0.57.

To provide insights not just in terms of the EQ-5D index but also in SF-36 pain scales, we assessed the influence of our binary EQ-5D labels on the reported SF-36 pain scores. Comparing the mean SF-36 pain scores across our two label classes, revealed the binary index's ability to distinguish between SF-36 pain levels. The mean SF-36 pain score was significantly lower for the negative EQ-5D indexes (13.7) compared to the non-negative ones (30.7).

Table 9.1: Train-Test Split

		Class 0	Class 1	All
Train	Male	21 (26.9%)	57 (73.1%)	246
	Female	45 (26.8%)	123 (73.2%)	
Test	Male	13 (22.0%)	46 (78.0%)	134
	Female	27 (36.0%)	48 (64.0%)	

Appendix B: EQ-5D Predictions and Pre-Processing

We aimed to predict EQ-5D health indices from measurements passively collected by the iFeel app from patients' cell phones. These included app usage (app type and duration) and cellphone events (e.g., the screen on/off) and were collected continuously during the clinical study.

Our aim and setting pose a few inherent challenges:

- Data's nature – many apps and events can be monitored using the iFeel app. However, at each point, only a handful are active. Thus, the data is highly sparse. Also, we did not collect any specific app data or content to preserve privacy. Only indications for app usage were used.
- Dataset size – data was collected continuously, 24/7, for six months, but for a relatively small number of subjects. Thus, we had a reasonable amount of data per patient (intra-patient examples) but a relatively small amount of cross-patient data (inter-patient sequences). Also, there was a fair amount of missing data due to connectivity issues, version updates, etc.
- Labels – our labels were based on patient subjective responses to the EQ-5D questionnaire. We had various numbers of labels per patient, as each patient decided when and how many times to fill out their questionnaire. Moreover, the self-reports reflected patients' own views of their health with their unique scaling.
- Data and label alignment – syncing between labels derived from the (higher level) EQ-5D responses and the (low level) cellphone features is also challenging: An EQ-5D response reflects periodical patient condition. In contrast, the high-frequency cell phone data demonstrates the patient's instantaneous activities and state. Hence, the self-assessment labels and the collected cellphone measurements act at different time scales and reflect different levels of information. Thus, duplicating the same label over multiple adjacent feature vectors or labeling just one instance and utilizing semi-supervised methods were unsuitable solutions.

We addressed these challenges via a carefully designed preprocessing step, followed by a two-step classification method to maximize the utilization of labeled and unlabeled data. At the preprocessing step, we first mapped the various apps into predefined app

types. Then, we aggregated the data on an hourly basis and constructed feature vectors where each feature recorded hourly usage (in seconds) of a specific app type. Applying these actions reduced the feature space dimension and sparsity.

To sync between the labels and the feature vectors, we grouped the feature vectors from the preceding week for each self-report (label). Thus, a labeled example was a set of 181 feature vectors with a single EQ-5D index score, i.e., the set's label. It should be noted that for all patients, most weeks had no label since the patient did not fill out a questionnaire. Hence, we ended with many unlabeled feature vectors and a few labeled sets.

To overcome these unique characteristics of our problem, we designed a dedicated two-step method for learning behavioral patterns and deviations. First, we use unsupervised learning techniques to utilize the immense amounts of unlabeled cellphone usage data. As described in subsection 10.0.1, we cluster the vectors and change their representation to capture hourly behavioral patterns. Then, in 10.0.2, we train supervised classification models to predict the class of the labeled vectors.

10.0.1 Unsupervised Representation Learning

Our pipeline starts by clustering the (unlabeled) feature vectors using the k-means clustering algorithm to reflect hourly behavioral patterns. The outcome was a set of clusters, each reflecting a typical hourly behavior for this patient population.

Our pipeline starts by clustering the (unlabeled) feature vectors using the K-means clustering algorithm. The outcome is a set of clusters, each reflecting a typical hourly behavior for this patient population. The pipeline includes two optional transformations preceding the clustering step. Both are designed to refine the clusters by reducing dimensionality and sparsity and making the clusters more distinct.

The first transformation, termed the Pre-Clustering Transformation (PCT), is a deep auto-encoding, followed by Uniform Manifold Approximation and Projection (UMAP) [75] learning of the encoder's latent representation. The PCT component transforms the sparse, high-dimensional raw feature vectors into a dense, lower-dimensional feature vectors embedded in a smooth manifold. This transformation contributes to the stability of the proceeding clustering procedure.

The second transformation, termed Discriminative Clustering Transformation (DCT), starts by clustering and mapping the feature vectors to the nearest clusters. It provides a pseudo label (the mapped cluster) for each feature vector. Then, we train a two-header autoencoder-classifier model (a.k.a multi-task learning [121]) and extract the latent representation. The DCT component essentially implements a self-learning procedure designed to increase the separability of the latent representation before the (original) unsupervised clustering procedure.

In the next pipeline step, after clustering, each feature vector is mapped to the closest cluster, as measured by Mahalanobis distance [19]. The distances from each cluster were binned into ten quantiles, and each vector ascribed to the cluster belonged to one of them. Thus, each vector was mapped to a specific distance quantile in its closest cluster. Unique vectors (representing deviations from common behavioral patterns) were

mapped to higher quantiles. As a result, each labeled week (with a self-assessment) was projected onto the obtained clusters, such that each hourly vector within the week was assigned to its corresponding quantile in the nearest cluster.

Finally, every labeled set of 181 hourly feature vectors was represented by a single vector of counts. The counts recorded the number of vectors assigned to each quantile of each cluster. The outcome was a labeled dense vector representing an entire week, where each entry corresponded to a specific quantile holding the aggregated counts of hourly feature vectors that were mapped to higher quantiles. In essence, this serves as an empirical approximation of the feature vectors' inverse CDF, capturing typical behavioral patterns and deviations throughout the week. Figure 10.1 illustrates the construction of representation from the raw data, through the behavioral clustering, to the count vector.

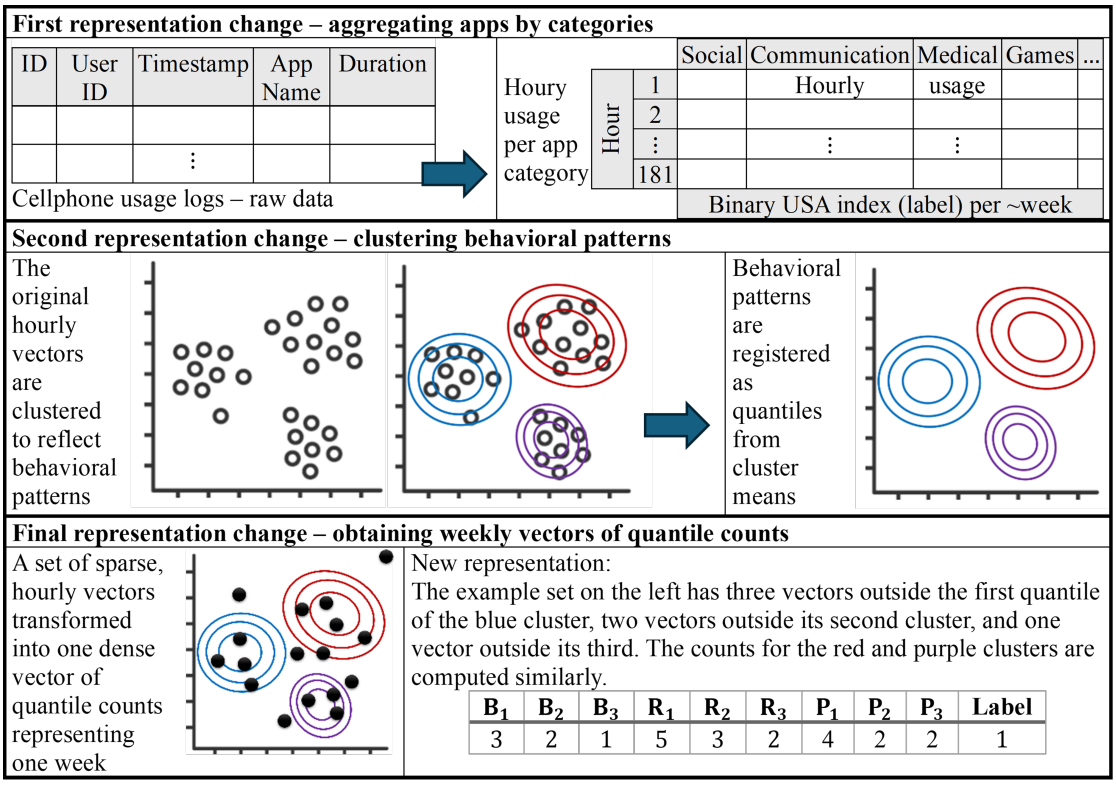


Figure 10.1: Two-step Representation. Mapping hourly vectors to behavioral clusters. B_i, R_i, P_i: Number of vectors outside the i-th quantile of the blue, red, and purple clusters, respectively.

10.0.2 Supervised Classification

Standard supervised learning methods and best practices were applied at this step. After further preprocessing the data using feature selection, traditional classification methods were applied: logistic regression (LR), random forest (RF), and k-nearest neighbors (kNN). The outcome was a model designed to predict a patient's state given behavioral patterns from the preceding week, as captured by their cellphone usage.

We trained separate models for men and women as well as gender-agnostic models for comparison. For each gender - Male, Female, All - all possible combinations of pre-clustering transformation, K, and classification model were evaluated.

For model evaluation, we used a test set formed by cellphone data and self-assessment labels of patients excluded from both the representation learning and the classifier training. Hyperparameter optimization was performed using grid search cross-validation on the training set, while the held-out test set was used to evaluate model performance on unseen data. In section 10.0.3, we report performance on the test set and provide statistical guarantees using Fisher and Barnard's tests.

10.0.3 ResultsandDiscussion

Representation Results

The raw cellphone data were mapped to 56-dimension feature vectors representing hourly app type usage (in seconds). After cleansing, we had an unlabeled dataset of 95165×56 vectors. Using the k-means clustering algorithm with 50 initial starts, we transformed the labeled example sets into $K \times 9$-dimension feature vectors in the representation change phase, and ended up with 246 labeled vectors in the training set and 134 labeled vectors in the test set.

We present the results of gender-specific modeling, which outperformed models trained on both men's and women's data. Table 10.1 presents the best classification results for each gender (Male, Female, All) and each pipeline (Standard, PCT, DCT, or both) with the chosen clustering K (16, 24, 32, 64, or 128) and classification model (RF, LR, or kNN). Note that the "Standard" pipeline refers to the clustering phase without any optional preceding variants, PCT or DCT, as described in subsection 10.0.2. To help identify powerful settings, the highest values for each metric appear in bold. The table reflects the robustness of our classification results, as they are similar across all settings.

Table 10.1: Best Classification Results per Gender and Pre-clustering Transformation

	Setting			F1-Score			Class 0		Class 1		AUC
Gender	Pipeline	K	Model	Acc.	M. avg.	W. avg.	Prec.	Recall	Prec.	Recall	ROC
Male	Standard	24	kNN	0.797	0.607	0.760	0.600	0.231	0.815	0.957	0.594
Male	PCT	32	kNN	**0.898**	**0.832**	**0.891**	**0.889**	0.615	0.900	**0.978**	0.797
Male	DCT	16	kNN	0.831	0.699	0.810	0.714	0.385	0.846	0.957	0.671
Male	PCT+DCT	24	LR	0.831	0.777	0.838	0.588	**0.769**	**0.929**	0.848	**0.809**
Female	Standard	16	kNN	0.787	0.755	0.780	0.762	0.593	0.796	**0.896**	0.744
Female	PCT	32	RF	**0.827**	**0.807**	**0.824**	**0.792**	0.704	0.843	**0.896**	**0.800**
Female	DCT	32	RF	0.773	0.767	0.778	0.639	**0.852**	**0.897**	0.729	0.791
Female	PCT+DCT	24	LR	0.787	0.778	0.790	0.667	0.815	0.881	0.771	0.793
All	Standard	24	LR	0.754	0.704	0.753	0.590	**0.575**	**0.821**	0.830	**0.702**
All	PCT	16	kNN	0.739	0.637	0.714	0.609	0.350	0.766	**0.904**	0.627
All	DCT	32	LR	**0.761**	**0.706**	**0.757**	**0.611**	0.550	0.816	0.851	0.701
All	PCT+DCT	128	kNN	0.724	0.624	0.702	0.560	0.350	0.761	0.883	0.616

Acc. = Accuracy; M. avg. = Macro average; W. avg. = Weighted average; Prec. = Precision.

Classification Results

For the men, the **PCT with $K = 32$ and kNN** setting achieved the best results. The $K = 32$ was chosen for the K-means clustering algorithm and the kNN model used the k=5 nearest neighbors for the classification. Even though it did not achieve the best recall on class 0 or precision on class 1, it performed best according to all the other metrics and appeared to be the most stable setting. Overall, the kNN algorithm performed well in our task, as it frequently appears in the classification results table.

As for the women, the most stable setting was **PCT with $K = 32$ and RF**. It significantly outperformed all other models in every F1-Score measure while maintaining high precision and recall values.

Both Male and Female models consistently outperformed the gender-agnostic models (All). While gender-specific modeling should be considered when applicable, gender-agnostic modeling can help with non-binary cases or when app users prefer not to mention their gender.

Fisher and Barnard's tests were used to determine the classification significance of our best settings: PCT with kNN for the men and PCT with RF for the women. As both tests showed highly statistically significant results (Fisher test $P \leq 5 \times 10^{-6}$, Barnard's test $P \leq 2 \times 10^{-6}$), it is unlikely to receive our predictions purely by chance.

תקציר

סיווג סדרות זמן היא בעיה נפוצה בתחומים רבים. בעיות אלו מאופיינות לעיתים קרובות בקשרים תוך-סדרתיים בתוך התכונות עצמן, וכן בקשרים בין-סדרתיים בין אותן תכונות לאורך זמן. פיתוח מודל כללי אשר מסוגל ללכוד את שני סוגי הקשרים הללו בצורה יעילה מהווה אתגר משמעותי. במציאות, איסוף מידע איכותי הוא לעיתים קרובות משימה מורכבת, ותהליך התיוג עלול להיות יקר מאוד ולדרוש מומחיות תחומית. לכן, פעמים רבות קיימות מערכות נתונים גדולות שבהן רק חלק קטן מהדוגמאות מתויגות – מצב טיפוסי ללמידה חצי-מפוקחת.

בעבודה זו אנו מציעים גישה חדשה בשם Gaussian Process Variational Information Bottleneck. מדובר במודל קצה-לקצה אשר מטרתו ללמוד ייצוג קומפקטי של סדרת הזמן תוך שימור המידע ההכרחי בלבד לצורך סיווג מדויק. בניגוד למודל GP-VAE, אשר מתמקד בשימור מידע לשם שחזור ודורש תהליך דו-שלבי לצורך סיווג, ה-GP VIB מכוון את תהליך הלמידה ישירות למטרת הסיווג. מממצאי הניסויים עולה כי המודל GP-VIB מציג ביצועים טובים יותר מהשיטות הקיימות במאגרי סדרות זמן מרובי-משתנים כגון UEA. בנוסף, אנו מראים כי המודל שלנו מסוגל להתמודד גם עם סדרות זמן בדגימה לא סדירה (ISTS), Irregularly Sampled Time Series – כולל מאגרי נתונים רפואיים מהעולם האמיתי.

יתרה מזאת, אנו מציגים הרחבה חצי-מפוקחת של המודל GP-VIB, אשר עושה שימוש בנתונים לא מתויגים כדי לשפר את ביצועי הסיווג על הדגימות המתויגות. תוצאותינו מראות כי הגרסה החצי-מפוקחת של המודל עולה בביצועיה על הגרסה המפוקחת. כמו כן, אנו מראים שהשיטה המוצעת שלנו משיגה תוצאות עדיפות על הקיים במאגר נתונים אמיתי אשר עוסק בחיזוי מצבים בריאותיים של מטופלים הסובלים מכאב כרוני, בהתבסס על נתוני שימוש בטלפון נייד שנאספו על ידי מכון מדעי הנתונים.

עבודה זו נכתבה בהנחיית ד"ר שי פיין, ראש המכון למדעי הנתונים, אוניברסיטת רייכמן

אוניברסיטת רייכמן

בית־ספר אפי ארזי למדעי המחשב

התכנית לתואר שני (M.Sc.)־ מסלול מחקרי

וואריאציה לצוואר בקבוק למידע עם מקדמים של תהליכים גאוסיאנים לסיווג חצי מפוקח של סדרות בזמן

איתמר אפרתי

עבודת תזה המוגשת כחלק מהדרישות לשם קבלת תואר מוסמך M.Sc.
במסלול המחקרי בבית ספר אפי ארזי למדעי המחשב, אוניברסיטת רייכמן

מאי 2025