Imperial College of Science, Technology and Medicine
Department of Mathematics

Variational Autoencoders on Hilbert spaces

Lorenz Wolf

Supervised by Dr A. Duncan

Submitted in part fulfilment of the requirements for the degree of
Master's of Science in Statistics of Imperial College London, September 2021

Abstract

In this project we study Variational Autoencoders on Hilbert spaces as generative models for functional data and justify its suitability with the Grid Refinement Invariance Principle. We find that the basis expansion is a key component of our model and can make or break the suitability of the model for a specific problem. In particular, the choice of basis should reflect properties of the data set such as smoothness or roughness. The sensitivity of the model regarding the latent dimension is analysed and we find that the model is reasonably robust besides for the data-driven basis, based on Functional Principal Components. To avoid mode collapse experienced when training the standard model on a multimodal distribution we propose a generalisation to the conditional VAE to Hilbert Spaces. To evaluate the performance we apply the models to three different data sets, one simulated, and two observed in real world applications (Electricity consumption and chemometrics). The potential significance of the proposed conditional model is shown by augmenting an unbalanced data set with synthetic samples from our model. With this approach we manage to significantly increase the test accuracy of a classifier for the unbalanced classification problem, thus improving its generalisability. Finally, the conditional model is used in combination with a standard model to generate data from a bivariate joint distribution.

Acknowledgements

First and foremost, I want to thank Dr Andrew Duncan for being an amazing supervisor and for the invaluable insight and guidance he provided. I always enjoyed our discussions and felt inspired and excited to work on the new ideas. Thank you for giving me the freedom to explore but also being there whenever I had questions.

I want to thank my family for supporting me throughout my studies and my friends (special shoutout to Robin, Victoire, Paul, Adam, Oriane and Leo) who made the MSc at Imperial as fun and exciting as it was. Last but not least, I want to thank my flat mates for simply being the best.

Finally, a thank you goes to the Basil Furneaux Memorial fund for MSc scholarship.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation and Objectives

Functional data, data where an observation is considered a function, arises in many different fields. Important examples include but are not limited to economics, medicine and chemometrics. For example in a medical study the concentration of a certain protein in the blood of patient n at time t could be considered as a functional observation $x_n(t)$ [KR18].

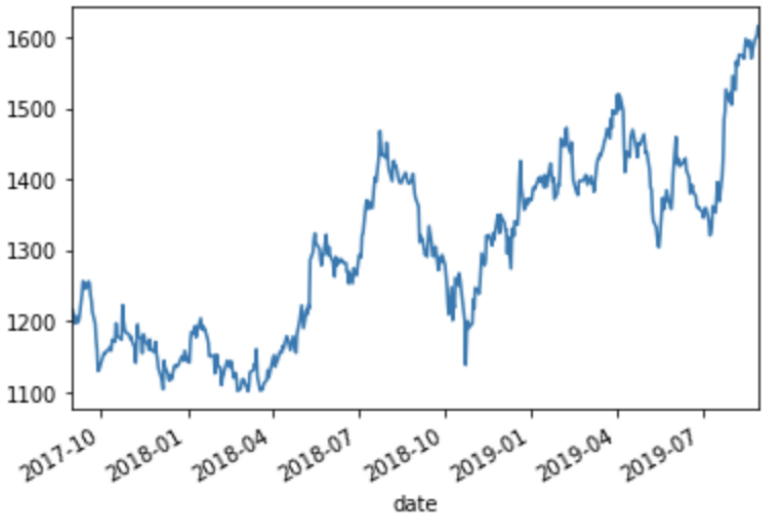

Figure 1.1: Stock price over time.

With developments in modern technology (among others wearable devices), more and more data

are collected continuously over a time interval or discretely at many time points in an interval. This is for example the case in the stock market where price changes occur in fractions of seconds 1.1. Another example is the electricity demand measured over time. Longitudinal data is a common type of functional data but not the only one. For example probability distributions over multiple measurements for each subject or images could be considered as functional data. The presence of functional data in many real-world problems requires the study of methods suitable for such data and as a consequence the field of functional data analysis has grown dramatically in the past decades.

Treating data as functions instead of multivariate vectors can be advantageous as it for example allows the evaluation at every time point as well as the use of derivatives. However, as we will see in Chapter 2, another reason is that classical methods can break down when applied to functional data and care must be taken in the development of new methods. A key principle for the development of methods for functional data is the so called Grid Refinement Invariance Principle [LCF15], which ensures robustness under changing dimensionality of the discretised functional data.

The abundance of functional data and the growing size of data sets require scalable methods for functional data which are able to capture complicated highly non-linear relationships in the data. For multivariate data exactly this is being achieved by Deep Learning models. In recent years the Deep Learning community has made vast progress in applying machine learning algorithms to solve complex problems (e.g. see [HZRS15], [KTS+14] and [JEP+21]). This progress is partly due to improved data availability and faster computations but also driven by advances in learning algorithms and model architectures. The successes of deep learning and the characteristics of functional data motivate us to explore scalable methods at the intersection of functional data analysis and deep learning.

In machine learning the area of Generative modelling has received a lot of attention recently since it can leverage the availability of large amounts of unlabeled data which. The goal is often data synthesis but can also be density estimation or labelling of data sets. An important application is data set augmentation to improve the performance of classification models for

unbalanced classification tasks by upsampling a drastically underrepresented class in the training data. This is the motivating example for the study of a generative model for functional data. In particular, inspired by [MFB20] we study Variational Autoencoders on Hilbert spaces to synthesise functional data and propose the conditional Variational Autoencoder on Hilbert spaces for the task of upsampling underrepresented classes.

1.2 Recent developments

Very recently attempts have been made to combine Deep Learning methodology with Functional Data. In [Gus16] a framework for training deep neural networks on infinite dimensional input spaces is proposed. The authors of [HSWH21] propose a Functional Autoencoder for representation learning on for functional data. [RR21] introduces a new approach for function on function regression with continuous neurons. Last but not least, in Mishra et al. [MFB20] proposed πVAE, a generative model for stochastic processes, and use it to perform Bayesian inference. This is a very active field of research and hence many more recent advancements aiming at developing deep learning methods for functional data exist and often build on ideas such as Functional Principal Components or basis representations.

1.3 Contributions

In this thesis we perform an in depth analysis of a novel approach to generative modelling for functional data inspired by πVAE [MFB20] and suggest several generalisations such as the conditional Variational Autoencoder on Hilbert spaces.

1. In Chapter 2 we concisely present the theoretical background required for functional data analysis. A particular focus lies on basis representations of functional data and Hilbert space theory. We combine this theory with that of deep generative modelling and Variational Autoencoders.

2. We study the use of Variational Autoencoders on Hilbert spaces as generative models for functional data . A comprehensive comparison of several different bases on data sets with different characteristics is performed in Section 3.4.

3. We transfer the use of inverse auto-regressive flows to Variational Autoencoders on Hilbert Spaces and provide a comparative analysis in Section 3.6.

4. We propose the use of a conditional model to generate data from multimodal functional distributions. The applicability of this method is shown by improving the generalisability of a classifier for an unbalanced classification problem on functional data. Additionally, we propose the use of the conditional model for multivariate data generation and present an application to a bivariate functional data set.

1.4 Statement of Originality

The work contained in this thesis is my own work unless stated otherwise.

Lorenz Wolf 03. September 2021, London

Chapter 2

Background Theory

2.1 Functional Data Analysis (FDA)

A well known part of Statistics is multivariate analysis where we generally deal with data sets consisting of observations $x_i \in \mathbb{R}^d$, i.e. finite dimensional vectors. On the other hand, Functional Data Analysis (FDA) is concerned with observations $x_i(t)$, $t \in T$ (with T and index set, commonly a real interval) that are considered to be functions characterised by infinite degrees of freedom. Functional Data has gained more and more attention, partly due to the progress in data collection methods allowing high frequency data of for example stock prices, but also audio recordings or images can be considered as functions to exploit certain properties.

The term functional data analysis was introduced by Ramsay and Dalzell in 1991 [RD91]. Since then the field of FDA benefited from a lot of work. In the following sections we cover the basic ideas of FDA, Hilbert space theory which is required to understand a key model of functional data, and several key techniques and results.

Having introduced FDA as the analysis of curves and surfaces it is important to remind ourselves that in reality the actual data are discrete in most cases. More specifically each observation consists of multiple evaluations of a function at for example different time points. Thus, when data is described as being functional it is referred to the intrinsic structure of the data, i.e. an

underlying function which gives rise to the data. It is therefore rather a difference in how the data is treated than an actual difference in the data between multivariate analysis and functional data analysis. At first this may seem like adding unnecessary constraints and complexity to the problem, but there are several advantages of adopting a functional data approach.

Firstly, treating each observation as several evaluations from an underlying function allows the evaluation at every point in time. Secondly considering rates of change is a powerful tool in FDA [RS05]. Furthermore, in many cases it is reasonable to assume smoothness of the functions which adds valuable information as evaluations close to each other can be expected to have similar function values.

Finally, methods for multivariate data can break down for functional data due to the high dimensionality of the observations compared to the number of observations in the data set. To illustrate this we provide the example given in Chapter 1 of [Dun21].

Example 2.1. *Consider modelling a sample* $\mathbf{X}_1, \ldots, \mathbf{X}_N$ *as i.i.d. realisations of a stochastic process* $\mathbf{X}(t)$ *with* $\mathbb{E}(\mathbf{X}^2(t)) < \infty$*. Aiming at testing whether the mean function* $m(t) = \mathbb{E}(\mathbf{X}(t))$ *is equal to 0 we perform a standard Hotelling's T-squared test based on the sample mean* $\bar{\mathbf{x}} \in \mathbb{R}^M$ *of multivariate measurements* $\mathbf{x}_1, \ldots, \mathbf{x}_N$*. To compute the test statistic* $\bar{\mathbf{x}}^T \hat{\Sigma}^{-1} \bar{\mathbf{x}}$ *we need to compute the sample covariance matrix* $\hat{\Sigma}$*. However, for functional data it is common to have* $M \gg N$ *in which case the sample covariance matrix is singular. It follows that we require methods which are robust for* M *and are still reliable as* $M \to \infty$*.*

This idea of robustness is described by the **Grid Refinement Invariance Principle (GRIP)** which is fundamental to Functional Data Analysis [LCF15].

Definition 2.1 (GRIP)**.** *GRIP states that methods for functional data should be robust under changes of the dimension of the representation as long as the dimension is large enough to give an accurate representation.*

Consequently the two main approaches for developing methods for functional data are to either devise a method for true functional data and project it into finite dimensions or to devise a method for discrete data and ensure that the method is well defined as $M \to \infty$ [Dun21].

In the literature functional data is categorised into dense and sparse functional data. These categories refer to the grid on which the evaluations are available. No strict approach for categorising functional data as dense or sparse exists, but attempts have been made in [ZW16]. Since high available high frequency data falls in the dense category and classical methods struggle for large $M \gg N$ the focus in this report lies on dense functional data. In general the 'power' of functional data analysis depends heavily on the density of evaluation points [RS05]. Intuitively, in areas of high curvature we require a higher density in order to be able to get a good picture of the underlying curve.

2.1.1 Functional Data as Stochastic Processes

To make the notion of functional data mathematically precise let $(\Omega, \mathcal{F}, \mathbb{P})$ be a probability space. Consider a stochastic process characterised by the collection

$$\{X(t,\omega) : t \in I, \omega \in \Omega\}$$

where $X(t,\cdot)$ is an $\mathcal{F}$ -measurable function on the sample space Ω. For simplicity, the *omega* argument is suppressed in the following.

A sample path or a realisation for the process is the collection of real numbers obtained when $X(t)$ is observed for every $t \in I$. In Functional Data Analysis the index sets of the processes giving rise to the data are some interval of the real line or more generally $\mathbb{R}^d$).

In reality the infinite dimensional functional data are not observed, rather a discretised version of it, e.g. evaluations of the function at several points in time. Thus, typically data sets consist of samples of the form

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \text{ where } \mathbf{x}_i = \left(X_i\left(t_j\right)\right)_{j=1}^{M}, \text{ for } t_1, \ldots, t_M \in I$$

I.e. n sample paths from a stochastic process with each path being evaluated at M locations. It is important to note that the points at which each sample path is evaluated could be irregularly

spaced and be different for each path. Furthermore it might be the case that the number of evaluations is not the same for all sample paths in the data set.

Before moving on to Hilbert space theory, a very elegant and practical framework to deal with functional data, it is worth to briefly note that common estimates of the mean and covariance functions, namely the sample mean function and sample covariance function follow intuitively from the scalar variable case by taking the pointwise mean

$$\bar{X}_N(t) = \sum_{n=1}^{N} X_n(t)$$

and similarly for the estimate of the covariance function

$$\hat{c}(s,t) = \frac{1}{N-1} \sum_{n=1}^{N} \left(X_n(t) - \bar{X}_N(t)\right) \left(X_n(s) - \bar{X}_N(s)\right).$$

2.1.2 Hilbert Space Theory

Hilbert spaces are a very useful framework for working with functions. We assume the reader is familiar with notions of vector spaces, inner products and completeness for metric spaces.

Definition 2.2 (Hilbert Space)**.** *A Hilbert space is a complete inner product space.*

Of particular interest for functional data analysis is the space $L^2([0,1])$.

Definition 2.3. *Define $L^2([0,1])$ to be the space of all Lebesgue measurable real-valued functions f such that*

$$\int_0^1 f^2(t)dt < \infty.$$

This is a Hilbert space with the inner product defined as

$$\langle f_1, f_2 \rangle = \int_0^1 f_1(t) f_2(t) dt$$

and the vector space operations $(f_1 + f_2)(t) = f_1(t) + f_2(t)$ and $(\lambda f_1)(t) = \lambda f_1(t)$.

The inner product space properties can be easily verified. To show completeness of $L^2([0,1])$ Alkhiezer and Galzman [AG93] note that a sequence of functions which is Cauchy $\{f_m(t)\}_{m=1}^{\infty}$ has a subsequence for which

$$\int_0^1 \left|f_{k_{r+1}}(t) - f_{k_r}(t)\right|^2 dt < \frac{1}{8^r} \quad (r = 1, 2, 3, \ldots).$$

It can then be shown that this subsequence converges to a function f(t) on a set $I^\star \subseteq [0,1]$, equivalently almost everywhere on [0,1]. Using uniform convergence of said subsequence on subsets I_s (an increasing sequence with $\lim_{s\to\infty} I_s = I^\star$) and the Cauchy property the limit can be taken through the integral. It follows that f is indeed in $L^2([0,1])$ and

$$\lim_{m\to\infty} \int_a^b |f_m(t) - f(t)|^2 \, dt = 0,$$

which completes the sketch of the proof. For a full proof the reader is referred to [AG93]. This can be extended to the space $L^2(\mathcal{D})$ where $\mathcal{D}$ is any compact subset of $\mathbb{R}^d$. For many problems in FDA the space $L^2(\mathcal{D})$ is too large and we want to exploit smoothness so it is useful to restrict ourselves to so called Sobolev spaces.

Definition 2.4. *Denote by $H^K([0,1])$ the space of functions $f \in L^2([0,1])$ whose (weak) derivatives up to order K are also contained in $L^2([0,1])$.*

This is a Hilbert space with the inner product defined by

$$\langle x, y\rangle_{\mathcal{H}^\kappa} = \sum_{k=0}^{K} \int D^{(k)}x_(t) D^{(k)}y_(t) dt,$$

where $D^{(k)}x$ denotes the order k weak derivative of the function x [Dun21].

2.1.3 Representing functions through basis expansions

With the number of evaluations per observation often being large it is of interest to concisely summarize the information contained in one observation. A common technique to achieve this is to decompose the function into a linear combination of basis functions, and then characterise

the function by the resulting (infinite) vector of coefficients. For example the common functional normal linear regression model makes use of basis expansions to project the optimisation problem from functions to scalar coefficients [KR18].

Definition 2.5. *A Hilbert space $\mathcal{H}$ is called separable if there exist countable $\{e_1, e_2 \ldots\}$, satisfying $\langle e_j, e_j \rangle = 1 \ \forall j$ and $\langle e_j, e_i \rangle = 0 \ \forall j \neq i$, such that every $x \in \mathcal{H}$ can be written as*

$$x = \sum_{j=1}^{\infty} a_j e_j.$$

Then $\{e_1, e_2 \ldots\}$ is called an orthonormal basis for $\mathcal{H}$

In particular we have that for a separable Hilbert space the coefficients in the above sum are defined by $a_j = \langle x, e_j \rangle$. Furthermore, Parseval's theorem states that

$$\langle x, x \rangle = \sum_{j=1}^{\infty} \langle x, e_j \rangle.$$

This can readily be applied to the space $L^2([0,1])$. In practice the infinite sum is often truncated after the $B-th$ term for a sufficiently large B. Thus, we approximate the linear combination of the full basis by

$$x(t) \approx \sum_{k=1}^{B} c_k e_k(t),$$

where $c_k = \int x(t) e_k(t) dt$ and $\{e_k(t)\}_{k=1}^{B}$ are some standard collection of basis functions. The function can then be represented by the vector of coefficients $(c_1, \ldots, c_B)$. This transfers the analysis of infinite dimensional functions to finite dimensional vectors. Here B is also called the dimension of the basis expansion. With $B = M$ (M the number of evaluations) this corresponds to exact interpolation. The smaller B the smoother the interpolation hence B is often treated as additional parameter [RS05]. The choice of basis is very important. For a well chosen basis the coefficients themselves are interesting descriptors of the data and smaller B allows a reasonable interpolation of the data [RS05]. In the following paragraphs we introduce several orthonormal bases for the space $L^2([0,1])$.

Fourier basis

One of the most well known bases is the Fourier basis, which has many applications in signal processing and data analysis.

Definition 2.6. *The Fourier basis is given by the functions*

$$e_1(t) = 1,\ e_2(t) = \sqrt{2}sin(2\pi t),\ e_3(t) = \sqrt{2}cos(2\pi t).$$

The Fourier basis can easily be generalised to other interval lengths. Due to its periodic nature as a composition of sine and cosine functions it is especially widely used for periodic data. A benefit of the Fourier basis is its very efficient computation of the coefficients with the FFT algorithm [CT65] when M is even and evaluation points are equally spaced.

The strength of this basis lies on stable functions, while it struggles to interpolate functions with strong local features. The expansions obtained by the Fourier basis are generally smooth and hence may be inappropriate for data subject to discontinuities [RS05]. Consequently, it is often of interest to also consider other bases.

Wavelet basis

While basis expansions with for example the Fourier basis only allow the analysis in the frequency domain wavelets stand out by enabling the analysis at different scales, also called multiresolution analysis. In other words, it allows us to not only identify frequencies in the signal but also when those frequencies are present. Consequently and due to a strong mathematical background combined with fast computation wavelets have found various applications in for example data compression, noise removal and pattern recognition (see e.g. [BB94] for a famous example) .

To obtain the continuous wavelet transform we first need to define the mother wavelet ψ. In

order to be a mother wavelet the Fourier transform $\widehat{\psi}$ of a function ψ must satisfy

$$\int_{-\infty}^{\infty} \frac{|\widehat{\psi}(\omega)|^2}{|\omega|} d\omega < \infty$$

which is also called admissibility condition [Pat09]. This ensures that the mother wavelet vanishes at $-\infty$ and ∞ and changes sign. In fact it is thanks to the decay that wavelets extract information in the time domain as well. The mother wavelet is then dilated and translated into functions

$$\psi_{jk}(t) = \frac{1}{\sqrt{|k|}} \psi \left(\frac{t-j}{k} \right).$$

A wide range of mother wavelets exists, but all of them are designed to ensure dilations and translations are orthogonal and represent an orthonormal basis. The wavelet transform maps the function $x(t) \in L^2(\mathbb{R})$ to the time-scale space by

$$W_x(j,k) = \int_{-\infty}^{\infty} \psi_{jk}(t) x(t) dt.$$

In reality we are not able to use the complete wavelet transform but rather specify scales and dilations so that the continuous wavelet transform yields a two dimensional array. In this report we use the Ricker wavelet defined by

$$\psi(t) = \frac{2}{\sqrt{3\sigma}\pi^{1/4}} \left(1 - \left(\frac{t}{\sigma} \right)^2 \right) e^{-\frac{t^2}{2\sigma^2}}$$

and with an example visualised in Figure 2.1.

A key strength of wavelets is that due to the different scales wavelets are less affected by discontinuities in the signal compared to for example Fourier transforms[RS05]. This follows from the fact that a discontinuity will only affect the wavelets which are non-zero at the discontinuity.

We note that due to problems with the implementation of continuous wavelet transform and the use in combination with a convolutional Variational Autoencoder the discrete wavelet transform with the Daubechies 2 wavelet has been implemented instead

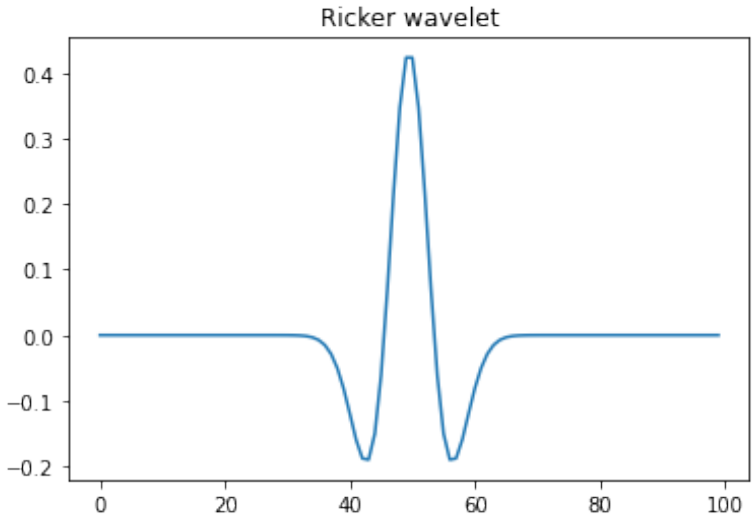

Figure 2.1: Visualising the Ricker wavelet or also called Mexican hat wavelet.

Thus far we have only discussed expansions that are obtained by representing the original function as a linear combination of a set of pre-specified basis functions such as the Fourier basis.Another approach is to design a basis that encodes the structure of the data by for example computing it empirically from the data at hand. The optimal orthonormal basis to represent data in can be found via Functional Principal Components.

2.1.4 Karhunen Loève Expansion and Functional Principal Components

First, recall that for any $f \in L^2(\mathcal{D})$ with an orthonormal basis $\{e_n\}_{n=1}^{\infty}$ we can write

$$f(t) = \sum_{n=1}^{\infty} f_n e_n(t),$$

were $f_n = \int_{\mathcal{D}} f(t)e_n(t)dt$. Also note that the notion of eigenvalues and eigenfunctions for an operator on a Hilbert space is as expected.

Definition 2.7. *Let L be a linear operator on a Hilbert space $\mathcal{H}$. Then λ is called an eigenvalue of L and $x \in \mathcal{H}$ is the corresponding eigenfunction if $Lx = \lambda x$.*

A key result for stochastic processes and in FDA is the Karhunen Loève theorem which gives a similar expansion for a stochastic process. The idea behind it being that we can express a stochastic process as a combination of random variables and eigenfunctions of an integral operator to be defined below.

First let us define the general Integral operator T_K associated with kernel K by

$$(T_K f)(t) = \int_0^1 K(t,s)f(s)ds.$$

Then Mercer's theorem describes how to obtain a basis induced by a kernel K satisfying certain properties.

Theorem 2.1 (Mercer's theorem)**.** *Let K be a continuous symmetric non-negative definite kernel. Then there is an orthonormal basis $\{e_n\}_{n\in\mathbb{N}}$ of $L^2([0,1])$ consisting of eigenfunctions of T_K such that the corresponding eigenvalues $\{\lambda_n\}_{n\in\mathbb{N}}$ are non-negative. The eigenfunctions corresponding to non-zero eigenvalues are continuous on $[0,1]$ and K has the representation*

$$K(s,t) = \sum_{n=1}^{\infty} \lambda_n e_n(s)e_n(t)$$

where the convergence is absolute and uniform.

More specifically, let $\{X_t,\ t \in [0,1]$ be a zero mean process on $L^2([0,1])$ with continuous covariance function $\gamma(s,t) = Cov(X_s, X_t)$. Define the integral operator T_γ associated with γ by

$$(T_\gamma f)(t) = \int_0^1 \gamma(t,s)f(s)ds.$$

We note that the covariance function is symmetric and non-negative. Hence in combination with the continuity assumption applying Mercer's theorem to $\gamma(t,s)$ yields that the eigenfunctions of T_γ form an orthonormal basis of $L^2([0,1])$. Furthermore, T_γ has a sequence of non-negative eigenvalues. This is linked to stochastic processes via the Karhunen Loève theorem.

Theorem 2.2 (Karhunen Loève)**.** *Let $\{X_t,\ t \in [0,1]$ be a zero mean process on $L^2([0,1])$ with*

continuous covariance function $\gamma(s,t)$. *Then*

$$X_t = \sum_{n=1}^{\infty} \xi_n e_n(t), \quad t \in [0,1],$$

where $\xi_n = \int_0^1 X_t e_n(t)dt$ *and* $\{\lambda_n, e_n(t)\}_{n=1}^{\infty}$ *are the eigenvalues and eigenfunctions of* T_γ. *Furthermore, we have that* $\mathbb{E}\xi_n = 0$ *and* $\mathbb{E}(\xi_n \xi_m) = \lambda_n \delta_{n,m}$.

The series obtained through the Karhunen Loève expansion converges in L^2 to X(t), uniformly in t. The convergence follows by Mercer's theorem and a full proof can be found in [Pav18]. Note that in the Karhunen Loève expansion the coefficients ξ_n are in fact random variables containing information about the variance of the stochastic process around the eigenfunctions. It follows that realisations of the stochastic process can be understood as realisations of the random coefficients $\{\xi_n\}$.

In the general case of the Karhunen Loeve expansion the random variables $\{\xi_n\}$ are only uncorrelated, which can still lead to very complex dependencies and the expansion does not yield a practical scheme for simulation. However, note that applying the Karhunen Loève expansion to a Gaussian second-order process X_t with continuous covariance function yields $\{\xi_n\}$, which are the time integrals of a Gaussian process and consequently Gaussian random variables. Since jointly distributed, uncorrelated Gaussian random variables are independent, the Karhunen Loève expansion becomes

$$X_t = \sum_{n=1}^{\infty} \sqrt{\lambda_n} \xi_n e_n(t),$$

where $\{\xi_n\}_{n=1}^{\infty}$ are independent $\mathcal{N}(0,1)$.

Thus, Gaussianity is commonly assumed since independence is required for many theoretical results and allows simple methodology [DH10].

In practice, we are interested representing data using only a finite number of basis functions. Therefore it is important to choose this set of finite basis functions to adequately capture the variability and structure of the data. Functional Principal Components provides an approach

to obtain an optimal (in an appropriate sense) basis.

Functional Principal Components (FPCA)

FPCA is a powerful tool in FDA that is very similar to PCA for multivariate data. As in the multivariate case, with FPCA we aim to investigate the modes of variation of functional data. Framing the choice of B basis functions to represent the data in as an optimisation problem FPCA allows us to obtain the corresponding optimal basis.

Let $\mathbf{X}$ be a zero mean square integrable random variable in $L^2([0,1])$ and denote the inner product of $L^2([0,1])$ by $\langle \cdot, \cdot \rangle$. Then consider the problem of finding an orthonormal basis of B-elements which best approximates the random variable. Therefore, we aim to minimise the loss function for a given orthonormal basis $u_1, \ldots, u_B$ defined by

$$S(u_1, \ldots, u_B) = \mathbb{E} \left\| \mathbf{X} - \sum_{i=1}^{B} \langle \mathbf{X}, u_i \rangle u_i \right\|^2 .$$

Due to the orthonormality of the basis functions minimising the loss is equivalent to maximising $\sum_{n=1}^{B} \langle T_\gamma u_n, u_n \rangle$ where T_γ is as defined in Theorem 2.2. This however is achieved by choosing $u_1, \ldots, u_B$ to be the eigenfunctions of T_γ corresponding to the B largest eigenvalues. Thus, FPCA is really a truncated Karhunen Loève expansion.

Since in reality we observe discretised versions of the underlying stochastic process of the form

$$\{\mathbf{x}_1, \ldots, \mathbf{x}_n\}, \text{ where } \mathbf{x}_i = \left(X_i\left(t_{ij}\right)\right)_{j=1}^{M}, \text{ for } t_{i1}, \ldots, t_{iM} \in I$$

which means that we need to estimate the eigenfunctions from the data. For dense data this is commonly achieved by approximating the covariance function by the empirical covariance function [BHK09], then computing the estimated eigenfunctions (in practice eigenvectors that can be interpolated if needed). For sparse data more care is required and often smoothing methods are applied first (e.g. see [HMW06], [YMW05]).

An advantage of FPCA is that we can choose the dimensionality of the expansion based on the percentage of variability explained of the total variability, often taken to be 0.95. This allows an automated choice of the dimensionality B.

FPCA has been applied in many different areas which range from linguistics [ACE10] to time series data in finance [BH05] (more examples are given in [RS02]). It has also been used in medicine for example in [VGS05] the authors apply FPCA to fMRI scans. FPCA is especially popular because of its use for dimensionality reduction and also visualisation.

2.1.5 Basis construction through kernels

We have already encountered kernels in the special case of the covariance function. In this subsection we consider kernels in general and outline the use of kernels to construct a basis.

Kernels form an important part in statistical learning and machine learning. They are especially interesting since they can be used for several types of data, importantly for us also functional data [MFSS16]. The so called kernel trick has been successfully applied to methods such as PCA and Support Vector Machines [CV95] and can generally be applied to any algorithm that can be expressed in the form of an inner product $\langle x, y \rangle$.

Specifically if an algorithm depends on inputs only via $\langle x, y \rangle$ it is often of interest to introduce non-linearity. In general this can be done by applying a feature map $\varphi : \mathcal{X} \to \mathcal{F}$, mapping the inputs into a higher dimensional feature space $\mathcal{F}$. We then evaluate the inner product in $\mathcal{F}$ by

$$k(x, y) = \langle \varphi(x), \varphi(y) \rangle_{\mathcal{F}}$$

where $\langle \cdot, \cdot \rangle_{\mathcal{F}}$ denotes the inner product in $\mathcal{F}$.

Definition 2.8 (non-negative definite kernel)**.** *A function $k : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a non-negative definite kernel if it is symmetric, i.e., $k(\mathbf{x}, \mathbf{y}) = k(\mathbf{y}, \mathbf{x})$, and:*

$$\sum_{i,j=1}^{n} c_i k\left(\mathbf{x}_i, \mathbf{x}_j\right) c_j^{\star} \geq 0$$

for any $n \in \mathbb{N}$, $\{\mathbf{x}_i\}_{i=1}^n \in \mathcal{X}$ *and any* $c_1, \ldots, c_n \in \mathbb{C}$.

The following theorem with proof in [Aro50] makes explicit when such a kernel trick can be applied.

Theorem 2.3. *k is a non-negative definite kernel if and only if there exists a Hilbert space $\mathcal{F}$, and a function $\varphi : \mathcal{X} \to \mathcal{F}$ such that $\forall x, y \in \mathcal{X}, k(x, y) = \langle \varphi(x), \varphi(y) \rangle_{\mathcal{F}}$.*

So for any non-negative definite kernel we can find a corresponding feature space and feature map. It can be shown that the feature space and feature map are unique up to isomorphisms [Aro50]. Furthermore, such a kernel defines a reproducing kernel Hilbert space (RKHS) $\mathfrak{H}$. In fact $\mathfrak{H}$ is the feature space $\mathcal{F}$. The canonical feature map is given by $\varphi(x) = k(x, \cdot)$. This yields the so called reproducing properties, for all $f \in \mathfrak{H}$ and all $x \in \mathcal{X}$:

$$\langle k(\cdot, x), f \rangle = f(x) \text{ and } \langle k(\cdot, x), k(\cdot, y) \rangle = k(x, y)$$

Definition 2.9. *A Hilbert space $\mathfrak{H}$ of functions is a reproducing kernel Hilbert space ($RKHS$) if the evaluation functionals $L_{\mathbf{x}}[f]$ defined as $L_{\mathbf{x}}[f] = f(\mathbf{x})$ are bounded, i.e., for all $\mathbf{x} \in \mathcal{X}$ there exists some $C > 0$ such that*

$$|L_{\mathbf{x}}[f]| = |f(\mathbf{x})| \leq C \|f\|_{\mathfrak{H}}, \quad \forall f \in \mathfrak{H}$$

Note that in practice it can often be hard to derive the feature map explicitly. Many different types of kernels exist, some of the more popular kernels include the Gaussian, Square exponential and Matérn kernels. In this report we consider the Matérn kernel defined by

$$k_M(\mathbf{x}, \mathbf{x}') = \frac{2^{1-\nu}}{\Gamma(\nu)} \left(\frac{\sqrt{2\nu}\, \|\mathbf{x} - \mathbf{x}'\|_2}{\sigma} \right) K_\nu \left(\frac{\sqrt{2\nu}\, \|\mathbf{x} - \mathbf{x}'\|_2}{\sigma} \right), \quad \sigma > 0, \nu > 0$$

where K_ν is the modified Bessel function of second kind of order ν and Γ is the Gamma function. We choose the Matérn kernel for the following reason: While for example the Gaussian kernel defines an RKHS whose elements are infinitely many times differentiable, i.e. very smooth, the

Matérn kernel defines a more flexible RKHS. In fact the RKHS is a Sobolev space (Definition 2.4) where the number of derivatives (i.e. smoothness) is controlled by the parameters of the Matérn kernel. Controlling the smoothness can be understood as managing the trade-off between bias and variance. Learning smoother functions leads to higher bias but less data is required to learn the function, while rougher functions have less bias but more data will be needed to learn them.

It is important to note that the Matérn kernel is translation invariant since it only depends on $\mathbf{x}$ and $\mathbf{x}'$ through $\mathbf{x} - \mathbf{x}'$. Translation invariant non-negative definite kernels can be characterised by Bochner's theorem, which we state for kernels on $\mathbb{R}^d$ below.

Theorem 2.4 (Bochner's theorem)**.** *A complex-valued bounded continuous kernel* $k(\mathbf{x}, \mathbf{x}') = \psi(\mathbf{x} - \mathbf{x}')$ *on* $\mathbb{R}^d$ *is positive definite if and only if there exists a finite non-negative Borel measure* Λ *on* $\mathbb{R}^d$ *such that*

$$\psi(\mathbf{x} - \mathbf{x}') = \int_{\mathbb{R}^d} e^{\sqrt{-1}\boldsymbol{\omega}^\top(\mathbf{x}-\mathbf{x}')} \mathrm{d}\Lambda(\boldsymbol{\omega})$$

Bochner's theorem ensures that when properly scaled the Fourier transform of a translation invariant kernel is a proper probability distribution. Bochner's theorem provides the theory behind Random Fourier Features a technique in machine learning to approximate the feature map [RR08].

As with the covariance function for functional principal components, Mercer's theorem (Theorem 2.1) ensures the existence of the kernel induced basis. In practice, where discretised versions of the data are observed, to obtain a basis via the Matérn kernel we first evaluate the kernel on a dense grid $t_1, \ldots, t_n \in I$ and obtain the Gram matrix defined by $G_{ij} = k(t_i, t_j)$ for $i, j = 1, \ldots, n$. The basis is then obtained by computing the eigenvectors of the Gram matrix which can be interpolated to obtain eigenfunctions. For more details see Appendix A. The absolute and uniform convergence in Mercer's theorem yields that the infinite dimensional RKHS can be approximated by finite dimensions.

2.1.6 Rough path theory and signatures

Recently the signature transform has seen a rise in machine learning applications. The study of rough paths and signatures dates back to K. T. Chen who developed theory for piecewise smooth paths [Che77].

In this section we restrict ourselves to piecewise differentiable paths, however, the theory can be extended to paths with bounded variation. By path we mean a continuous map $X : [a, b] \to \mathbb{R}^d$ with the coordinate paths $(X_t^1, \dots, X_t^d)$, where each X^i is a real valued path. Now for any integer $k \geq 1$ and indexes $i_1, \dots, i_k \in \{1, \dots, d\}$ we can iteratively define

$$S(X)_{a,t}^{i_1,\dots,i_k} = \int_{a<s<t} S(X)_{a,s}^{i_1,\dots,i_{k-1}} dX_s^{i_k}. \tag{2.1}$$

or equivalently

$$S(X)_{a,t}^{i_1,\dots,i_k} = \int_{a<t_k<t} \cdots \int_{a<t_1<t_2} dX_{t_1}^{i_1} \dots dX_{t_k}^{i_k}. \tag{2.2}$$

Then the signature of the path X is given by the sequence

$$S(X)_{a,b} = \left(1, S(X)_{a,b}^1, \dots, S(X)_{a,b}^d, S(X)_{a,b}^{1,1}, S(X)_{a,b}^{1,2}, \dots\right) \tag{2.3}$$

with the superscripts running along the set

$$W = \{(i_1, \dots, i_k) \mid k \geq 1, i_1, \dots, i_k \in \{1, \dots, d\}\}.$$

In practice we often consider the truncated signature up to some depth $m \in \mathbb{N}$. Furthermore, note that since we only observe discretised versions of the underlying path interpolation is applied to join the evaluations [Rei17].

An important property causing interest in the signature transform is its invariance under time reparameterisations [CK16]. One of the only pieces of information that is lost when taking the signature transform is the speed at which a path is traversed. This can in fact be a

great advantage since for example in pattern recognition or computer vision the only thing of interest is the resulting curve obtained by traversing on the path. Furthermore, by definition it encodes global features of the path. Additionally, the length of the signature is not dependent on the number of evaluations available per path. The length of the signature of a path on $\mathbb{R}^d$ up depth m has d^m terms. This can be an advantage but also a drawback. The more detail we want to capture, the more levels of the signature are required which can lead to large feature vectors. Finally, importantly any continuous function of the path can be arbitrarily well approximated by a linear function of its signature [BKA+19] providing a strong motivation for the use of the signature transform for feature extraction. Consequently the signature transform has been applied in Machine Learning among others in motion recognition [LZJ17] and Finance ([LNA19], [KLA19]).

The signature transform is similar to a basis expansion, but it does not require the choice of a basis so is non-parametric.

In [HL05] it is shown that the signature of paths with bounded p-variation is unique (extended from piecewise regular paths). This raises the question whether the signature can be inverted to obtain the corresponding path. The inversion of the signature transform is a complicated topic. The theory for the inversion is studied in [LX18] and then further analysed in [Gen17]. A recent advancement is the insertion algorithm [CL19]. Motivated by the signature of simple paths it aims to reverse engineer the path by combining information from adjacent levels of the signature. Specifically, the algorithm is based on approximating higher level terms from lower level terms in the signature. While it works well in theory the signature inversion via the insertion algorithm can only determine an $m + 2$ point approximation to a path based on its signature up to depth m (where the first point is required as input to map the reconstructed path onto the original).

Another approach to path reconstruction from the signature exploits the fact that the signature transform is differentiable [BKA+19]. Thus, given the truncated signature $\mathrm{Sig}^N(\mathbf{x})$ of an

unknown $\mathbf{x} \in \mathcal{S}(\mathbb{R}^d)$ we can apply gradient descent to minimise the loss

$$L(\mathbf{y}; \mathbf{x}) = \left\| \mathrm{Sig}^N(\mathbf{y}) - \mathrm{Sig}^N(\mathbf{x}) \right\|_2^2 \quad \text{for } \mathbf{y} = (y_1, \ldots, y_n) \in \mathcal{S}\left(\mathbb{R}^d\right).$$

An exciting idea is presented in [BKA+19] where the signature transform is embedded into deep learning architectures as a layer itself. A spin-off of resulted in the Signatory package [KL20] which enables efficient computation of the signature and also contains an implementation of the insertion algorithm for signature inversion.

2.1.7 Maximum mean discrepancy

Closely related to the theory discussed on kernels and hilbert spaces is the maximum mean discrepancy (MMD). The MMD is an integral probability metric [Mül97] which can be used to measure the discrepancy between to samples. In [WD20] a kernel two sample test for functional data which relies on the MMD is proposed. We follow this approach for our experiments in Chapter 3. Given a kernel k and the associated RKHS $\mathcal{H}_k(\mathcal{X})$ we define $\mathcal{P}$ the set of Borel probability measures on $\mathcal{X}$. Furthermore, assuming k is measurable define $\mathcal{P}_k \subset \mathcal{P}$ as the set of all $P \in \mathcal{P}_k$ such that $\int k(x,x)^{\frac{1}{2}} dP(x) < \infty$. For $P, Q \in \mathcal{P}_k$ we define the Maximum Mean Discrepancy denoted $\mathrm{MMD}_k(P, Q)$ as follows

$$\mathrm{MMD}_k(P, Q) = \sup_{\|f\|_{\mathcal{H}_k(\mathcal{X})} \leq 1} \left| \int f dP - \int f dQ \right|.$$

Using the kernel trick it can be shown that the MMD can simply be obtained by an inner product on the RKHS which only involves expectations and can be unbiasedly estimated by a Monte Carlo estimator such as

$$\widehat{\mathrm{MMD}}_k\left(X_n, Y_n\right)^2 := \frac{1}{n(n-1)} \sum_{i \neq j}^{n} h\left(z_i, z_j\right),$$

where $h\left(z_i, z_j\right) = k\left(x_i, x_j\right) + k\left(y_i, y_j\right) - k\left(x_i, y_j\right) - k\left(x_j, y_i\right)$ [WD20]. Note the above estimator can be straightforwardly generalised to unbalanced sample sizes.

Several kernels can be considered but based on simulations performed in [WD20] we use CEXP, the SE-T kernel with T based on the cosine-exponential kernel. For $F \in \mathbb{N}$ define

$$k_{\cos(F)}(s,t) = \sum_{n=0}^{F-1} \cos(2\pi n(s-t)) \text{ on } [0,1]^2,$$

then the cosine-exponential kernel is

$$k_{\text{c}-\exp(F,l)}(s,t) = e^{-\frac{1}{2l^2}(s-t)^2} k_{\cos(F)}(s,t)$$

In addition, due to dependence on the kernel we also report results with the squared-exponential T kernel with $T(x) = (x, x^2)$.

Definition 2.10. *For $T : \mathcal{X} \to \mathcal{Y}$ the squared-exponential T kernel $(SE-T)$ is defined as*

$$k_T(x,y) = e^{-\frac{1}{2}\|T(x)-T(y)\|_{\mathcal{Y}}^2}.$$

As shown in the simulations in [WD20] the MMD is a powerful tool for comparing two distributions supported over a real, separable Hilbert space such as $L^2(\mathcal{D})$ with $\mathcal{D} \subset \mathbb{R}^d$.

2.2 Deep Learning

2.2.1 Generative Modelling

Generative versus discriminative modelling divides machine learning. In discriminative modelling the goal is to learn a predictor given an observation, one common example is learning the classification of an observed image as dog or cat with for example a deep neural network. On the other hand, generative models learn the simulate the generation process of the data in the real world [KW19], i.e. they learn to model the distribution of the training samples. Intuitively speaking the model is trained to generate data which can't be told apart from the training data. More precisely generative models are trained so that the model samples $\tilde{\mathbf{x}} \sim p_\theta(\cdot)$ come

from the same distribution as that from which the training data is truly sampled, $\mathbf{x} \sim p_d(\cdot)$.

Generative models are trained via unsupervised learning which comes with the great advantage of learning from unlabeled data and thus simplifies the collection of data considerably. However, regardless of the decreased effort during data collection, unlabelled data still contains valuable information. Generative modelling extracts this information and leverages it. Generative models have various applications including image/ audio/ video synthesis such as text-to-image conversion [ZXL+16], super-resolution [LTH+16] and speech or music synthesis. Furthermore, generative models can be applied in density estimation and out of distribution detection to identify anomalies [DHL19].

One might ask why the generation of synthetic data and generative modelling in general is of importance. The applications of generative models are immense. One key aspect is Data augmentation. When training classifiers or other models it can be very useful to augment the data set before training in order to avoid biases or increase the performance of a classifier. A great demonstration of this approach can be found in [FADK+18] where the authors achieve a significant improvement of the trained classifier through synthetic data augmentation. While data augmentation can be extremely useful, in certain industries such as finance and medicine it is often not possible to share data at all due to privacy issues. However, synthetic data can be shared with out any privacy concerns and ideally still carries most information contained in the original data set. Additionally, generative models are applied in semi-supervised learning and neural network pre-training for problems where the collection of labeled data is expensive see for example [GDCS19].

The most common types of generative models are Energy-Based models (EBM)[SK21], Variational Autoencoders (VAE) [KW13], Generative Adversarial Netowrks (GAN) [GPAM+14], Autoregressive models, and Normalising Flows ([KPB19], [PNR+19]). Since the beginning of Generative modelling in the 1980s Energy-Based models have been improved upon by advances in deep learning architectures and the availability of larger data sets. Most of the improvements are based on the introduction of latent variables that are easily sampled and then transformed. However, in most cases this is an intractable problem which lead to development of approximate

inference techniques and applications of methods from for variational inference [JGJS99]. For a good overview of generative modelling and a comparison of the most common models we refer to [BTLLW21].

In order for generative models to be applicable to real world problems they are usually required to satisfy the following criteria [RMW14]:

1. Flexibility to allow the model to capture complex structures in the data.
2. Efficient generation of synthetic data from the inferred model.
3. Computational tractability and scalability to high dimensional data.

A key issue with EBMs is the slow, even intractable, training for high dimensional problems [LCH06], which is related to slow mixing times of MCMC methods required for sampling [BTLLW21]. This is where Variational Autoencoders come in. VAEs have gained popularity due to their ability to efficiently generate samples and yet yielding good sample diversity [BTLLW21]. In the following sections we will describe VAEs and the underlying inference process in detail. We will give an overview of recent improvements to the vanilla VAE and in particular describe π-VAE, a newly proposed extension to encode priors on function spaces [MFB20].

2.2.2 Variational Autoencoder

Variational Autoencoders were first proposed in [KW13] and [RMW14] and since have found various applications and have been improved by several generalisations and extensions some of which will be discussed in later subsections.

Assume the data $\mathbf{X} = \{\mathbf{x_i}\}_{i=1}^{N}$ consists of N i.i.d. samples of the random variable $\mathbf{x}$, which is assumed to be truly distributed according to $\mathbf{x} \sim p_d(\mathbf{x})$. We choose to model it by $p_\theta(\mathbf{x})$, where θ are the model parameters. From the i.i.d assumption it follows that the log-likelihood of the data given the model and parameters is

$$log\ p_\theta(\mathbf{X}) = \sum_{\mathbf{x}\in\mathbf{X}} log\ p_\theta(\mathbf{x}). \tag{2.4}$$

Maximising the likelihood or equivalently the log-likelihood, also called Maximum Likelihood Estimation (MLE), is a frequentist approach and commonly applied to fitting probabilistic models.

Based on equation 2.4 we will only consider the likelihood of a single data point $\mathbf{x}$ (dropping the subscript i) in the following.

VAEs make use of latent variables, denoted by $\mathbf{z}$, by first mapping the inputs into the latent space via the encoder and then reconstructing the input from the latent space representation via the decoder. Thus, the full model can be summarised by $p_\theta(\mathbf{x}|\mathbf{z})$ with the prior $p_\theta(z)$ and the posterior $p_\theta(\mathbf{z}|\mathbf{x})$. The problem with latent variable models is that optimising the model via maximum likelihood is intractable. This is due to the graphical model representing the joint distribution

$$p_\theta(\mathbf{x}, \mathbf{z}) = p_\theta(\mathbf{x}|\mathbf{z})p_\theta(z) \tag{2.5}$$

which yields the marginal distribution in the form

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}, \tag{2.6}$$

commonly called the marginal likelihood. Unfortunately, for slightly more complex models, such as neural networks with a non-linear hidden layer, this integral is intractable or no convenient estimator exists ([GBC16], [KW19]). Thus, approximate inference is needed.

Evidence Lower Bound (ELBO)

Instead of maximising the intractable likelihood, the true posterior $p_\theta(\mathbf{z}|\mathbf{x})$ (intractable) is approximated with a learnt function $q_\phi(\mathbf{z}|\mathbf{x})$, also called the recognition model [KW13].

An important tool in Machine Learning and approximate inference is the Kullback-Leibler (KL) divergence, which measures the difference between two probability distributions.

Definition 2.11 (KL Divergence)**.** *Given two separate probability distributions* $P(\mathrm{x})$ *and* $Q(\mathrm{x})$

over the same random variable x*, the KL divergence is defined by*

$$D_{\mathrm{KL}}(P\|Q) = \mathbb{E}_{\mathrm{x}\sim P}\left[\log \frac{P(x)}{Q(x)}\right] = \mathbb{E}_{\mathrm{x}\sim P}[\log P(x) - \log Q(x)]. \tag{2.7}$$

Having defined the KL divergence note that each term in the sum in Equation 2.4 can be written as

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = D_{KL}\left(q_{\phi}(\mathbf{z} \mid \mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})\right) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}), \tag{2.8}$$

where $\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x})$ is the evidence lower bound (ELBO), also called variational lower bound on the marginal likelihood of a data point x, given by

$$\begin{aligned}\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) &= \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}\left[-\log q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) + \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})\right] &(2.9)\\ &= -D_{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{z})\right) + \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})\right] &(2.10)\end{aligned}$$

[KW13]. Using Jensen's inequality it can be shown that the KL divergence is non-negative. Hence it follows from Equation 2.8 that

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \geq \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}).$$

Notice that the ELBO is independent of the intractable true posterior and only depends on the approximate posterior. Thus, instead of optimising the marginal likelihood the ELBO is optimised w.r.t the variational parameters ϕ and the inference parameters θ. Therefore, it is required to back-propagate gradients through the stochastic sampling layer $\tilde{\mathbf{z}} \sim q_{\phi}(\mathbf{z}|\mathbf{x})$. Estimating the gradient of the ELBO w.r.t. the inference parameters is straight forward. However, the estimation of the gradient w.r.t. the variational parameters requires more attention. The naive Monte Carlo estimator of the gradient of the ELBO is not suitable for learning as it exhibits high variance so that an alternative estimator is needed [PBJ12].

Before discussing the optimisation consider the definition of the ELBO. In Equation 2.10 the KL divergence term can be interpreted as a regulariser aiming to bring the approximate posterior $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$ close to the prior distribution $p_{\boldsymbol{\theta}}(\mathbf{z})$ [GBC16]. The second term is known as the

reconstruction loss between the model reconstructions and the original inputs. On the other hand, considering Equation 2.8 it is clear that maximising the ELBO brings the approximate posterior closer to the true posterior, i.e. improves the encoder. Simultaneously, it approximately maximises the marginal likelihood, as required [KW19]. Furthermore, note that the parameters $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$ are shared between all data points. This approach is called amortised inference (e.g. see [HBWP12]) and enables the model to scale to large data sets.

Reparameterisation Trick

In the case of continuous $\mathbf{z}$ the reparameterisation trick allows back-propagation through the sampling layer, i.e. computing the gradients w.r.t. the variational parameters $\boldsymbol{\phi}$. Instead of sampling directly from $q_\phi(\mathbf{z}|\mathbf{x})$ a noise variable $\boldsymbol{\epsilon}$ is sampled from a suitable prior $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ and then transformed via a differentiable function giving $\mathbf{z} = g_\phi(\boldsymbol{\epsilon}, \mathbf{x})$ [KW13]. This reparameterisation allows the estimation of gradients of expectations of a function $f(\mathbf{z})$ with respect to $q_\phi(\mathbf{z}|\mathbf{x})$ with Monte Carlo estimates [KW13]. In the case where g(.) is invertible using a change of variables we have that $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] = \mathbb{E}_{p(\epsilon)}[f(\mathbf{z})]$. Hence aggregating over several samples we can use the Monte Carlo estimator $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[f(\mathbf{z})] \simeq \nabla_\phi \sum_{\ell=1}^{L} f(\mathbf{z}_\ell)$, where $\mathbf{z}_\ell$ for $\ell = 1, \ldots, L$ is transformed noise $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ [KW19]. Using this reparameterisation the Monte Carlo estimator of the ELBO of a data point $\mathbf{x}$ is given by

$$\boldsymbol{\epsilon}_\ell \sim p(\boldsymbol{\epsilon}) \tag{2.11}$$

$$\mathbf{z}_\ell = \mathbf{g}(\boldsymbol{\phi}, \mathbf{x}, \boldsymbol{\epsilon}_\ell) \tag{2.12}$$

$$\tilde{\mathcal{L}}(\boldsymbol{\theta}, \boldsymbol{\phi}, \mathbf{x}) = \frac{1}{L} \sum_{\ell=1}^{L} \log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}_\ell) - \log q_\phi(\mathbf{z}_\ell \mid \mathbf{x}), \quad \text{for } L \in \mathbb{N}. \tag{2.13}$$

Note that L is commonly chosen to be equal to 1. Thanks to the reparameterisation the gradient of this estimator with respect to the variational parameters ϕ can be easily computed via automatic differentiation and yields an unbiased estimator of the true gradient [KW19]. Hence it enables the approximate minimisation of $-\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x})$ (Equation 2.10) via stochastic gradient descent algorithms such as Adam [KB14], which are efficient and scalable. Furthermore, note

that in many cases the KL divergence can be integrated analytically which only leaves the estimation of $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}\left(\mathbf{x} \mid \mathbf{z}\right)\right]$ and thus generally yields a lower variance.

For computational efficiency it is important that $p(\boldsymbol{\epsilon})$ can be easily sampled from. Additionally, flexibility is required in order to fit to the true posterior which can be complex [BTLLW21].

Partly thanks to methods from stochastic simulation the reparameterisation trick can be used to sample from various distributions $q_\phi(\mathbf{z}|\mathbf{x})$ [KW13]. In particular, for any distribution which is part of a location-scale family the reparameterisation trick can be applied by sampling $\boldsymbol{\epsilon}$ from the standard distribution with location=0 and scale=1 and then applying $g(.) = \text{location} + \text{scale} \times \boldsymbol{\epsilon}$, where location and scale are determined by the encoder (recognition model).

Gaussian VAE

While many different variations of the VAE exist, the most com mon version uses a Gaussian prior. Most basic is the use of a Gaussian distribution with diagonal covariance matrix as prior so that $q_\phi(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z};\ \boldsymbol{\mu},\ \boldsymbol{\sigma}^2\mathbf{I})$ [BTLLW21]. This yields the following end-to-end formulation of the VAE:

$$(\boldsymbol{\mu},\ \log(\boldsymbol{\sigma})) = \text{Encoder}(\boldsymbol{\phi},\ \mathbf{x}) \tag{2.14}$$

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma}\boldsymbol{\epsilon} \quad \text{where } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{2.15}$$

$$\hat{\mathbf{x}} = \text{Decoder}(\boldsymbol{\theta},\ \mathbf{z}) \tag{2.16}$$

In the case where both distributions $q_\phi(\mathbf{z}|\mathbf{x})$ and $p_\theta(\mathbf{z})$ are Gaussian the KL divergence term in the ELBO can be evaluated analytically, so that we can benefit from the lower variance in the estimation of the ELBO.

In the case of continuous $\mathbf{z}$ and with $p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$ a Gaussian distribution, the marginal likelihood can be seen as an infinite mixture of Gaussians [KW19].

The choice of a Gaussian with diagonal covariance matrix can be easily extended to a full covariance by letting the encoder network output a lower triangular matrix $\mathbf{L}$ instead of $\mathbf{log}(\boldsymbol{\sigma})$

and then setting $\mathbf{z} = \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon}$. This corresponds to the covariance matrix $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$.

The issue of blurry samples

VAEs obtain state-of-the-art results and can even be applied as manifold learning algorithms [GBC16]. However, the main drawback is that samples are blurry which is especially noticeable for image generation. This blurriness has been attributed to the VAE approximation of the maximum likelihood objective. More specifically, if the encoder is not discriminative enough and maps different data points $\mathbf{x}$ to the same $\mathbf{z}$ in the latent space blurry samples will be observed as they are essentially an average [ZSE17].

Another issue is the flexibility of the variational posterior $q_\phi(\mathbf{z}|\mathbf{x})$ since

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = D_{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})\right) + \mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x})$$

tells us that the approximate objective of maximising the ELBO is equivalent to the true objective of maximising the likelihood when $D_{KL}\left(q_\phi(\mathbf{z} \mid \mathbf{x}) || p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})\right) = 0$. This raises the question how can the KL divergence term be decreased.

2.2.3 Improvements and recent work on VAEs

VAE with Inverse Autoregressive Flows

Generative models in general and VAEs specifically are a very active field of research. Hence several extensions to the originally proposed VAE model have been suggested. One of which is the introduction of normalising flows to increase the flexibility of the variational posterior. This has been first introduced by Rezende and Mohamed in [RM15] and then developed further for high dimensional spaces by Kingma et al. with the Inverse Autoregressive Flow (IAF) [KSJ+16]. Kingma et al. proposed the use of IAF due to its flexibility and scalability to high dimensions.

The idea of normalising flows is to start with a known distribution $\mathbf{z_0} \sim q(\mathbf{z_0}|\mathbf{x})$ and then

to iteratively transform this distribution into a more flexible distribution via a transformation $f_t(\cdot)$ to yield distributions $\mathbf{z_t} = f_t(\mathbf{z_{t-1}}, \mathbf{x})$.

Given that the Jacobian determinant of the transformations $f_t(\cdot)$ can be computed the probability density functions of the iterates $\mathbf{z_t}$ can be computed via change of variables [KSJ+16]. In particular, for the $T-th$ iterate we have that

$$\log q\left(\mathbf{z}_T \mid \mathbf{x}\right) = \log q\left(\mathbf{z}_0 \mid \mathbf{x}\right) - \sum_{t=1}^{T} \log \det \left| \frac{d\mathbf{z}_t}{d\mathbf{z}_{t-1}} \right|. \tag{2.17}$$

To understand IAF first note that it is the inverse of the Masked Autoregressive Flow (MAF) first introduced in [PPM17]. In Autoregressive models the joint density is modelled as a product of conditionals. For the MAF each conditional is modeled as a Gaussian:

$$p\left(x_i \mid x_{1:i-1}\right) = \mathcal{N}\left(x_i \mid \mu_i, \sigma_i^2\right), \quad i = 1, \ldots, D \tag{2.18}$$

where the input $\mathbf{x} \in \mathbb{R}^D$ [PPM17]. Here a MADE network [GGML15] is used to output the mean and standard deviation parameters. The MADE network satisfies the autoregressive property:

$$\left.\begin{aligned} \mu_i &= f_{\mu_i}(x_{1:i-1}) \\ \sigma_i &= f_{\sigma_i}(x_{1:i-1}) \end{aligned}\right\} \quad i = 1, \ldots D.$$

Commonly $f_{\sigma_i}(x_{1:i-1})$ is taken to be the log-standard deviation giving the forward transformation

$$x_i = f_{\sigma_i}\left(x_{1:i-1}\right) z_i + f_{\mu_i}\left(x_{1:i-1}\right), \quad i = 1, \ldots, D, \tag{2.19}$$

where $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. It follows from Equation 2.19 that the forward transformation can only be computed sequentially. However, the inverse transformation

$$\mathbf{z} = \frac{\mathbf{x} - f_{\boldsymbol{\mu}}(\mathbf{x})}{f_{\boldsymbol{\sigma}}(\mathbf{x})}$$

is parallelisable which is required for scalability to $D \gg 1$ [KSJ+16]. Such a transformation is referred to as IAF step and we denote it by $\mathbf{z} = f_{IAF}(\mathbf{x})$. To obtain more expressive posteriors several IAF steps can be composed together to give $\mathbf{z}_t = f_{IAF}^{(t)}(\mathbf{z}_{t-1})$ for $t = 1, \ldots, T$.

Suppose the encoder outputs the distribution $q_\phi(\mathbf{z}_0|\mathbf{x})$, then after T steps of IAF the log density is given by

$$\log q_\phi\left(z_T \mid x\right) = \log q_\phi\left(z_0 \mid x\right) + \sum_{i=1}^{T} \sum_{k=1}^{l} \log \sigma_k^{(i)}, \tag{2.20}$$

where l denotes the dimension of the latent space, the IAF parameters are contained in ϕ and $q_\phi(\cdot)$ is used to denote the density at each step. To conclude we sample from this distribution and pass the samples through the decoder to obtain $p_\theta(\mathbf{x}|\mathbf{z}_T)$. For an overview of the use of IAF within VAE see Figure 2.2.

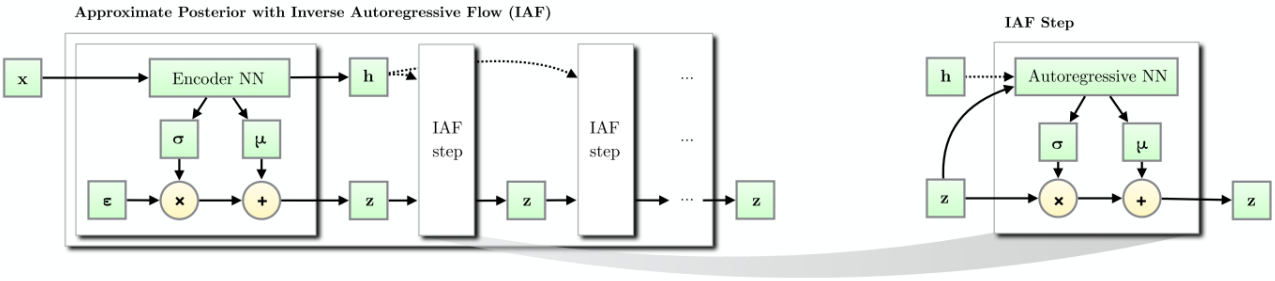

Figure 2.2: Concept VAE with IAF to increase expressivity of the posterior. The final $\mathbf{z}$ is then used as input to the decoder. The Figure is taken from [KSJ+16].

Conditional VAE

Another neat extension of the standard VAE is the conditional VAE [SLY15], which enables the use of additional information, such as categories. It is based on the constraint of the Standard VAE that multimodal distributions are hard to model and even knowing the modes is not taken advantage in the encoder nor the decoder. Specifically assume that in addition to the observations $\mathbf{x}_i$ (e.g. images of clothing items) we have other information $\mathbf{c}_i$ (e.g. the item category like shoe, dress, etc.). The basic idea is to simply condition on $\mathbf{c}$ which yields $q_\phi(\mathbf{z}|\mathbf{x}, \mathbf{c})$ and $p_\theta(\mathbf{x}|\mathbf{z}, \mathbf{c})$ as the variational and inference models of the conditional VAE. The

variational lower bound is straight forwardly computed to be

$$\log p_\theta(\mathbf{x}|\mathbf{c}) - D_{KL}[q_\phi(\mathbf{z}|\mathbf{x},\mathbf{c})\|p_\theta(\mathbf{z}|\mathbf{x},\mathbf{c})] = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x},\mathbf{c})}[\log p_\theta(\mathbf{x}|\mathbf{z},\mathbf{c})] - D_{KL}\left(q_\phi(\mathbf{z}|\mathbf{x},\mathbf{c})\|p_\theta(\mathbf{z}\mid\mathbf{c})\right).$$

This formulation of the VAE enables us to learn multimodal distributions. Furthermore, the conditional VAE can be applied to image labelling. But also in general for prediction. For example the model has originally been proposed for image completion, i.e. given a quadrant ($\mathbf{c}$) of the image it would predict the remaining quadrants ($\mathbf{x}$). The prdiction can for example be obtained by drawing $\mathbf{z}$ and taking the average of the posterior:

$$\mathbf{x}^* = \arg\max_{\mathbf{x}} \frac{1}{L}\sum_{l=1}^{L} p_\theta\left(\mathbf{x}\mid\mathbf{c},\mathbf{z}^{(l)}\right), \mathbf{z}^{(l)} \sim p_\theta(\mathbf{z}\mid\mathbf{c})$$

2.3 VAE on Hilbert Spaces

With more and more data being of functional form we require generalisations of existing methods to functional data. Different such generalisations have been suggested for deep learning models. Mishra et al. proposed $\boldsymbol{\pi}VAE$ which utilisies VAEs for Bayesian inference. $\boldsymbol{\pi}VAE$ learns a basis decomposition of sampled functions and then the latent space is used with the decoder to perform inference yielding state-of-the-art performance on spatial interpolation tasks [MFB20].

$\boldsymbol{\pi}VAE$ is distinguished from a standard VAE by embedding the data into a function space and then learning to reconstruct the linear map from this feature space to the output space of function evaluations.

Assume we have N function draws, where each draw consists of evaluations at M locations. Specifically, the i$^{\text{th}}$ element in the data consists of M pairs $\{(s_i^k, x_i^k)\}_{k=1}^M$ (i $\in$ 1,...,N denoting a pair of (input, output) locations. Here s_i^k and x_i^k could be multidimensional, i.e. $s_i^k \in \mathbb{R}^m$ for $m \in \mathbb{N}$ and similarly for x_i^k, but in this report we restrict ourselves to $s_i^k \in \mathbb{R}$ as this is a common form of functional data. Each input location s_i^k is mapped to a higher dimensional

feature space by the map $\Phi(\cdot) : \mathbb{R} \to \mathbb{R}^d$. The authors propose a combination of an RBF network and a Neural Network but note that any explicit feature representation for an RKHS (Reproducing Kernel Hilbert Space) can be used. The feature space is then mapped to the output space via a linear mapping β by $\hat{x}_i^k = \beta_i^T \Phi(s_i^k)$. Observe that since β is independent of the locations it allows the prediction of function evaluations at arbitrary locations s. Hence an advantage of $\boldsymbol{\pi}VAE$ is that in its general form it allows non equally spaced evaluation locations. Furthermore, importantly note that while the feature map $\Phi(\cdot)$ is shared across locations and for all N samples, the linear mapping is sample specific. I.e. for each of the N samples an individual linear map $\{\beta_i\}_{i=1}^N$ is learnt. Then a VAE takes the linear maps $\{\beta_i\}_{i=1}^N$ as inputs and is trained to encode and decode these. Thus, instead of learning to reconstruct function evaluations (like a standard VAE) πVAE learns to reconstruct the mappings from a feature space to the output space. This gives the following end-to-end formulation [MFB20]:

$$\hat{x}_{e,i}^k = \beta_i^\top \Phi\left(s_i^k\right) \tag{2.21}$$

$$[z_\mu, z_{sd}]^\top = e\left(\eta_e, \beta_i\right) \tag{2.22}$$

$$\mathcal{Z} \sim \mathcal{N}\left(z_\mu, z_{sd}^2 \mathbb{I}\right) \tag{2.23}$$

$$\hat{\beta}_i = d\left(\eta_d, \mathcal{Z}\right) \tag{2.24}$$

$$\hat{x}_{d,i}^k = \hat{\beta}_i^\top \Phi\left(s_i^k\right). \tag{2.25}$$

Learning is performed via maximisation of the ELBO which in this case contains an additional term for the feature map and is given by

$$\underset{\eta_e, \eta_d, \Phi, \beta_i}{\arg\max} \quad p\left(x_i^k \mid \beta_i, s_i^k, \phi, \eta_e\right) + p\left(x_i^k \mid \mathcal{Z}, s_i^k, \phi, \eta_d\right) - \mathrm{KL}\left(\mathcal{N}\left(z_\mu, z_{sd}^2 \mathbb{I}\right) \| \mathcal{N}(0, \mathbb{I})\right).$$

Assuming a Gaussian Likelihood this simplifies to the loss

$$\underset{\eta_e, \eta_d, \Phi, \beta}{\arg\min} \quad \left(x_i^k - \beta_i^T \Phi\left(s_i^k\right)\right)^2 + \left(x_i^k - \hat{\beta}_i^T \Phi\left(s_i^k\right)\right)^2 + \quad \mathrm{KL}\left(\mathcal{N}\left(z_\mu, z_{sd}^2 \mathbb{I}\right) \| \mathcal{N}(0, \mathbb{I})\right).$$

Inference with πVAE is performed using MCMC methods to sample from the posterior. For

more details on πVAE (especially on the inference step) we refer to [MFB20].

Similar to the idea in but only aiming to learn function representations for clustering with an Autoencoder, the authors of [HSWH21] proposed FAE. FAE is an Autoencoder which handles functional data by replacing scalar weights by functional weights and the scalar product by the inner product of $\mathbf{L}^2$ in the first and last layers of the Autoencoder.

In this thesis we consider functional weights given by a linear combination of a set of basis functions $\{\varphi_k\}_{k=1}^B$. Specifically, we define the functional weight $w \in L^2([0,1])$ by

$$w(t) = \sum_{k=1}^{B} c_k \varphi_k(t), \quad c_k \in \mathbb{R}.$$

Hence given the functional input $\mathbf{x} \in L^2([0,1])$ the output of node i in the first layer of the encoder is given by

$$H_E(\mathbf{x}) = \sigma\left(\int_0^1 w_i(t)\mathbf{x}(t)dt + b_i\right) \tag{2.26}$$

$$= \sigma\left(\int_0^1 \left(\sum_{k=1}^{B} w_{ik}\varphi_k(t)\right)\mathbf{x}(t)dt + b_i\right) \tag{2.27}$$

$$= \sigma\left(\sum_{k=1}^{B} w_{ik}\int_0^1 \varphi_k(t)\mathbf{x}(t)dt + b_i\right), \tag{2.28}$$

where w_i is the functional weight, b_i is a bias term and $\sigma(\cdot)$ the activation function e.g. Relu. This resembles a functional layer with functional neurons which were first proposed by Rossi and Conan-Guez [RCG05]. Thereafter, the encoder is composed of standard fully connected dense layers (or just the functional layer with one more dense layer outputting $\log(\boldsymbol{\sigma})$ and $\boldsymbol{\mu}$). Similarly the final layer of the decoder network transforms scalar values $x_1, \ldots, x_B$ into a function of the form

$$H_D(x_1, \ldots, x_B)(t) = \sum_{k=1}^{B} w_k x_k \varphi_k(t) + b.$$

The use of a set of basis functions allows a parsimonious way to learn as we can learn on the basis weights. This is more efficient than the more general approach taken in [HSWH21] as it reduces the number of trainable model parameters and hence reduces the amount of data

needed to optimise the model parameters [GBC16]. Furthermore, basis functions are known to help when dealing with irregular data [KR18]. On the other hand, the choice of basis restricts the modelling capabilities and is and needs to be considered carefully.

We emphasise that this algorithm is designed for functional Data and then can be projected to finite-dimensional data. Hence the method is aligned with GRIP. In the limit as $B \to \infty$ this approach allows us to invoke the universal approximation theorem for neural networks generalised to functional neurons in [RCG05]. Thus, theoretically, we should be able to find optimal q_ϕ and p_θ. However, in practice we are limited by B the number of basis functions. It is crucial to note that the choice of basis is an important aspect of this model since different bases span spaces with different levels of flexibility for the same number of basis functions. Hence in addition to the classical hyperparameters of a neural network such as number of neurons, number of hidden layers, activation function etc. we also have the choice of basis and the number of basis functions.

Concurrently to this thesis, Rao and Reimherr published a novel approach to function-on-function regression with continuous neurons [RR21]. Instead of only considering functional weights in the first and last layer, functional weights throughout the whole network were proposed. Their FBNN is very similar to our approach and is something that should be extended, i.e. VAE with continuous neurons throughout.

Chapter 3

Experiments

In this chapter we present the simulations and results conducted. In the following sections we refer to the samples generated with the model by synthetic data or samples while a generated data set used for training will be referred to as simulated.

3.1 Data Sets

For the purpose of our simulations we will us the following data sets. An overview of the data set sizes is given in Table 3.1.

3.1.1 Sugar spectra data

The sugar spectra data set results from the field of chemometrics. This data set was collected as part of an experiment to determine ash content in sugar from the emission spectra. The data set was downloaded from [Sug] and first analysed in [MNE+98]. For each sample emission spectra are recorded at several wavelengths, 7 in total. 268 samples are collected and a spectra for each sample consists of 571 recorded data points. The spectra for one of the wavelengths are visualised in Figure 3.1.

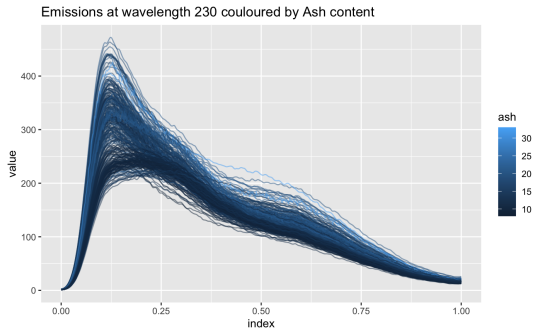

Figure 3.1: Sugar emission spectra at wavelength 230. The spectra are coloured according to the ash content in the corresponding sugar sample.

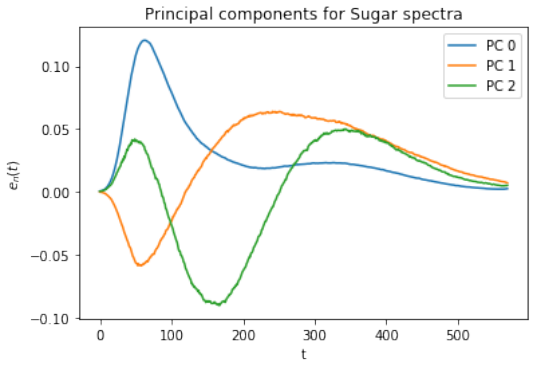

Figure 3.2: First 3 normalised functional principal components for the sugar spectra.

3.1.2 Gridwatch data

This data set is part of publicly available and consists records the electricity consumption. We consider each day as a single functional observation. The measurements are taken every 5 minutes which leads to a total of 288 evaluations per day. In total the data set contains 533 observations. This data is interesting as especially the weekend days lead to a multi-modal distribution. The data is shown in Figure 3.4.

3.1.3 Simulated data

Since both, the sugar spectra data set and the gridwatch data set only contain a smaller number of samples in each category preventing us from reasonably splitting the data into a train and test sets for the analysis or even applying cross validation we additionally generate simulated data for some of our experiments.

For the simulated data set we sample from a Gaussian process prior with mean 0 and covariance function given by the RBF kernel defined by

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right)$$

The length scale parameter σ is set equal to 1. We generate 2000 samples each evaluated at 100 equally spaced grid points between 0 and 4. Some of the sampled functions are shown in

Figure 3.3.

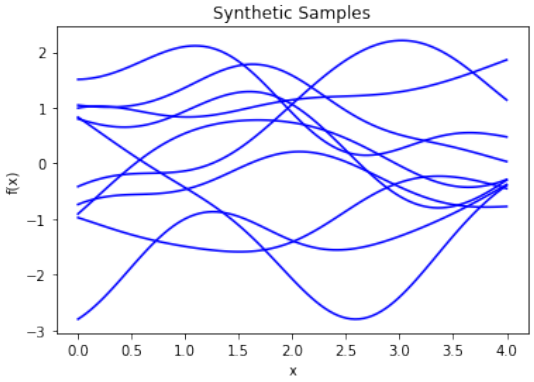

Figure 3.3: Simulated data sampled from Gaussian process prior.

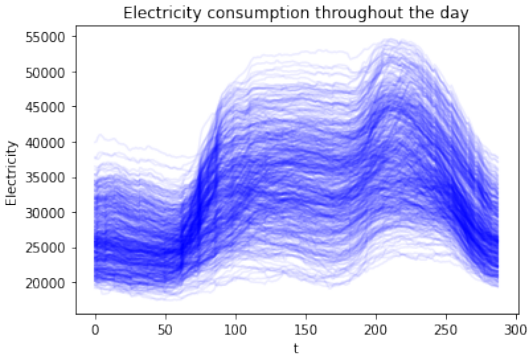

Figure 3.4: Gridwatch data set. The time steps on the x-axis represent 5 minute intervals.

Data set	N	M
Simulated GP	4000	100
Sugar Spectra	268	571
Gridwatch	532	288

Table 3.1: Data set summaries, with N the number of samples and M the number of evaluations per sample.

3.2 Implementation

We note that no implementation of $\boldsymbol{\pi}$VAE is publicly available at the time this thesis was written. The models are implemented in *Python* using *Tensorflow* and *Keras*. For basis expansions we use *Numpy* and *Pycwt* (continuous wavelet transform, *Pywt* for discrete wavelet transform) and the *Signatory* package for the signature transform. The code for VAEs with inverse autoregressive flows is adapted from the Github repo [IAF]. Some problems during training are caused by the scaling of the inputs. It is good practice to scale the inputs before training since for large unscaled inputs the model loss can tend to infinity and convergence can be very slow. However, simply scaling the inputs to be between 0 and 1 via a min-max scaling leads to very noisy samples generated from the inference model. We suspect that this is caused by only scaling the function values and hence disturbing the relation with the derivatives. For

the purpose of this report the scaling parameter is tuned manually, but we note that this issue should be further analysed in future studies.

During model training, where enough data is available to allow a sensible train, validation, test split, the validation loss is monitored for the purpose of early stopping to prevent overfitting. Additionally, common tools to regularise deep learning models are Dropout [SHK+14], Parameter regularisation, and Batch Normalisation [IS15]. Dropout randomly assigns zero weights to a specified percentage of the nodes in order to prevent a small number of nodes from dominating the model. Batch Normalisation increases convergence speed and stabilises the model training by essentially normalising the outputs of a layer.

Note that due to problems in the implementation of the continuous wavelet transform with a convolutional VAE we use the discrete wavelet transform with the Daubechies 2 wavelet instead.

3.3 Model Criticism

Assessing the performance of generative models for functional data is a key component of this report. In the literature generative models are commonly assessed on the following criteria also chosen by [YJvdS19]:

1. Sample Diversity, i.e. are the samples distributed according to the distribution of the data. We will assess the diversity via visualisation of the samples and apply t-SNE [VdMH08] and PCA [BY95] for dimensionality reduction of the functional data (reducing over the time domain).
2. The samples should not be easy to tell apart from the original data. This will be assessed via the MMD between original and synthetic samples as well as the test error of a trained classifier.
3. Usefulness. This will be assessed in one of the applications where we use one of the models to augment the training data in order to improve a classifier.

The classifier to discriminate between generated samples and the true samples consists of a

bidirectional-LSTM a type of bidirectional recurrent neural network [SP97] with long short-term memory [HS97] coupled with a Dense layer (see Section B.3 for an example). Bidirectional-LSTMs have proven to be successful in applications to sequence or time series data [DKOZ05]. The classifier is then trained on 50% of the total data set (half and half split between generated samples and original samples). It is then evaluated on the remaining unseen 50% of the data set and the loss and classification accuracy are reported.

3.4 The choice of basis

As mentioned the choice of basis is crucial and different bases may be suitable for different data. In this section the different bases introduced in Chapter 2 are assessed based on the criteria outlined in the previous section (Section 3.3). For the Gridwatch and simulated data sets we use early stopping with patience 100 during training (maximal number of epochs is 1800) and compare the samples to an unseen test set. In the case of the Gridwatch the (train, validation, test)-split is (0.64, 0.16, 0.2). The corresponding split for the simulated data set is (0.35, 0.15, 0.5). For the sugar spectra no early stopping callback is implemented and the model is trained for full 1600 epochs. The details for the basis are as follows: The parameter of the Matérn kernel is tuned to be $\nu = 1.5$ leaving flexibility for rougher functions. The number of functional principal components in the FPCA basis is chosen so that 95% of the variance is captured. The number of basis functions in the Fourier basis is chosen visually to obtain a good trade-off between smoothing and interpolation.

Convergence of the training for the simulated data set can be observed in Figure 3.5. The obtained samples can be compared with the original in Figure 3.6 for the simulated data and in Figure 3.7 for the sugar spectra. The samples obtained via the FPCA basis appear to be smoother, especially for the spectra. This can be explained by the use of the small number functional principal components as basis functions which are rather smooth (see Figure 3.2). All bases except the Wavelet basis seem to capture the main features of the data and the distribution does indeed appear to represent that of the unseen test sets (besides for the Wavelet basis).

Figure 3.8 visualises the simulated test set and synthetic VAE samples in two dimensions obtained by applying t-SNE. While the Wavelet basis does not show good sample diversity, synthetic points and test points almost overlap perfectly for the other bases. This suggest a good approximation to the true distribution. The same methodology only for the sugar spectra is shown in Figure 3.9. A clear difference in terms of sample diversity can be observed. While the samples obtained from a model with FPCA basis show good diversity and match the original samples well, the Matérn kernel and Fourier bases appear to be more constraint. The same observation is made for the Gridwatch data. This is cause by the ability of FPCA to capture 95% of the variability in the data with a very small number of basis functions for the data sets here considered. Consequently, we expect the samples to match well after the dimensionality reduction. On the other hand we sacrifice flexibility by only using a small number of basis functions. The complete results can be found in Appendix B.

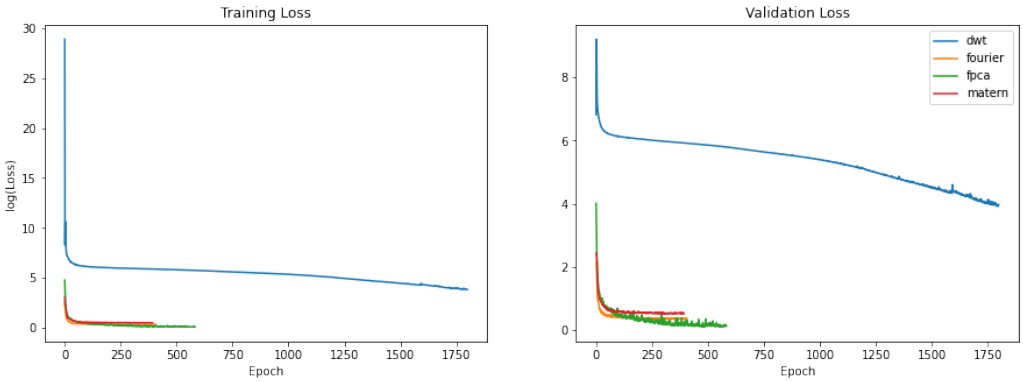

Figure 3.5: Train and validation loss for different basis on the simulated Gaussian process prior samples.

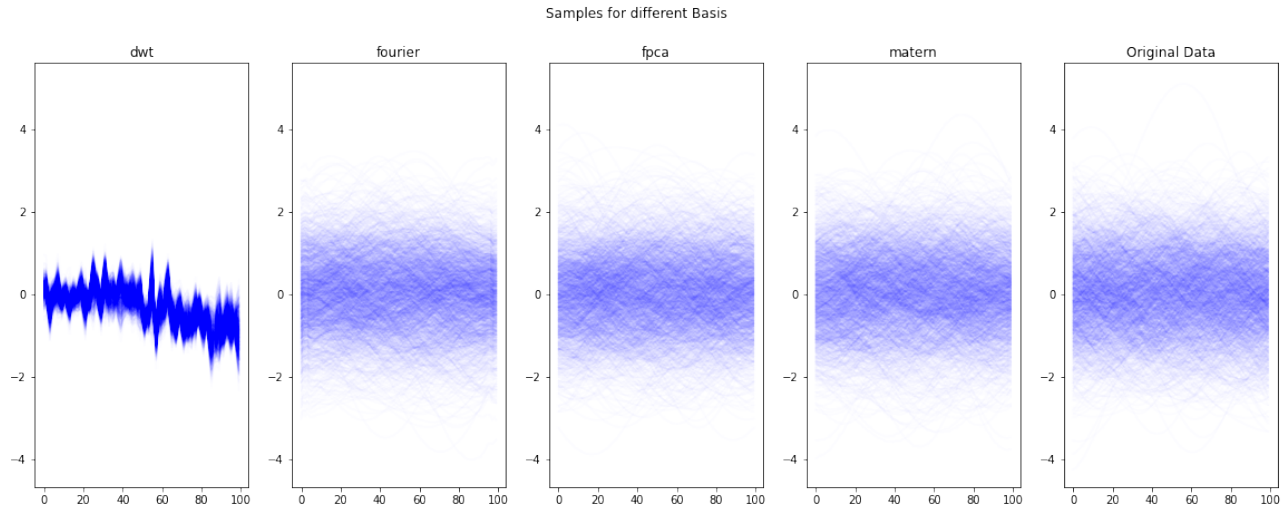

Figure 3.6: Generated synthetic samples for different basis on the simulated Gaussian process prior samples together with the generated test set.

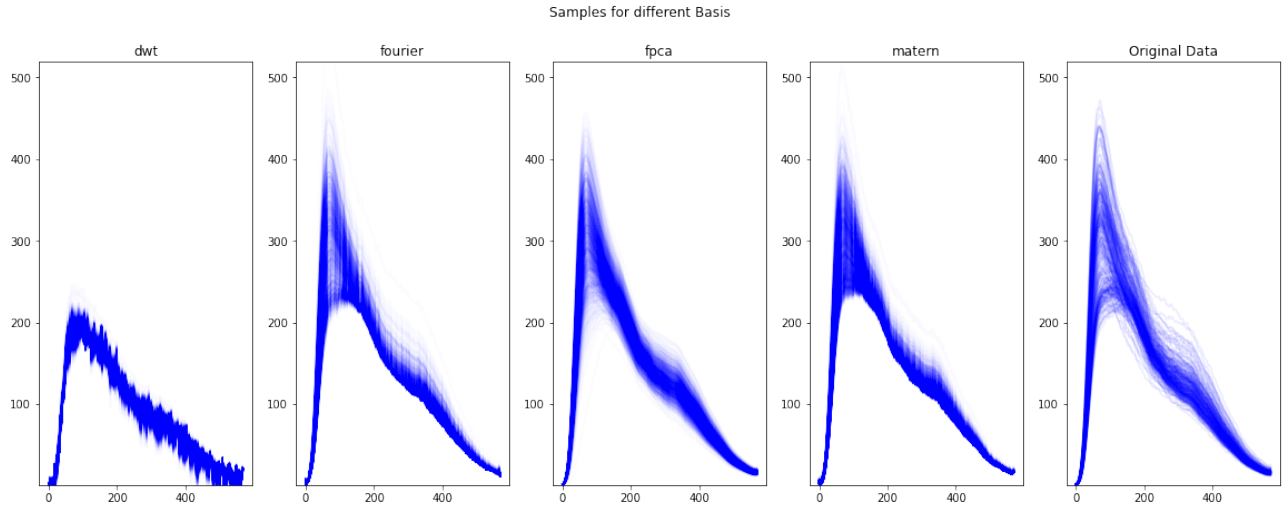

Figure 3.7: Generated samples for different basis on the Sugar spectra together with the original spectra.

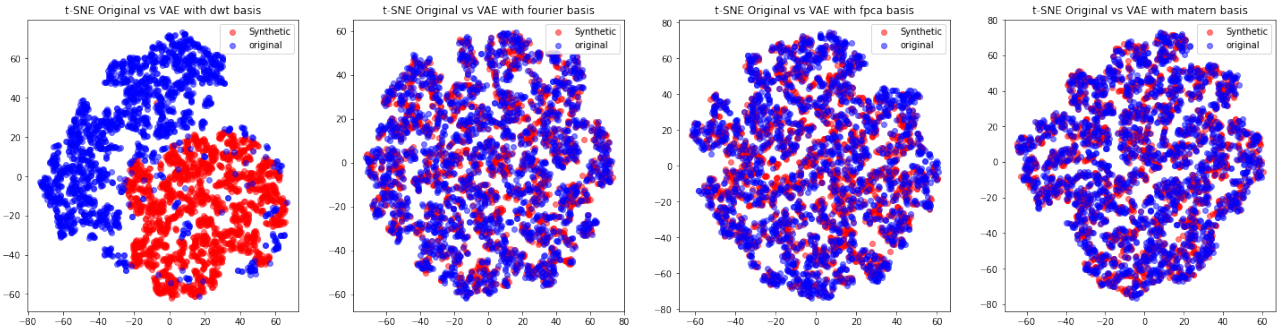

Figure 3.8: Synthetic samples and test data set for simulated GP samples projected into 2-D via t-SNE.

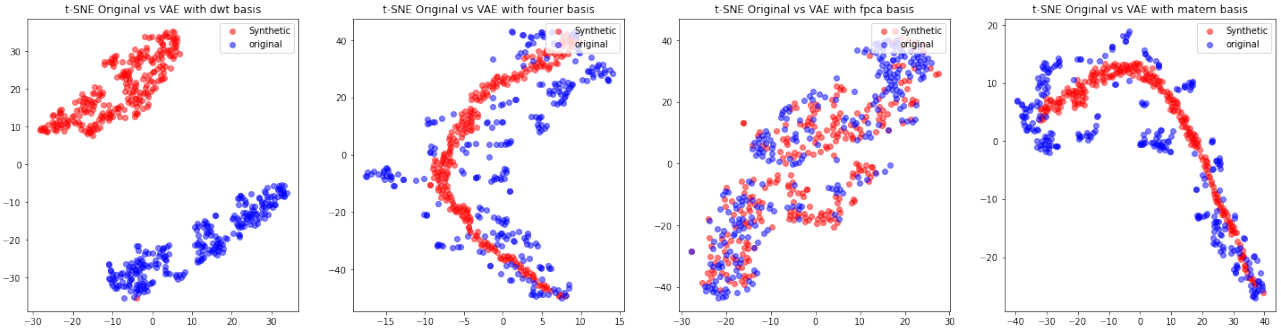

Figure 3.9: Synthetic samples and test data set for Sugar spectra projected into 2-D via t-SNE.

The results of the discriminative comparison are summarised in Table 3.2. On first glance, one notices that no clear best basis can be chosen based on the considered metrics and indeed the suitability does depend on the data set at hand. This is due to the different spaces spanned by the truncated bases. For the sugar spectra data set the FPCA basis yields the best results with big difference. The samples hardest to discriminate with the classifier are obtained from the FPCA and Matérn kernel bases. On the other hand, the Fourier basis performs well in terms of the MMD. It can be observed that for all except the Wavelet basis the lowest accuracy values when discriminating synthetic data from original data are approximately 0.5 which means the classifier is only as good as randomly assigning labels. Note that sometimes the estimate of the $\widehat{MMD}^2_{SQR}$ is uninformative for a comparison as it for example returns 0 for all bases in the case of the Gridwatch data (even for the Wavelet basis which yields samples far below the quality of the other bases).

Basis	Data	Epochs	$\widehat{\text{MMD}}^2_{SQR}$	$\widehat{\text{MMD}}^2_{CEXP}$	Loss	Accuracy
Wavelet	Simulated GP	1800	0.428	0.081	0.108	0.964
	Sugar Spectra	1600	3.16×10^{-45}	0.095	0.495	0.907
	Gridwatch	1259	0	7.4×10^{-8}	0.33	0.99
Fourier	Simulated GP	407	0.004	**0.0001**	0.688	0.534
	Sugar Spectra	1600	$\mathbf{1.41 \times 10^{-104}}$	0.036	0.577	0.701
	Gridwatch	285	0	$\mathbf{3.49 \times 10^{-22}}$	0.697	0.476
FPCA	Simulated GP	583	0.012	0.011	0.655	0.604
	Sugar Spectra	1600	**0**	**0.002**	**0.697**	**0.515**
	Gridwatch	1629	0	6.67×10^{-15}	**0.693**	**0.494**
Matérn	Simulated GP	393	**0.001**	0.001	**0.692**	**0.519**
	Sugar Spectra	1600	5.02×10^{-65}	0.045	0.640	0.746
	Gridwatch	405	0	3.34×10^{-10}	**0.694**	**0.5**

Table 3.2: Results of the discriminative comparison: The estimated MMD^2 for the different Bases and Datasets is reported. In the case of the Synthetic GP Data set the MMD estimates between an unseen test set and the VAE samples are reported. For the Sugar spectra data set the estimated MMD between the original data set and 2000 VAE samples is reported. Additionally, the average test accuracy and average validation loss over 5 independent runs of training the LSTM classifier for 20 epochs are reported.

Furthermore, observe that early stopping kicks in after only 300-400 training epochs for the Fourier and Matérn kernel bases compared to approximately 600 and 1600 for the FPCA basis. The models trained here are reasonable small and fast to train. However, this is an important consideration when training larger and deeper model architectures. Taking the different aspects into account, we emphasise once again that the choice of basis is dependent on the data set we wish to model. Based on our simulations and implementation the Wavelet transform is clearly the worst candidate. Both, the FPCA basis and the Matérn kernel basis, perform very well, however, the visualisations show more diversity in the case of FPCA. We end this section with a discussion of the suitability of the signature transform.

3.4.1 Signature transform

The insertion algorithm is the current state-of-the art for inversion of the signature transform. By inserting elements in lower levels to approximate higher levels in the signature, errors in the signature reconstruction have a strong influence on the final output after inversion. Experiments showed that reconstructions from lower depth signatures were more robust to noise (Figure

B.14). Additionally, to reconstruct a function input with a large number of observations to reasonable detail a high depth of the signature is required resulting in very high dimensional feature spaces. Specifically, the signature up to depth ℓ of a path in $\mathbb{R}^d$ has d^ℓ terms. Since we at least consider paths in $\mathbb{R}^2$ (locations, evaluations) this is a clear bottleneck. To avoid these issues we applied the signature transform and inversion to smaller path segments and concatenated the resulting reconstructions. This lead to better results (Figure B.15) but not comparable to the bases discussed in the previous section. Finally, since the signature transform loses the information of translation the start point needs to be passed into the inversion algorithm. Therefore, this might be a case for the conditional VAE where the start point is passed as condition.

While in our case of dense functional data with $M \geq 100$ for all data sets the signature transform did not prove suitable so far but we recognise its strong potential. In particular, for sparse functional data where the mentioned bottlenecks should not appear to be a problem.

3.5 Sensitivity Analysis

To analyse the influence of the latent dimension we perform a sensitivity analysis. We use the simulated data set and train the VAE with Matérn kernel basis for 500 epochs. The dimension of the latent variable is varied from 2 to 30 and for each 2000 synthetic samples are generated and compared to the unseen test set via the MMD. The results are presented in Figure 3.10. The plot shows that for the simulated data set a latent dimension of 5 is sensible. The significant increase for the FPCA basis is caused by the small number of basis functions so that a high latent dimension leads unnecessary complexity and worse performance. The corresponding results for the Gridwatch and sugar spectra data can be found in Appendix in Section B.4 and show more robustness for the FPCA based basis.

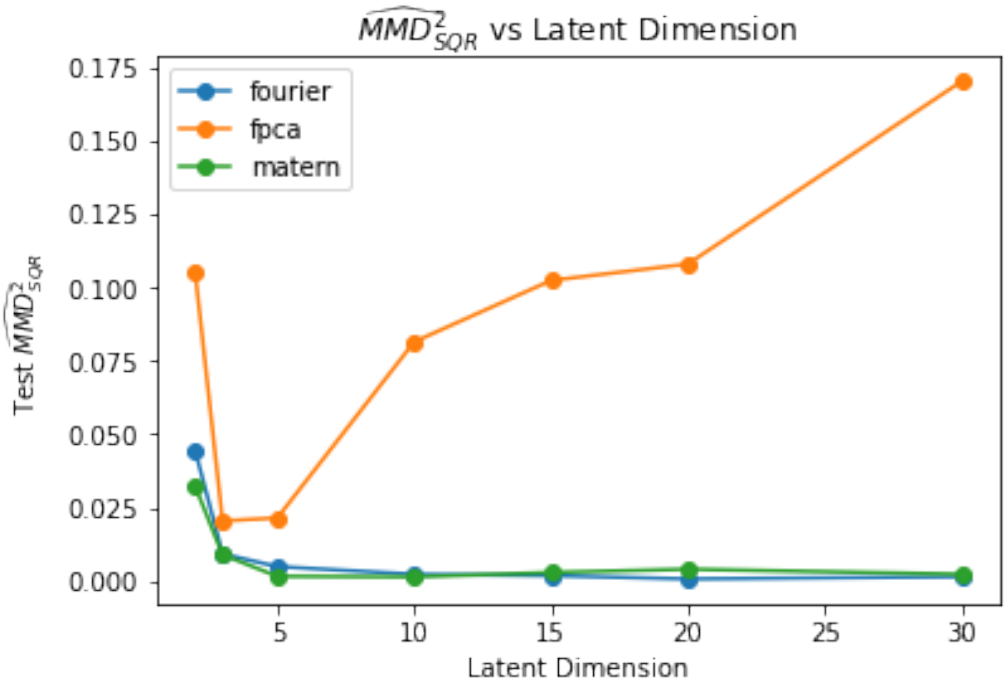

Figure 3.10: Simulated data sampled from Gaussian process prior. The dimensionality of the latent variable is varied from 2 to 30 and for each value the VAE is trained for 500 epochs with each basis. The estimated MMD between an unseen test set and the synthetic generated samples is reported.

3.6 Comparison between standard VAE and VAE with IAF

As discussed in Chapter 2 an improvement to the standard VAE is the use of inverse autoregressive flows to obtain a more flexible model which has an ELBO closer to the true likelihood. In this comparison we drop the wavelet and Fourier basis since they were outperformed by the basis based on FPCA and the Matérn kernel. We train the VAE with 4 IAF steps, a latent dimension of 5 (see B.5 for an example architecture) and report the results of a discriminative analysis in comparison to the standard VAE in Table 3.3. The discriminative comparison shows that for both bases on the simulated data and the sugar spectra data the VAE with IAF performs actually worse than the standard VAE. The performance gap is the most significant

on the simulated data set. For the simulated data this observation is supported by the visualisations in Figures 3.11 and 3.12 for Matérn kernel and FPCA based bases respectively. The corresponding Figures for the sugar spectra data set are Figures 3.13 and 3.14. Here, similarly to standard VAE, a strong difference in terms of sample diversity between the two bases can be observed with the samples using the FPCA based basis matching the data significantly better. The distributions for those data sets are not very complex which is why the standard model performs well. Additionally, when introducing IAF into the model we add a large number of trainable parameters. Hence the IAF model requires more training and is harder to optimise. This explains the decrease in performance for the presented examples. Only for the slightly more complex Gridwatch data set a small performance improvement can be observed by introducing IAF steps. While using the FPCA based basis yields good sample diversity (see Figure 3.16), the projection into 2-D for the Matérn kernel based basis does not coincide for synthetic and original data (see Figure 3.15). The discriminative comparison yields good results nevertheless (no drastic changes when increasing the complexity of the classifier significantly). For all simulations conducted the corresponding projections into 2-D via FPCA can be found in the Appendix (see Section B.5) and confirm the observations made.

Data set	Basis	Model type	$\widehat{\text{MMD}}^2_{\text{CEXP}}$	Loss	Accuracy
Simulated GP	FPCA	IAF	0.3	0.2455	0.9025
		Standard	0.011	0.655	0.604
	Matérn	IAF	0.2605	0.3409	0.8505
		Standard	0.001	0.692	0.519
Sugar	FPCA	IAF	0.0024	0.6961	0.5261
		Standard	0.002	0.697	0.515
	Matérn	IAF	0.23	0.6296	0.8209
		Standard	0.045	0.640	0.746
Gridwatch	FPCA	IAF	1.57×10^{-97}	0.6994	0.4756
		Standard	1.1×10^{-15}	0.6932	0.5
	Matérn	IAF	8.16×10^{-198}	0.6978	0.4756
		Standard	3.34×10^{-10}	0.694	0.5

Table 3.3: Simulation results for the discriminative comparison of standard VAE and the VAE with IAF. Loss and accuracy are the averages of 5 independent training runs with a different train test split each run. The estimates of MMD^2 are computed based on 2000 synthetic samples and computed with respect to the original data sets.

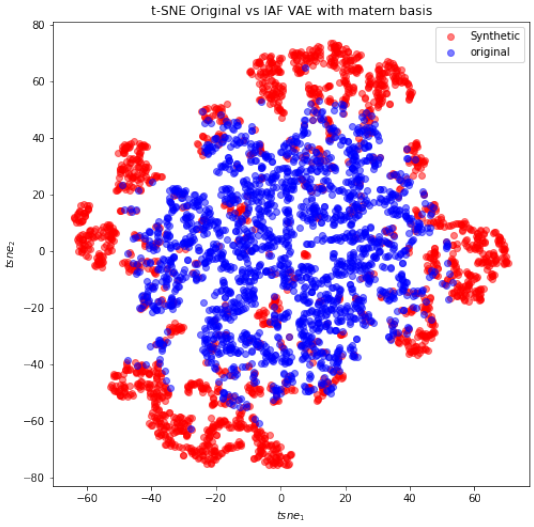

Figure 3.11: Synthetic samples from VAE with IAF with Matérn kernel based basis and test data set for simulated GP samples projected into 2-D via t-SNE.

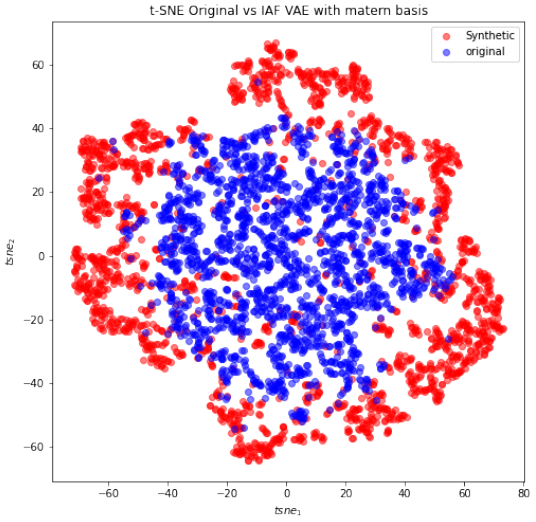

Figure 3.12: Synthetic samples from VAE with IAF with FPCA based basis and test data set for simulated GP samples projected into 2-D via t-SNE.

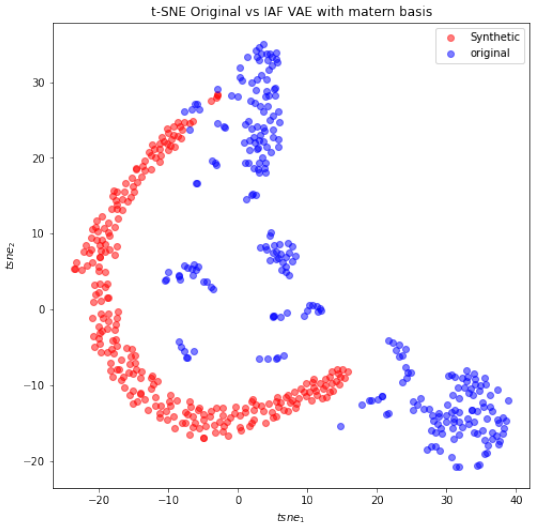

Figure 3.13: Synthetic samples from VAE with IAF with Matérn kernel based basis and sugar spectra data set projected into 2-D via t-SNE.

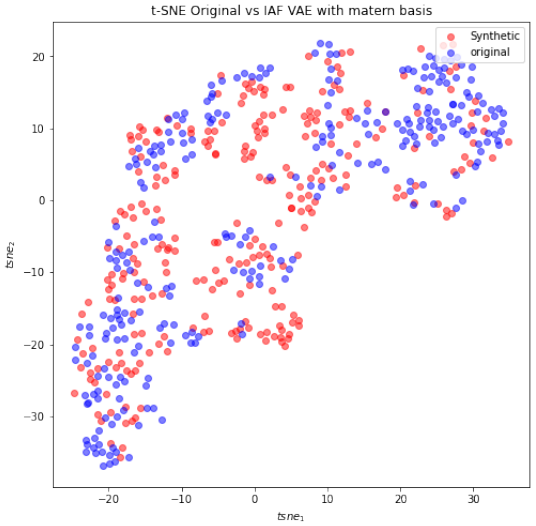

Figure 3.14: Synthetic samples from VAE with IAF with FPCA based basis and sugar spectra data set projected into 2-D via t-SNE.

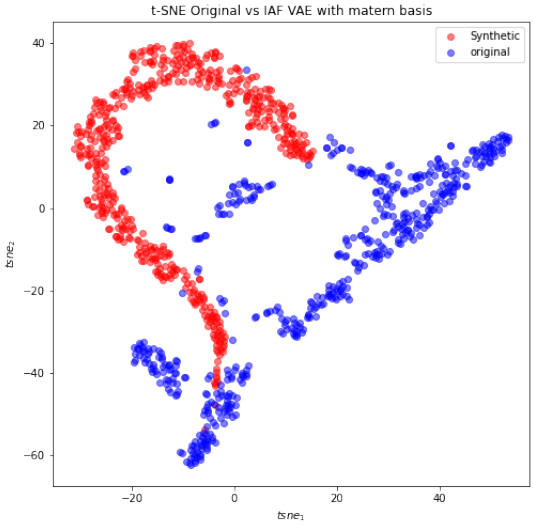

Figure 3.15: Synthetic samples from VAE with IAF with Matérn kernel based basis and Gridwatch data set projected into 2-D via t-SNE.

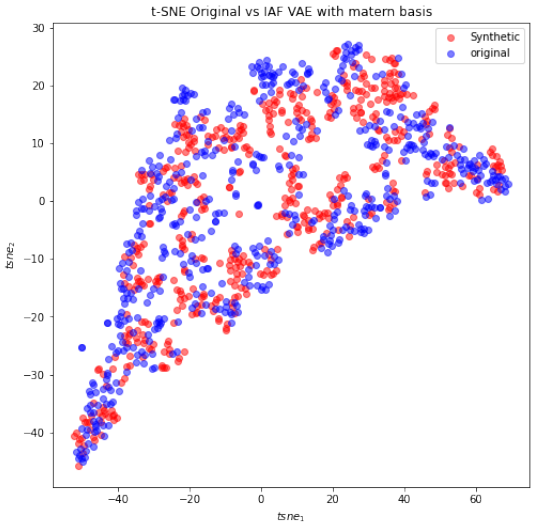

Figure 3.16: Synthetic samples from VAE with IAF with FPCA based basis and Gridwatch data set projected into 2-D via t-SNE.

3.7 Mode collapse and Conditional VAE

A common issue for VAE on multi modal distributions is mode collapse. The model will pick up and be able to synthesise samples for around the dominant modes of the distribution but miss other nodes which might belong to smaller classes. To this end we generalise the standard VAE on Hilbert spaces to the conditional model passing the category $\mathbf{c}$ of an input as conditional information. Since this input is not compatible with the functional layer, in addition to the functional neurons we have added a standard neuron with weight 1, bias 0, and linear activation function in the first layer to simply output $\mathbf{c}$. It is then treated as any other output of the first layer. In the decoder network $\mathbf{c}$ is appended to the sampled latent variable $\mathbf{z}$. We choose one-hot encoding to encode the categorical information.

Thus far we have only considered the sugar spectra at a single wave length. Here we treat each wavelength as its own category. The categories are balanced with 268 samples each. For the Gridwatch data the categories are represented by the day of the week. The categories are not

too unbalanced but considering that the two weekend days mainly differ from the weekdays an imbalance is present (see Table 3.4).

Weekday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday	Sunday
Count	62	61	73	64	81	96	95

Table 3.4: Category count for the Gridwatch data set.

For the comparison we train the conditional and the standard model on the training set and monitor the validation loss. The model architectures are equivalent up to the conditional input. The latent dimension is taken to be 5 and the split into train, validation and test set is determined by the proportions 0.5,0.2,0.3. The reported results are then obtained by a comparison to the test set. To ensure that the number of samples in each category are proportionally represented in the training and test set we use a stratified split.

The simulations are again run for all bases but we expect the FPCA and Matérn kernel bases to perform well due to reasonable performance on the multi-modal Gridwatch data and quick convergence in Section 3.4. A visual comparison to assess the diversity is shown for the sugar spectra data with FPCA basis in Figures 3.17 - 3.19 and for the Gridwatch data with Fourier basis in Figures 3.20 - 3.22. We observe a clear difference in sample diversity (see Figures 3.19 and 3.22 for sugar and Gridwatch data respectively). For both data sets the conditional VAE generates more diverse samples which match the distribution significantly better. The different modes are especially strongly separated in the sugar spectra data and the conditional VAE captures all modes while for the standard model two modes seem to have collapsed (see Figure 3.19). These observations are repeated for the other bases and when reducing the dimension via FPCA instead of t-SNE similar results are obtained.

The results of the discriminative comparison are summarised in Table 3.5. Even though the visual comparison has clearly favoured the conditional model on the sugar spectra data a lower $\widehat{MMD}^2_{CEXP}$ value is obtained with the samples from the standard model. Note that in terms of the test accuracy of the classifier the conditional model performs slightly better than the standard model. For the Gridwatch data the conditional model obtains a consistently smaller

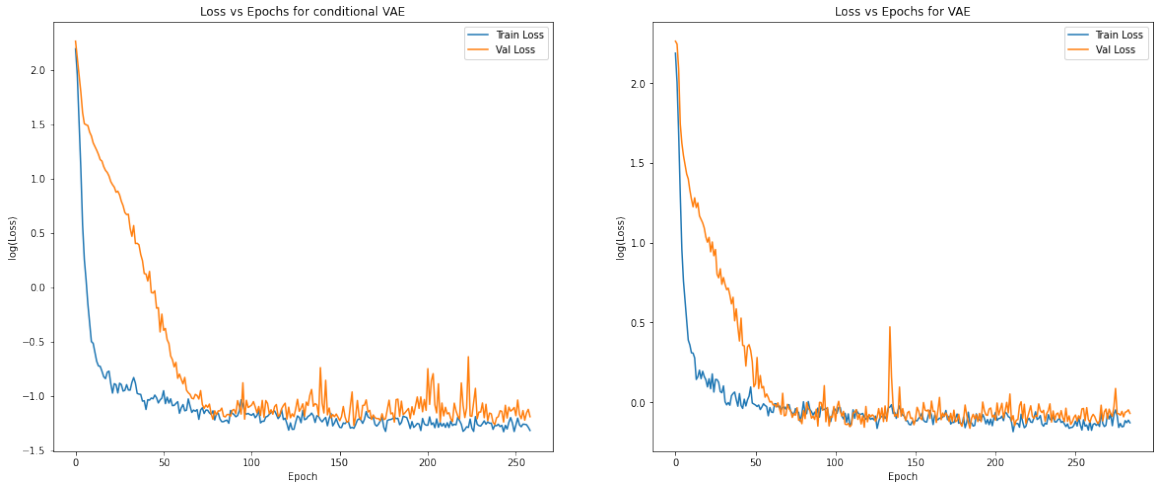

Figure 3.17: Training and validation loss for conditional and standard model trained on sugar spectra data.

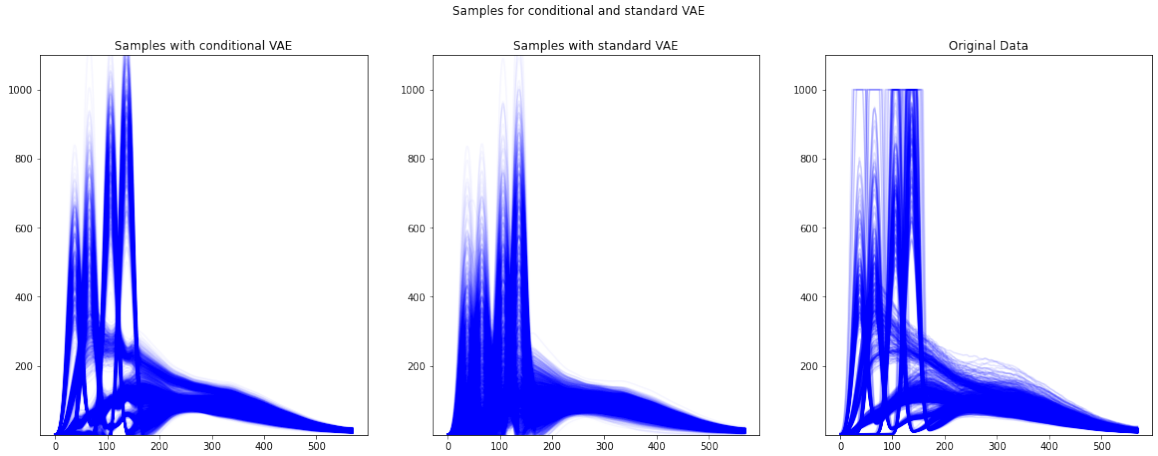

Figure 3.18: Samples from conditional and standard model with FPCA basis. The samples are obtained after training the models on the training split of the sugar spectra data set.

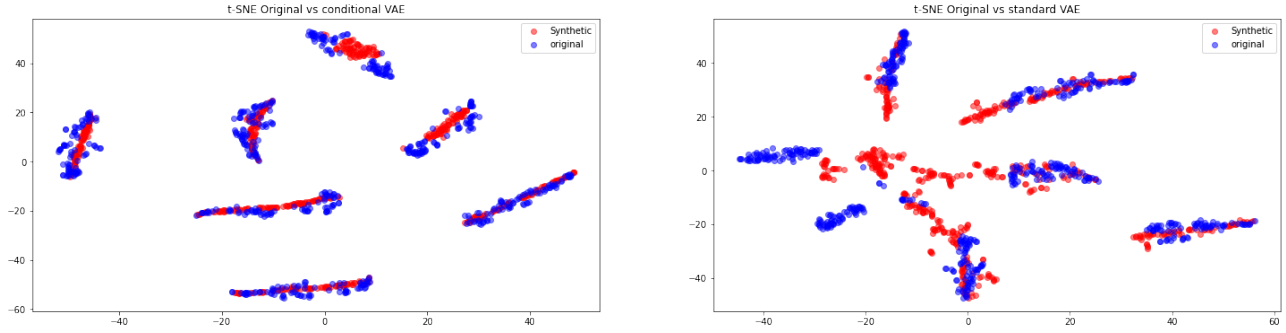

Figure 3.19: Samples from conditional and standard model with FPCA basis. The samples are obtained after training the models on the training split of the sugar spectra data set. And then projected into 2-D via t-SNE. Here the original data denotes a test set unseen during training.

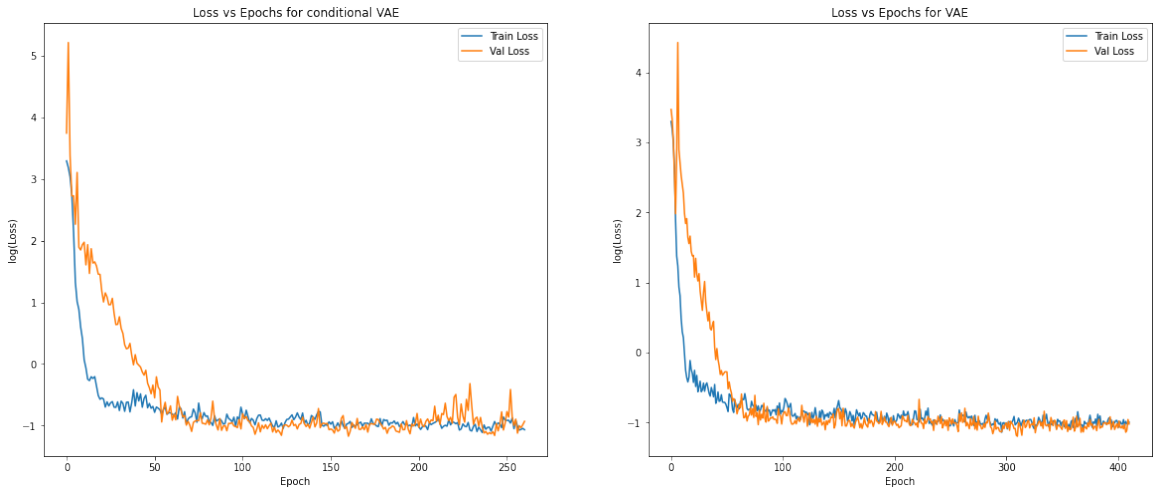

Figure 3.20: Training and validation loss for conditional and standard model trained on Gridwatch data.

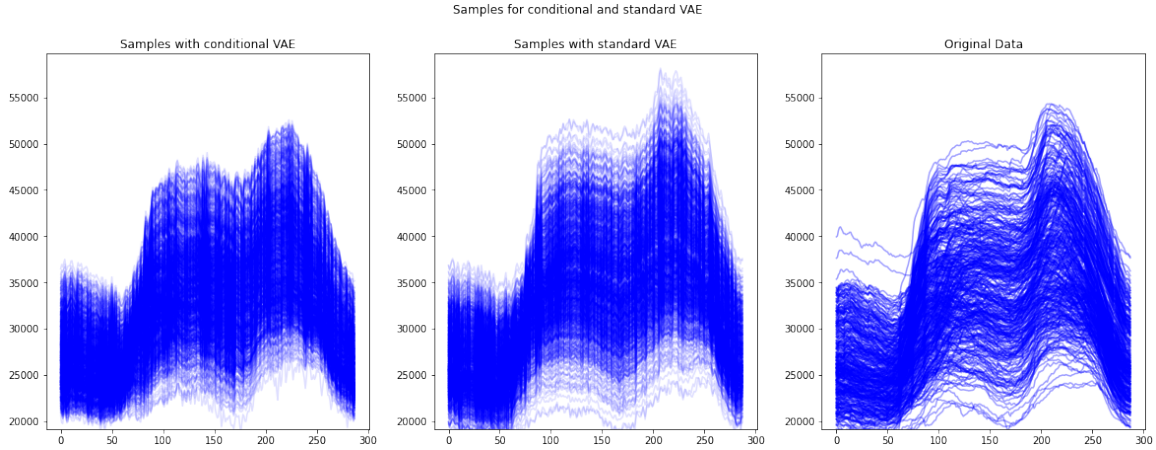

Figure 3.21: Samples from conditional and standard model with Fourier basis. The samples are obtained after training the models on the training split of the Gridwatch data set.

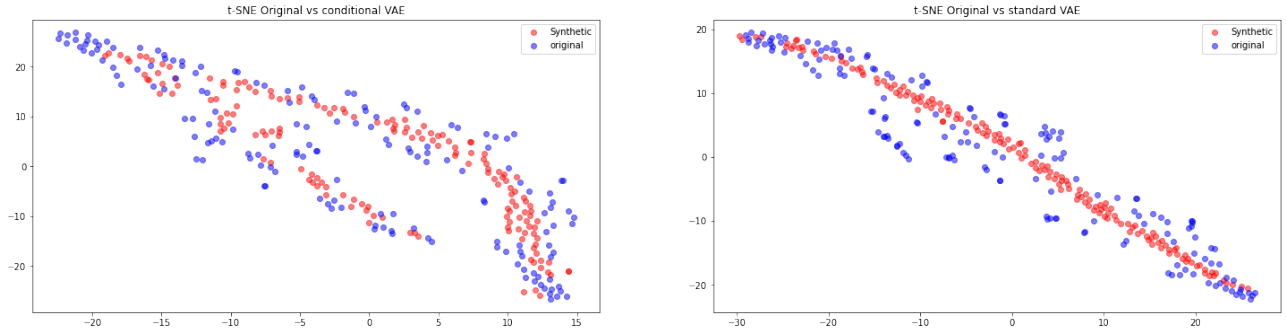

Figure 3.22: Samples from conditional and standard model with Fourier basis. The samples are obtained after training the models on the training split of the Gridwatch data set. And then projected into 2-D via t-SNE. Here the original data denotes a test set unseen during training.

$\widehat{MMD}^2_{CEXP}$. The trained classifier is not able to distinguish the samples with a test accuracy of 0.5 for both the conditional and standard model.

In conclusion the conditional model shows far superior sample diversity and is hence more suitable for multimodal distributions, even though this difference is not significantly reflected in the discriminative comparison.

Data set	**Basis**	**Model type**	$\widehat{\mathbf{MMD}}^{\mathbf{2}}_{\mathbf{CEXP}}$	**Loss**	**Accuracy**
Sugar	FPCA	Conditional	0.004	0.369	0.792
		Standard	0.001	0.416	0.799
	Matérn	Conditional	0.019	0.163	0.954
		Standard	0.003	0.133	0.96
	Fourier	Conditional	0.012	0.055	0.994
		Standard	0.007	0.024	0.999
Gridwatch	FPCA	Conditional	5.19×10^{-163}	0.6932	0.5
		Standard	3.8×10^{-93}	0.6932	0.5
	Matérn	Conditional	8.16×10^{-198}	0.6932	0.5
		Standard	7.74×10^{-73}	0.6932	0.5
	Fourier	Conditional	1.64×10^{-265}	0.6932	0.5
		Standard	7.34×10^{-70}	0.6932	0.5

Table 3.5: Simulation results for the discriminative comparison of conditional and standard models with different basis. Loss and accuracy are the averages of 5 independent training runs with a different train test split each run. The estimates of MMD^2 are computed based on 3939 and 1116 samples for the sugar spectra and Gridwatch data sets respectively (keeping category counts proportional).

3.8 Improving functional classifiers with conditional VAE on Hilbert spaces

A common technique to improve the performance and generalisability of a deep learning model, especially classifiers, is data augmentation (see e.g. [FADK+18]). This technique is especially useful for unbalanced classification problems which are common with applications such as credit card fraud detection or cyber security breaches.

So far we have omitted the fact that the sugar spectra data set has an additional response variable, the ash content of the sugar sample emitting the spectra. A histogram of the ash content is shown in Figure 3.23. The highest measured response is 33. We split the data set into to classes, the samples with high ash content > 18 and low ash content ≤ 18. This leads to the classes with 31 and 237 samples in the high and low category respectively (see Figure 3.24).

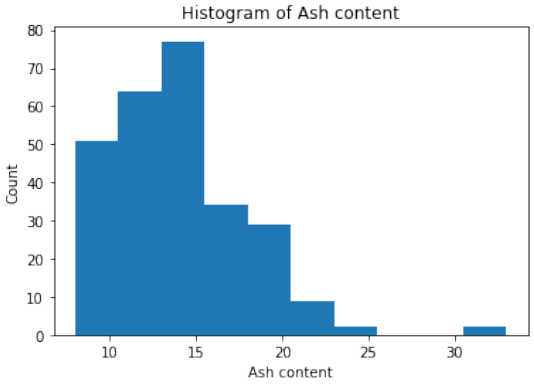

Figure 3.23: Histogram of ash content in sugar samples.

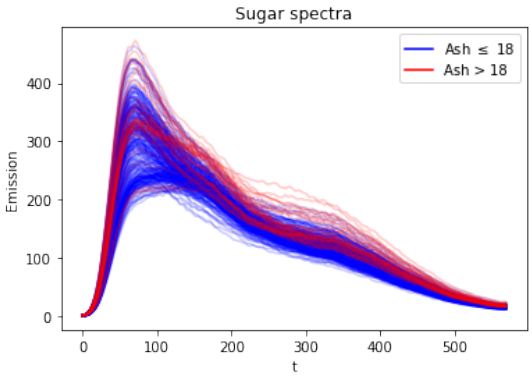

Figure 3.24: Sugar spectra at wavelength 325 coloured corresponding to high or low ash content.

Let us consider the following toy classification problem: The task for the model is to classify a sugar sample as having high or low ash content based on the corresponding spectrum at wavelength 325. This is an unbalanced classification problem since the data set contains only 31 samples with ash content higher than 18.

We split the data set into a train and test set. The test set consists of 16 samples with high ash content and 16 samples with low ash content. The train set contains the remaining 15 samples with high ash content and 221 samples with low ash content. Note the imbalance present in the training data which will lead to the classifier simply predicting all class labels as low ash content. I.e. naively training the classifier we expect an accuracy of 0.5 on the test set.

The classifier consists of a Bidirectional-LSTM layer followed by a dense layer. As loss the Categorical cross entropy loss is used and the model is optimised with the Adam algorithm. The model is trained on the train set and then evaluated on the test set with accuracy as metric. One training loop consists of 6 epochs (this has proven to be enough for the classifier to converge). Once a training loop is complete we augment the training data. Specifically we generate samples belonging to the underrepresented category with a conditional VAE with FPCA basis and add the synthetic samples to the train set. Then the classifier is reinitialised and trained for 6 epochs. We keep adding synthetic data until the number of samples in each category is equal. The resulting averaged test accuracies over 4 independent runs of this process are shown in Figure 3.25. As expected, without data augmentation the accuracy on the balanced test set is 0.5. As we increase the proportion of samples with high ash content by adding the synthetic data an increase in the averaged test accuracy to 0.61 can be observed. The jump is observed as the number of high ash content samples in the training data set reaches 70% of the number of samples with low ash content. Augmenting the data set further only leads to a small increase, which is common for this approach. This is a remarkable result as only using the same data we have managed to significantly increase classifier. It shows the usefulness of the synthetic samples and hence the applicability of the conditional VAE on Hilbert spaces.

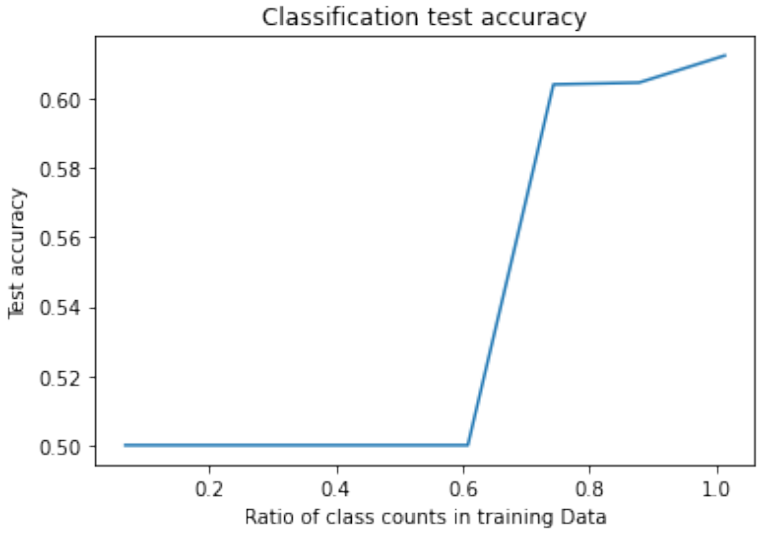

Figure 3.25: Averaged accuracy of 4 independent runs on test set plotted against the ratio of number of samples containing a high ash content and number of samples containing a low ash content.

3.9 Conditional VAE on Hilbert spaces for multivariate functional data synthesis

In section 3.8 we have discussed the application of conditional VAE to augment an underrepresented class of a data set in order to improve the generalisability of a classifier. However, instead of passing one-hot encoded category labels as conditional information we could pass a function. We propose the conditional VAE on Hilbert spaces as a generative model for multivariate functional data.

Consider the bivariate functional data set $(\mathbf{x_1}^{(i)}, \mathbf{x_2}^{(i)})_{i=1}^{N}$ where $\mathbf{x_j}^{(i)} \in L^2([0,1])$ for j=1,2 and $i = 1, \ldots, N$. To generate samples from the joint distribution we first train a standard VAE on Hilbert spaces to target the marginal distribution of $\mathbf{x_1}$. Then a conditional model is trained to generate samples $\mathbf{x_2}|\mathbf{x_1}$.

The idea is straight forward to implement with the only bootleneck being the high dimensionality of functional data. We avoid this by passing the FPCA coefficients needed of a sample $\mathbf{x_1}$

as conditional information to the conditional model.

We apply this approach to the sugar spectra data set. Even though we considered all observation as independent and the distinct wavelengths as different modes of the distribution for the purpose of analysing mode collapse, the data set is a multivariate functional data set. The first 10 samples for the wavelengths 325 and 290 are visualised in Figure 3.26.

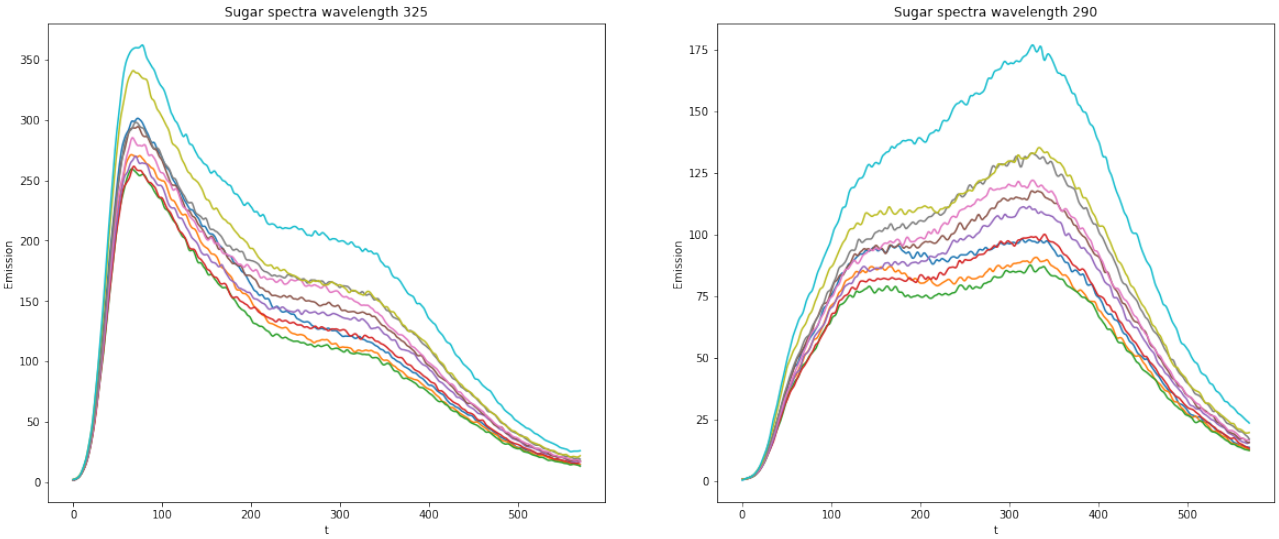

Figure 3.26: Some spectra for wavelength 325 (left) and 290 (right).

Due to the strong performance of the FPCA basis on the sugar spectra data (see section 3.4) it is the basis of choice here. To compare the bivariate samples to the original distribution we concatenate $\mathbf{x_1}$ and $\mathbf{x_2}$ reduce over the time dimension with t-SNE to visualise the synthetic and original samples. The results are shown in Figure 3.27. The plot shows that the synthetic samples seem to match the original samples fairly well. It can be observed, that the synthetic data is slightly over represented in the central area of the plot and lacking support on the edges such as bottom left. As before we train Bidirectional-LSTM classifier, this time taking as input the bivariate time series. Averaging over 5 independent training runs we report an average test loss of 0.678 and an average test accuracy of 0.624. This suggests that the synthetic samples are indeed reasonably similar to the original bivariate data. However, we note that this approach of conditioning on synthetic samples is suboptimal and in general a model to generate bivariate samples directly from the joint distribution should be preferred.

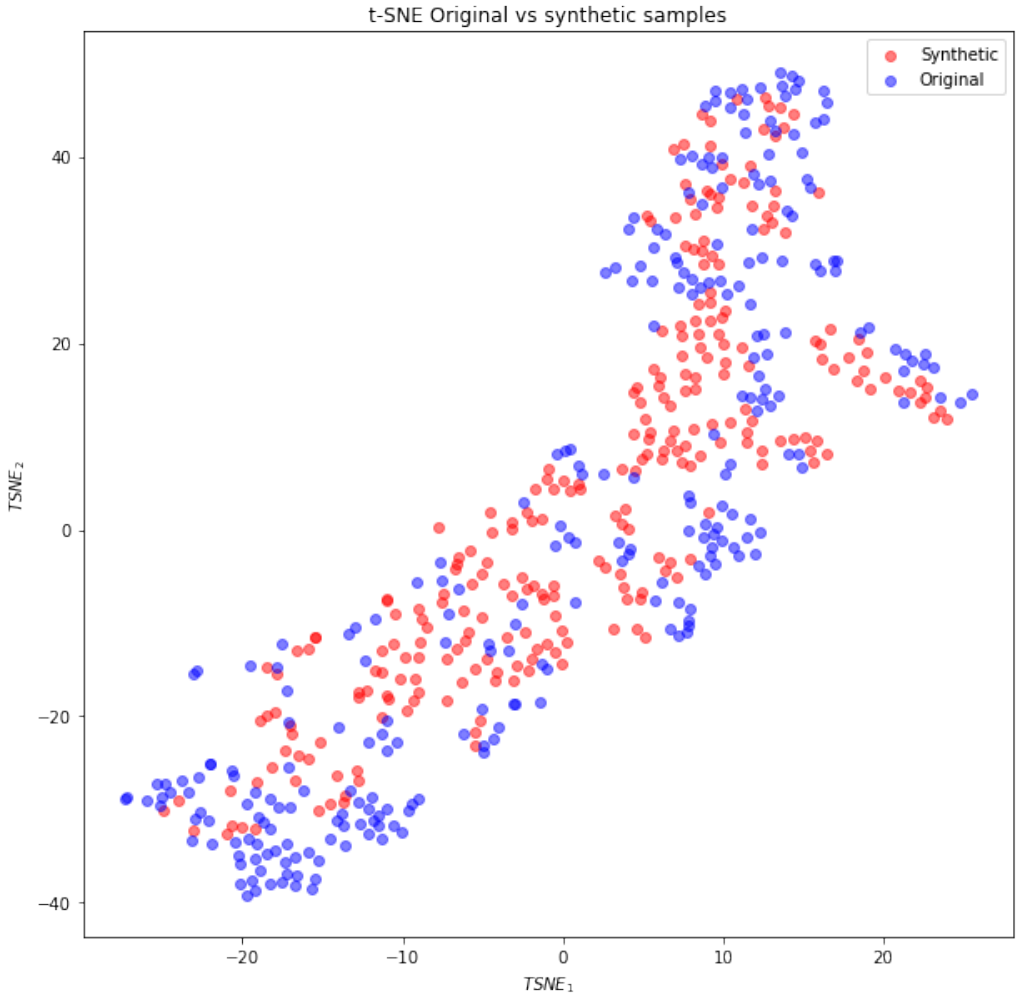

Figure 3.27: Concatenated $\mathbf{x_1}$ and $\mathbf{x_2}$ along time axis for synthetic samples and original, then reducing along time dimension with t-SNE and plotting the projection into 2-D.

Chapter 4

Conclusion

4.1 Summary of Thesis

We now summarise the work conducted as part of this thesis:

- Presented theory on Functional Data analysis in combination with deep generative modelling.
- Defined VAE on Hilbert spaces and gave justification based on GRIP (to the best knowledge of the author not seen so far).
- Extended VAE on Hilbert spaces with IAF steps.
- Extended VAE on Hilbert spaces to a conditional model.
- Analysed Wavelet basis, Fourier basis, FPCA based basis, Matérn kernel induced basis and the Signature for the VAE on Hilbert spaces by applying the model to three different data sets. A visual and discriminative comparison of the synthetic samples is conducted.
- Conducted a sensitivity analysis to analyse the robustness of the model with different bases.
- Compared the VAE on Hilbert spaces with and without IAF steps.
- Analysed mode collapse for VAE on Hilbert spaces and demonstrated improved performance for multimodal distributions with the conditional model.

- Demonstrated application of the conditional model by augmenting an unbalanced data set to improve the performance of a classifier.
- Presented the use of the conditional model to sample from a bivariate distribution.

4.2 Key results

The analysis of the different bases showed that both Wavelets and the signature transform are not suitable for the data sets here considered. The Fourier basis, FPCA based basis, and Matérn kernel induced basis performed well in the discriminative comparison. Especially, the Matérn kernel induced basis and the FPCA based basis stood out. Both showed good performance on the Gridwatch data and additionally the first performed well on the simulated data while the second was strong on the Sugar spectra data. The visual comparison of the sample diversity showed good results for the simulated data regardless of bases (aside from Wavelet and Signature). However, on the Sugar spectra data and the Gridwatch data the FPCA based basis resulted in synthetic samples which matched the original data significantly better than the less diverse synthetic samples obtained with the other bases.

Generally, a latent dimension of 5 was sufficient to model the data sets at hand and no significant performance improvement for higher latent dimensions was observed. The models with Matérn kernel induced basis and Fourier basis were robust under the latent dimension while on the simulated data the FPCA based model suffered performance loss when increasing the latent dimension to above 5.

Introducing IAF steps into the model lead to worse performance on the the Sugar spectra data and especially the simulated data set. The performance on the Gridwatch data slightly improved.

The most important aspect of the thesis is the conditional model. While in the multimodal setting the standard VAE on Hilbert spaces suffered from mode collapse we improved the performance by conditioning on the mode which a sample belongs to. We demonstrated the usefulness of the conditional model by applying it to an unbalanced data set. We obtained a

significant performance improvement of a classifier by augmenting its training data set with synthetic samples from the conditional model. Considering that the conditional VAE on Hilbert spaces was trained on the same data as the classifier, i.e. no additional data was used, this is a truly remarkable result.

4.3 Future Work

The opportunities for future work are plentiful and exciting. First, one could explore other basis such as the popular Splines. A comparison of the VAE on Hilbert spaces with state-of-the art time series generation models such as WaveNet [ODZ+16] or TimeGAN [YJvdS19] could be conducted, although we note that being designed for functional data rather than time series depending on the data set we expect to see better performance from the time series models. Another possibility is to explore this methodology for sparse functional data. To this end we might want to reconsider the signature transform. Furthermore, the suitability of the standard normal distribution as prior could be investigated since in higher dimensions the normal distribution behaves counter-intuitively. The most interesting option is to adapt the VAE to being fully functional in the sense of using functional neurons throughout(see [RR21]) and adapt the sampling layer.

Bibliography

[ACE10] John A. D. Aston, Jeng-Min Chiou, and Jonathan P. Evans. Linguistic pitch analysis using functional principal component mixed effect models. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 59(2), 3 2010.

[AG93] Alkhiezer and Glazman. *Theory of linear operators in Hilbert space*. Dover Publications Inc., New York, 1993.

[Aro50] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American Mathematical Society*, 68(3), 5 1950.

[BB94] J.N. Bradley and C.M. Brislawn. The wavelet/scalar quantization compression standard for digital fingerprint images. In *Proceedings of IEEE International Symposium on Circuits and Systems - ISCAS '94*. IEEE, 1994.

[BH05] Michal Benko and Wolfgang Härdle. Common Functional Implied Volatility Analysis. In *Statistical Tools for Finance and Insurance*. Springer-Verlag, Berlin/Heidelberg, 2005.

[BHK09] Michal Benko, Wolfgang Härdle, and Alois Kneip. Common functional principal components. *The Annals of Statistics*, 37(1), 2 2009.

[BKA+19] Patric Bonnier, Patrick Kidger, Imanol Perez Arribas, Cristopher Salvi, and Terry Lyons. Deep Signature Transforms. 5 2019.

[BTLLW21] Sam Bond-Taylor, Adam Leach, Yang Long, and Chris G. Willcocks. Deep Generative Modelling: A Comparative Review of VAEs, GANs, Normalizing Flows, Energy-Based and Autoregressive Models. 3 2021.

[BY95] Fred B Bryant and Paul R Yarnold. Principal-components analysis and exploratory and confirmatory factor analysis. 1995.

[Che77] Kuo-Tsai Chen. Iterated path integrals. *Bulletin of the American Mathematical Society*, 83(5), 9 1977.

[CK16] Ilya Chevyrev and Andrey Kormilitzin. A Primer on the Signature Method in Machine Learning. 3 2016.

[CL19] Jiawei Chang and Terry Lyons. Insertion algorithm for inverting the signature of a path. 7 2019.

[CT65] James W. Cooley and John W. Tukey. An algorithm for the machine calculation of complex Fourier series. *Mathematics of Computation*, 19(90), 5 1965.

[CV95] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3), 9 1995.

[DH10] Aurore Delaigle and Peter Hall. Defining probability density for a distribution of random functions. 2 2010.

[DHL19] Erik Daxberger and José Miguel Hernández-Lobato. Bayesian Variational Autoencoders for Unsupervised Out-of-Distribution Detection. 12 2019.

[DKOZ05] Włodzisław Duch, Janusz Kacprzyk, Erkki Oja, and Sławomir Zadrożny, editors. *Artificial Neural Networks: Formal Models and Their Applications – ICANN 2005*, volume 3697. Springer Berlin Heidelberg, Berlin, Heidelberg, 2005.

[Dun21] Andrew Duncan. Functional Data Analysis Notes. Technical report, 2021.

[FADK+18] Maayan Frid-Adar, Idit Diamant, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. GAN-based Synthetic Medical Image Augmentation for increased CNN Performance in Liver Lesion Classification. 3 2018.

[GBC16] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. The MIT Press, 2016.

[GDCS19] Suchin Gururangan, Tam Dang, Dallas Card, and Noah A. Smith. Variational Pretraining for Semi-supervised Text Classification. 6 2019.

[Gen17] Xi Geng. Reconstruction for the signature of a rough path. *Proceedings of the London Mathematical Society*, 114(3):495–526, 3 2017.

[GGML15] Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. MADE: Masked Autoencoder for Distribution Estimation. 2 2015.

[GPAM+14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. 6 2014.

[Gus16] William H Guss. Deep Function Machines: Generalized Neural Networks for Topological Layer Expression. 12 2016.

[HBWP12] Matt Hoffman, David M. Blei, Chong Wang, and John Paisley. Stochastic Variational Inference. 6 2012.

[HL05] Ben Hambly and Terry Lyons. Uniqueness for the signature of a path of bounded variation and the reduced path group. 7 2005.

[HMW06] Peter Hall, Hans-Georg Müller, and Jane-Ling Wang. Properties of principal component methods for functional and longitudinal data analysis. *The Annals of Statistics*, 34(3), 6 2006.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8), 11 1997.

[HSWH21] Tsung-Yu Hsieh, Yiwei Sun, Suhang Wang, and Vasant Honavar. Functional Autoencoders for Functional Data Representation Learning. *SIAM*, 2021.

[HZRS15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. 12 2015.

[IAF] IAF VAE. https://github.com/bjlkeng/sandbox/blob/master/notebooks/vae-inverse_autoregressive_flows.

[IS15] Sergey Ioffe and Christian Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. 2 2015.

[JEP+21] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, Alex Bridgland, Clemens Meyer, Simon A.A. Kohl, Andrew J. Ballard, Andrew Cowie, Bernardino Romera-Paredes, Stanislav Nikolov, Rishub Jain, Jonas Adler, Trevor Back, Stig Petersen, David Reiman, Ellen Clancy, Michal Zielinski, Martin Steinegger, Michalina Pacholska, Tamas Berghammer, Sebastian Bodenstein, David Silver, Oriol Vinyals, Andrew W. Senior, Koray Kavukcuoglu, Pushmeet Kohli, and Demis Hassabis. Highly accurate protein structure prediction with AlphaFold. *Nature*, 2021.

[JGJS99] Michael I Jordan, Zoubin Ghahramani, Tommi S Jaakkola, and Lawrence K Saul. An Introduction to Variational Methods for Graphical Models. *Machine Learning*, 37(2):183–233, 1999.

[KB14] Diederik P. Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. 12 2014.

[KL20] Patrick Kidger and Terry Lyons. Signatory: differentiable computations of the signature and logsignature transforms, on both CPU and GPU. 1 2020.

[KLA19] Jasdeep Kalsi, Terry Lyons, and Imanol Perez Arribas. Optimal execution with rough path signatures. 5 2019.

[KPB19] Ivan Kobyzev, Simon J. D. Prince, and Marcus A. Brubaker. Normalizing Flows: An Introduction and Review of Current Methods. 8 2019.

[KR18] Piotr Kokoszka and Peter Reimherr. *Introduction to Functional Data Analysis*. 2018.

[KSJ+16] Diederik P. Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improving Variational Inference with Inverse Autoregressive Flow. 6 2016.

[KTS+14] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Fei Fei Li. Large-scale video classification with convolutional neural networks. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2014.

[KW13] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. 12 2013.

[KW19] Diederik P. Kingma and Max Welling. An Introduction to Variational Autoencoders. *Foundations and Trends® in Machine Learning*, 12(4), 2019.

[LCF15] Jong Soo Lee, Dennis D Cox, and Michele Follen. A Two Sample Test for Functional Data. *Communications for Statistical Applications and Methods*, 22(2):121–135, 3 2015.

[LCH06] Yann Lecun, Sumit Chopra, and Raia Hadsell. A tutorial on energy-based learning. 8 2006.

[LNA19] Terry Lyons, Sina Nejad, and Imanol Perez Arribas. Numerical method for model-free pricing of exotic derivatives using rough path signatures. 5 2019.

[LTH+16] Christian Ledig, Lucas Theis, Ferenc Huszar, Jose Caballero, Andrew Cunningham, Alejandro Acosta, Andrew Aitken, Alykhan Tejani, Johannes Totz, Zehan Wang, and Wenzhe Shi. Photo-Realistic Single Image Super-Resolution Using a Generative Adversarial Network. 9 2016.

[LX18] Terry Lyons and Weijun Xu. Inverting the signature of a path. *Journal of the European Mathematical Society*, 20(7), 5 2018.

[LZJ17] Chenyang Li, Xin Zhang, and Lianwen Jin. LPSNet: A Novel Log Path Signature Feature Based Hand Gesture Recognition Framework. In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*. IEEE, 10 2017.

[MFB20] Swapnil Mishra, Seth Flaxman, and Samir Bhatt. piVAE: Encoding stochastic process priors with variational autoencoders, 2 2020.

[MFSS16] Krikamol Muandet, Kenji Fukumizu, Bharath Sriperumbudur, and Bernhard Schölkopf. Kernel Mean Embedding of Distributions: A Review and Beyond. 5 2016.

[MNE+98] L. Munck, L. Nørgaard, S.B. Engelsen, R. Bro, and C.A. Andersson. Chemometrics in food science—a demonstration of the feasibility of a highly exploratory, inductive evaluation strategy of fundamental scientific significance. *Chemometrics and Intelligent Laboratory Systems*, 44(1-2), 12 1998.

[Mül97] Alfred Müller. Integral Probability Metrics and Their Generating Classes of Functions. *Advances in Applied Probability*, 29(2), 6 1997.

[ODZ+16] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. WaveNet: A Generative Model for Raw Audio. 9 2016.

[Pat09] Ram Shankar. Pathak. *THE WAVELET TRANSFORM*. Atlantis Studies in Mathematics for Engineering and Science, 4. Atlantis Press, Paris, 1st ed. 2009. edition, 2009.

[Pav18] Grigorios A. Pavliotis. *Stochastic Processes and Applications SPAS2017, Västerås and Stockholm, Sweden, October 4-6, 2017* . Springer Proceedings in Mathematics & Statistics, 271. Springer International Publishing, Cham, 1st ed. 2018. edition, 2018.

[PBJ12] John Paisley, David Blei, and Michael Jordan. Variational Bayesian Inference with Stochastic Search. 6 2012.

[PNR+19] George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing Flows for Probabilistic Modeling and Inference. 12 2019.

[PPM17] George Papamakarios, Theo Pavlakou, and Iain Murray. Masked Autoregressive Flow for Density Estimation. 5 2017.

[RCG05] Fabrice Rossi and Brieuc Conan-Guez. Functional multi-layer perceptron: a non-linear tool for functional data analysis. *Neural Networks*, 18(1), 1 2005.

[RD91] J O Ramsay and C J Dalzell. Some Tools for Functional Data Analysis. *Journal of the Royal Statistical Society. Series B, Methodological*, 53(3):539–572, 1 1991.

[Rei17] Jeremy Reizenstein. Calculation of Iterated-Integral Signatures and Log Signatures. 12 2017.

[RM15] Danilo Jimenez Rezende and Shakir Mohamed. Variational Inference with Normalizing Flows. 5 2015.

[RMW14] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. 1 2014.

[RR08] Ali Rahimi and Benjamin Recht. Random Features for Large-Scale Kernel Machines. In J Platt, D Koller, Y Singer, and S Roweis, editors, *Advances in Neural Information Processing Systems*, volume 20. Curran Associates, Inc., 2008.

[RR21] Aniruddha Rajendra Rao and Matthew Reimherr. Modern Non-Linear Function-on-Function Regression. 7 2021.

[RS02] J.O. Ramsay and B.W. Silverman. *Applied Functional Data Analysis: Methods and Case Studies*. Springer New York, New York, NY, 2002.

[RS05] J. O. Ramsay and B. W. Silverman. *Functional Data Analysis*. Springer New York, New York, NY, 2005.

[SHK+14] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, 1 2014.

[SK21] Yang Song and Diederik P. Kingma. How to Train Your Energy-Based Models. 1 2021.

[SLY15] Kihyuk Sohn, Honglak Lee, and Xinchen Yan. Learning Structured Output Representation using Deep Conditional Generative Models. In C Cortes, N Lawrence, D Lee, M Sugiyama, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[SP97] M. Schuster and K.K. Paliwal. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11), 1997.

[Sug] Sugar Spectra data set. http://jeffgoldsmith.com/IWAFDA/shortcourse_sofr.html.

[VdMH08] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-SNE. *Journal of machine learning research*, 9, 2008.

[VGS05] Roberto Viviani, Georg Grön, and Manfred Spitzer. Functional principal component analysis of fMRI data. *Human Brain Mapping*, 24(2), 2 2005.

[WD20] George Wynne and Andrew B. Duncan. A Kernel Two-Sample Test for Functional Data. 8 2020.

[YJvdS19] Jinsung Yoon, Daniel Jarrett, and Mihaela van der Schaar. Time-series Generative Adversarial Networks. In H Wallach, H Larochelle, A Beygelzimer, F d Alché-Buc, E Fox, and R Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019.

[YMW05] Fang Yao, Hans-Georg Müller, and Jane-Ling Wang. Functional Data Analysis for Sparse Longitudinal Data. *Journal of the American Statistical Association*, 100(470), 6 2005.

[ZSE17] Shengjia Zhao, Jiaming Song, and Stefano Ermon. Towards Deeper Understanding of Variational Autoencoding Models. 2 2017.

[ZW16] Xiaoke Zhang and Jane-Ling Wang. FROM SPARSE TO DENSE FUNCTIONAL DATA AND BEYOND. *The Annals of statistics*, 44(5):2281–2321, 10 2016.

[ZXL+16] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, Xiaogang Wang, Xiaolei Huang, and Dimitris Metaxas. StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. 12 2016.

Appendix A

Additional theory

Theorem A.1. *A finite dimensional analog to Mercer's Theorem. Consider the gram matrix* $G_{ij} = k(x_i, x_j)$ *where* k *is a continuous symmetric non-negative definite kernel. Let* $f = (f_1, \ldots, f_n) \in \mathbb{R}^n$. *Then*

$$(T_k f)(\cdot) = \sum_{n=1}^{n} k(\cdot, x_i) f_i.$$

It follows that G *is non-negative definite and hence can be written as*

$$G = \sum_{n=1}^{n} \lambda_i v_i v_i^T,$$

with $\lambda_i \geq 0$. *It follows that*

$$k(x_i, x_j) = K_{ij} \tag{A.1}$$

$$= \sum_{t=1}^{n} \lambda_t v_{ti} v_{tj} \tag{A.2}$$

$$= \sum_{t=1}^{n} \lambda_t \psi_t(x_i) \psi_t(x_j) \tag{A.3}$$

where $\psi_t : \mathcal{X} \to \mathbb{R}$ *is given by* $\psi_t(x_i) = v_{t,i}$.

Appendix B

Additional results

B.1 Choice of Basis

B.1.1 Simulated Data

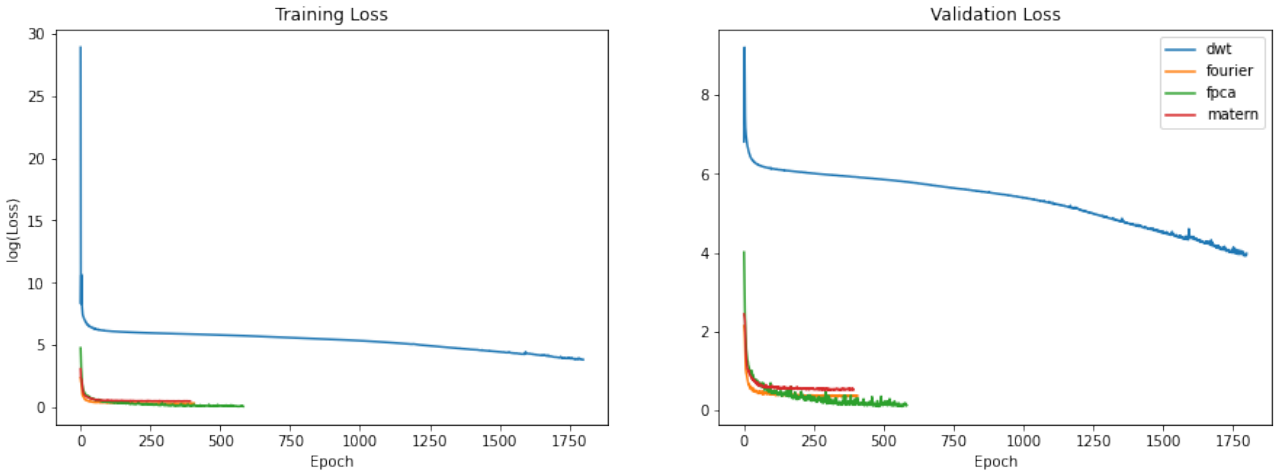

Figure B.1: Train and validation loss for different basis on the simulated Gaussian process prior samples.

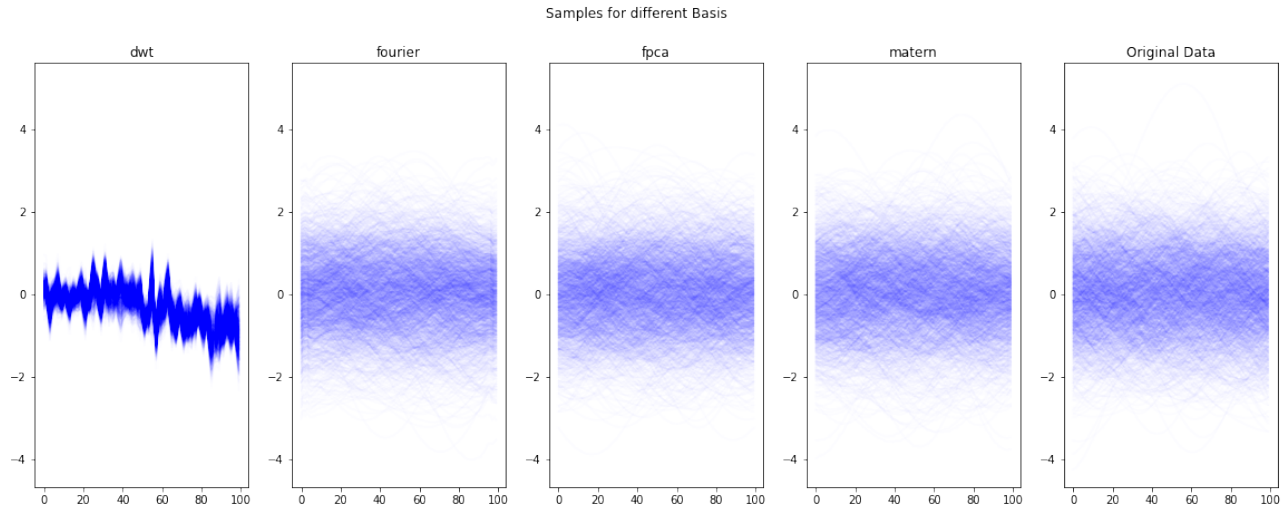

Figure B.2: Synthetic samples for different bases in comparison to the simulated Gaussian process test set.

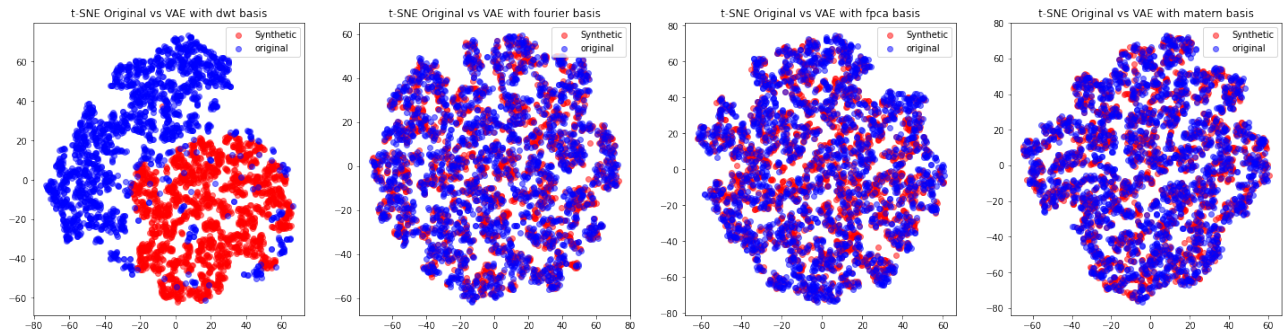

Figure B.3: Synthetic samples and true test set projected to 2-D via t-SNE for a visual comparison.

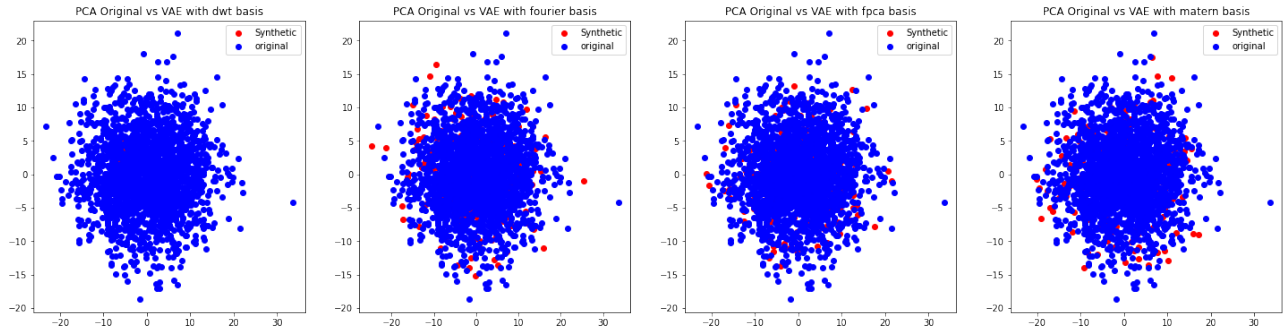

Figure B.4: Synthetic samples and true test set mapped to 2-D via FPCA for a visual comparison.

B.1.2 Gridwatch Data

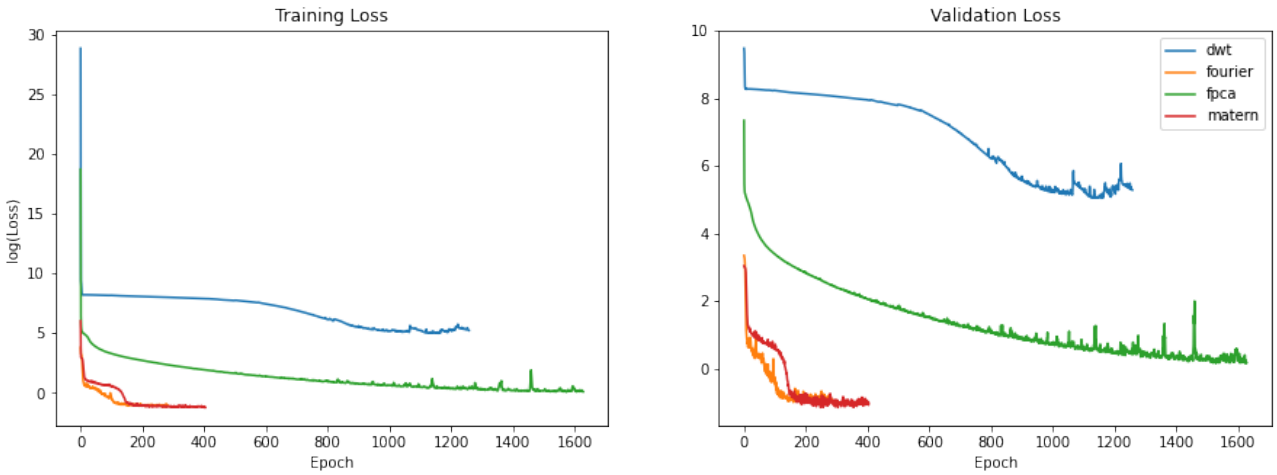

Figure B.5: Train and validation loss for different basis on the Gridwatch data.

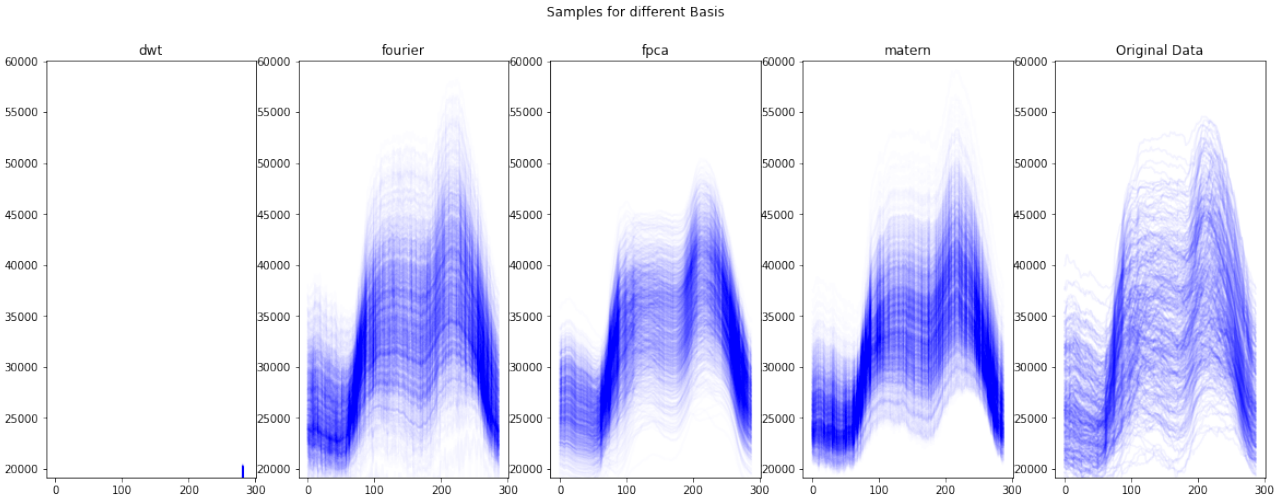

Figure B.6: Synthetic samples for different bases in comparison to the original Gridwatch data.

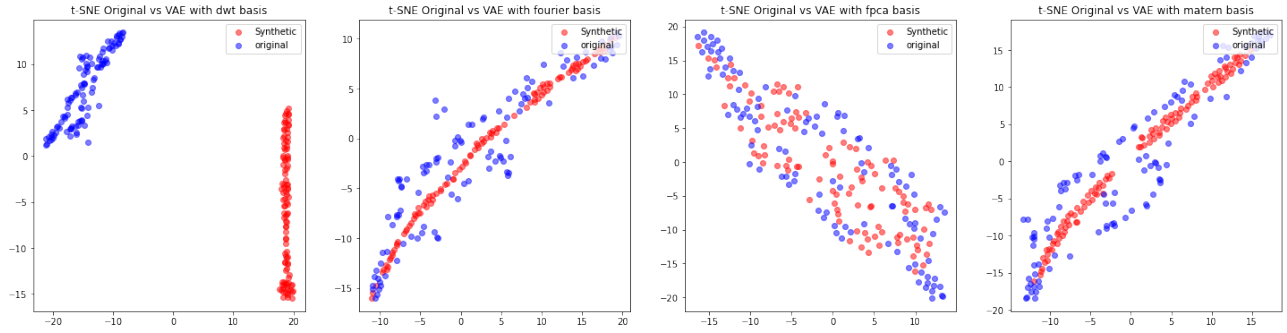

Figure B.7: Synthetic samples and true test set projected to 2-D via t-SNE for a visual comparison.

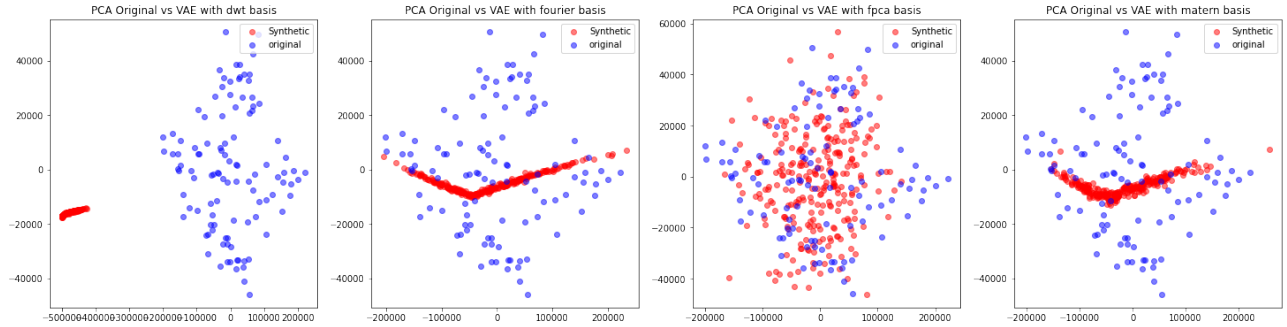

Figure B.8: Synthetic samples and true test set mapped to 2-D via FPCA for a visual comparison

B.1.3 Sugar spectra Data

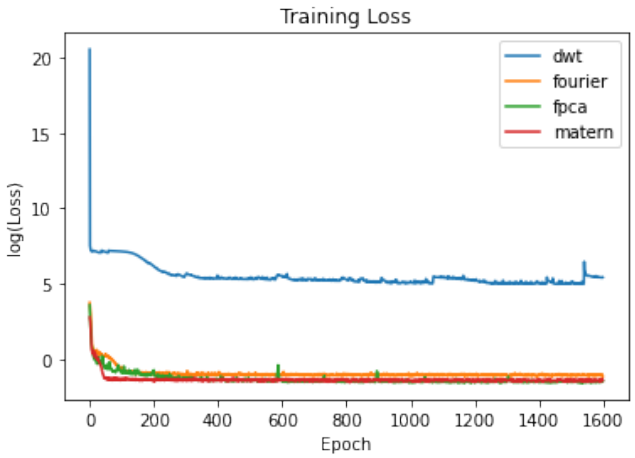

Figure B.9: Train loss for different basis on the sugar spectra data.

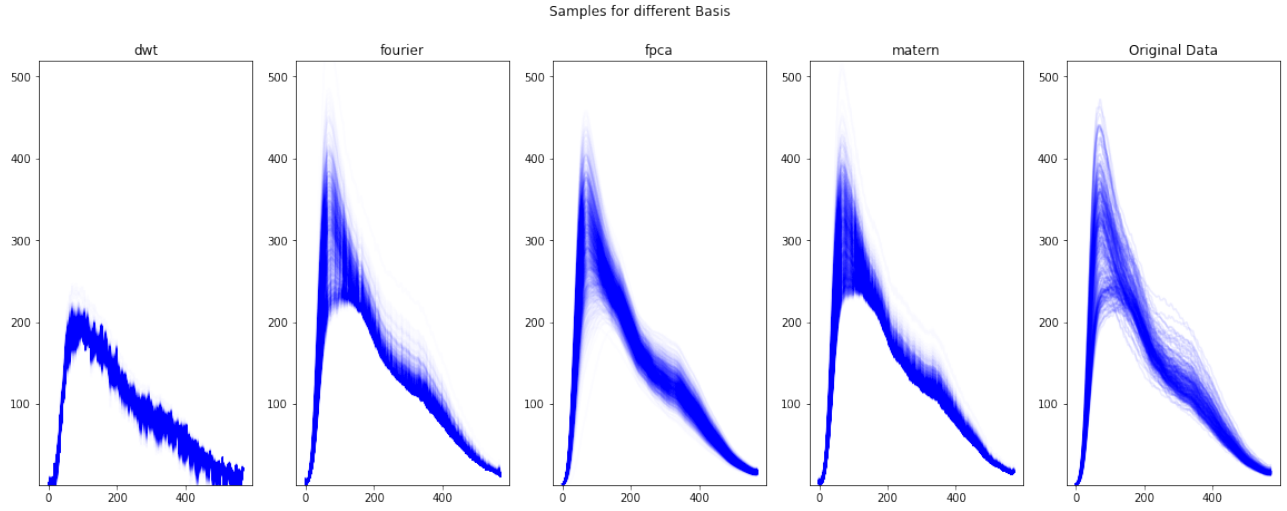

Figure B.10: Synthetic samples for different bases in comparison to the original sugar spectra data.

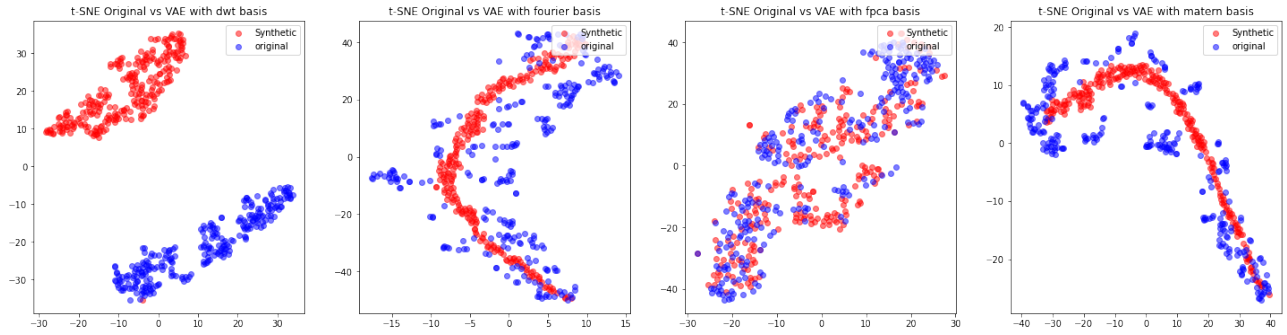

Figure B.11: Synthetic samples and true test set projected to 2-D via t-SNE for a visual comparison.

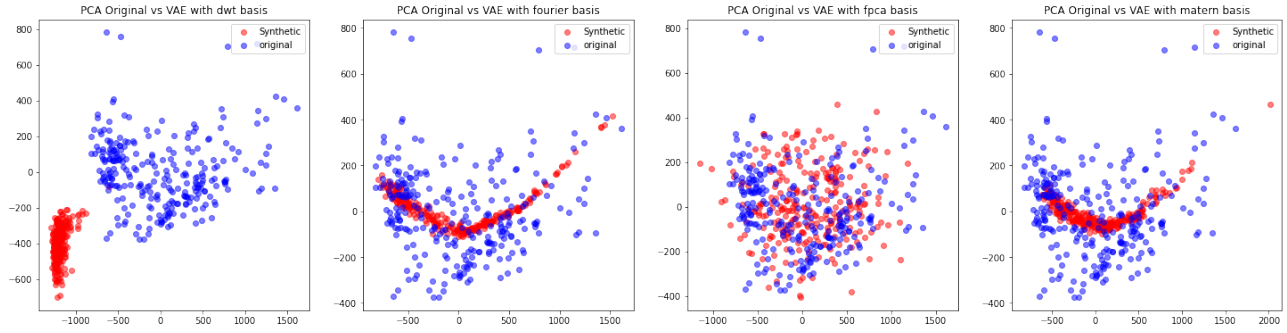

Figure B.12: Synthetic samples and true test set projected to 2-D via FPCA for a visual comparison.

B.2 Signature

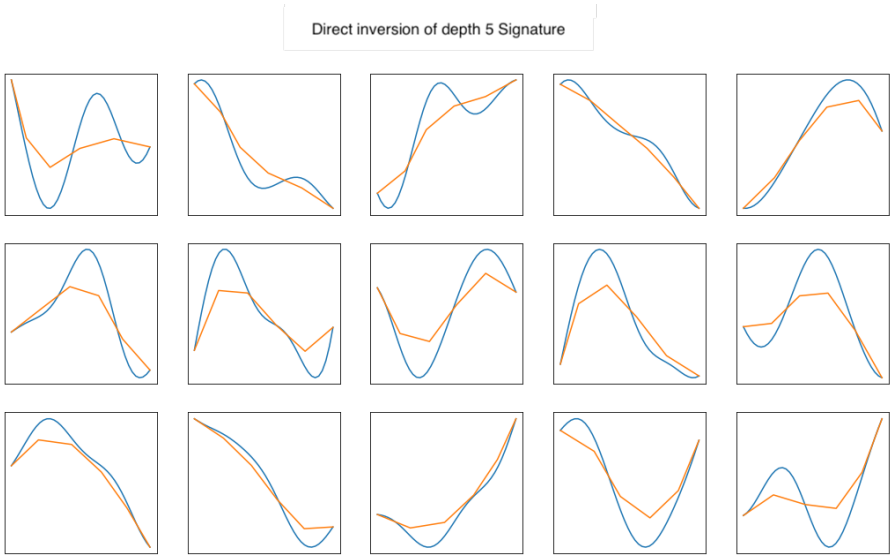

Figure B.13: The paths are samples from the Gaussian process prior (blue). The signature transform of depth 5 is taken and directly inverted (orange).

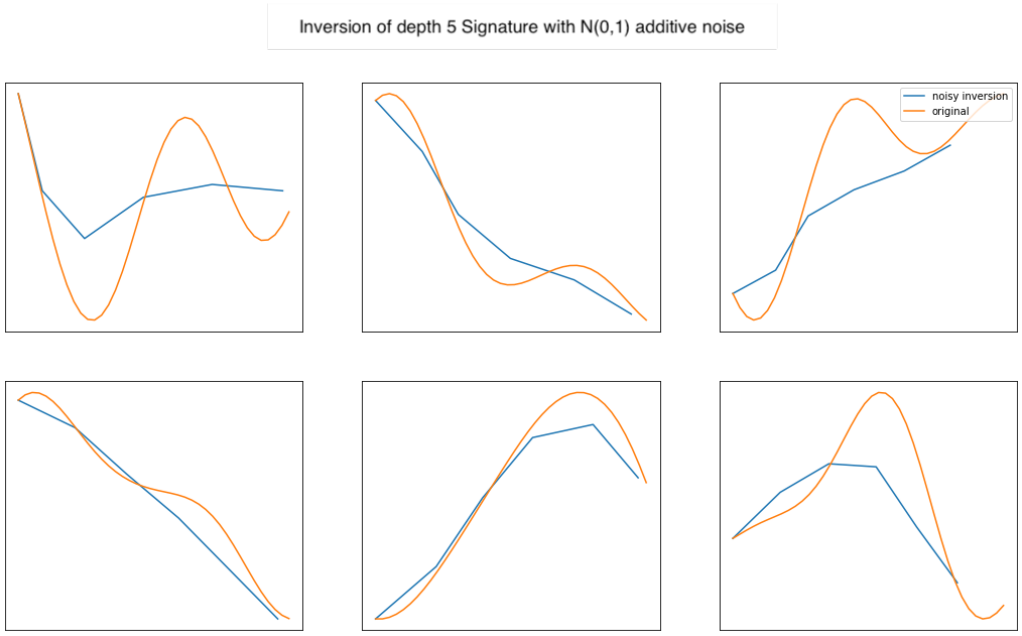

Figure B.14: The paths are samples from the Gaussian process prior. The signature transform of depth 5 is taken, standard normal noise is added and then the inverse transform is performed. The path approximations from the noisy signature are compared to the original paths. Notice that the inversion is reasonably robust to the additive noise. More experiments were performed but omitted here.

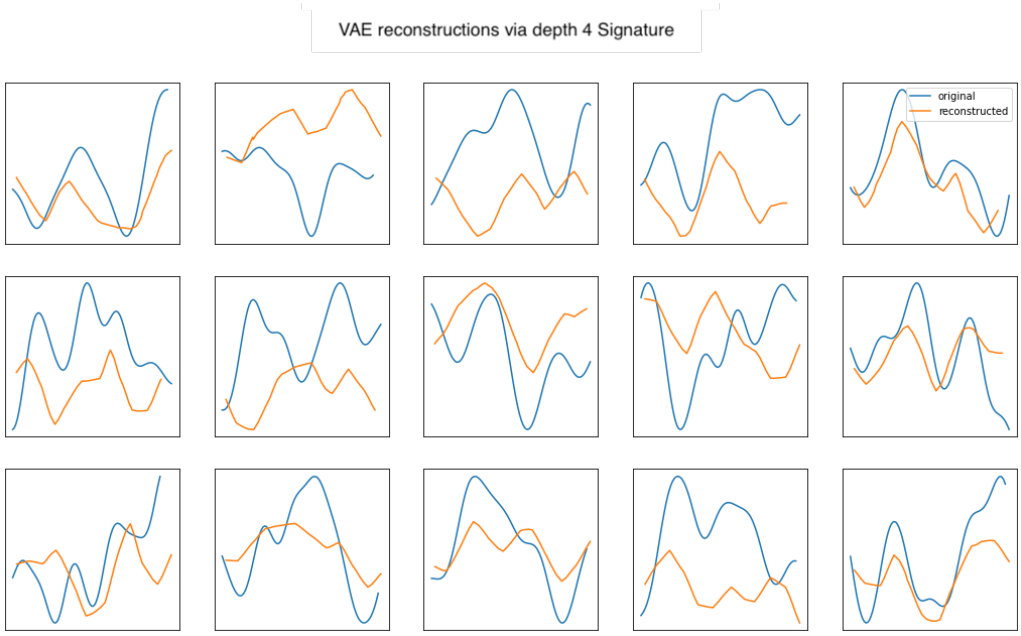

Figure B.15: Using the signature transform of depth 4 on path segments of length 10 and concatenated. The reconstructions obtained by a full pass through the model and the corresponding inputs are visualised

B.3 Bidirectional-LSTM classifier example

```
Model:
_________________________________________________________________
Layer␣(type)␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Output␣Shape␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣Param␣#
=================================================================
bidirectional_14␣(BiLSTM)␣␣␣␣␣(None,␣40)␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣3520
_________________________________________________________________
dropout_14␣(Dropout)␣␣␣␣␣␣␣␣␣(None,␣40)␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣0
_________________________________________________________________
dense_229␣(Dense)␣␣␣␣␣␣␣␣␣␣␣␣(None,␣2)␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣82
=================================================================
Total␣params:␣3,602
Trainable␣params:␣3,602
```

```
Non-trainable params: 0
```

B.4 Sensitivity Analysis

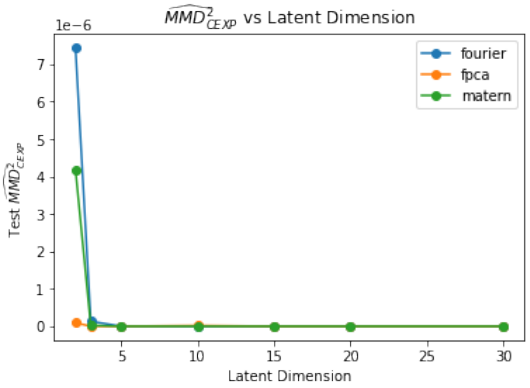

Figure B.16: The dimensionality of the latent variable is varied from 2 to 30 and for each value the VAE is trained for 500 epochs with each basis. The estimated MMD based on CEXP between the original Gridwatch data and the synthetic samples is reported.

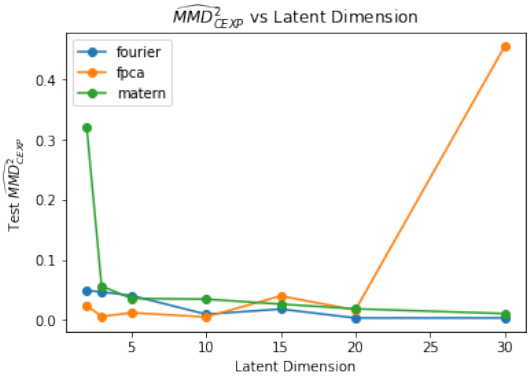

Figure B.17: The dimensionality of the latent variable is varied from 2 to 30 and for each value the VAE is trained for 500 epochs with each basis. The estimated MMD based on CEXP between the original sugar spectra data and the synthetic samples is reported.

B.5 Comparison between VAE with and without IAF

The following as an example architecture for VAE with 4 IAF steps. We note that the number of trainable parameters is 61,055.

```
    Model: Example of VAE with IAF
__________________________________________________________________________
Layer (type)                 Output Shape              Param #    Connected to
==========================================================================
input_6 (InputLayer)         [(100, 100)]              0
```

```
____________________________________________________________________________
enc_dense_1 (Dense)             (100, 100)           10100   input_6[0][0]
____________________________________________________________________________
enc_bn_1 (BatchNormalization)   (100, 100)           400    enc_dense_1[0][0]
____________________________________________________________________________
activation_15 (Activation)      (100, 100)           0       enc_bn_1[0][0]
____________________________________________________________________________
dropout_11 (Dropout)            (100, 100)           0      activation_15[0][0]
____________________________________________________________________________
enc_dense_2 (Dense)             (100, 100)           10100  dropout_11[0][0]
____________________________________________________________________________
enc_bn_2 (BatchNormalization)   (100, 100)           400    enc_dense_2[0][0]
____________________________________________________________________________
activation_16 (Activation)      (100, 100)           0      enc_bn_2[0][0]
____________________________________________________________________________
dropout_12 (Dropout)            (100, 100)           0      activation_16[0][0]
____________________________________________________________________________
lambda_9 (Lambda)               (100, 5)             0      dropout_12[0][0]
____________________________________________________________________________
dense_3 (Dense)                 (100, 5)             505    dropout_12[0][0]
____________________________________________________________________________
dense_4 (Dense)                 (100, 5)             505    dropout_12[0][0]
____________________________________________________________________________
lambda_10 (Lambda)              (100, 5)             0      lambda_9[0][0]
                                                            dense_3[0][0]
                                                            dense_4[0][0]
____________________________________________________________________________
dense_5 (Dense)                 (100, 5)             505    dropout_12[0][0]
____________________________________________________________________________
```

```
masking_dense_9 (MaskingDense)  (100, 5)            3430   lambda_10[0][0]
                                                           dense_5[0][0]
__________________________________________________________________________
masking_dense_8 (MaskingDense)  (100, 5)            3430   lambda_10[0][0]
                                                           dense_5[0][0]
__________________________________________________________________________
activation_17 (Activation)      (100, 5)            0      masking_dense_9[0][0]
__________________________________________________________________________
lambda_11 (Lambda)              (100, 5)            0      lambda_10[0][0]
                                                           masking_dense_8[0][0]
                                                           activation_17[0][0]
__________________________________________________________________________
lambda_12 (Lambda)              (100, 5)            0      lambda_11[0][0]
__________________________________________________________________________
masking_dense_11 (MaskingDense) (100, 5)            3430   lambda_12[0][0]
                                                           dense_5[0][0]
__________________________________________________________________________
masking_dense_10 (MaskingDense) (100, 5)            3430   lambda_12[0][0]
                                                           dense_5[0][0]
__________________________________________________________________________
activation_18 (Activation)      (100, 5)            0      masking_dense_11[0][0]
__________________________________________________________________________
lambda_13 (Lambda)              (100, 5)            0      lambda_12[0][0]
                                                           masking_dense_10[0][0]
                                                           activation_18[0][0]
__________________________________________________________________________
lambda_14 (Lambda)              (100, 5)            0      lambda_13[0][0]
__________________________________________________________________________
masking_dense_13 (MaskingDense) (100, 5)            3430   lambda_14[0][0]
```

```
                                                          dense_5[0][0]
__________________________________________________________________________________
masking_dense_12 (MaskingDense) (100, 5)            3430  lambda_14[0][0]
                                                          dense_5[0][0]
__________________________________________________________________________________
activation_19 (Activation)      (100, 5)            0     masking_dense_13[0][0]
__________________________________________________________________________________
lambda_15 (Lambda)              (100, 5)            0     lambda_14[0][0]
                                                          masking_dense_12[0][0]
                                                          activation_19[0][0]
__________________________________________________________________________________
lambda_16 (Lambda)              (100, 5)            0     lambda_15[0][0]
__________________________________________________________________________________
masking_dense_15 (MaskingDense) (100, 5)            3430  lambda_16[0][0]
                                                          dense_5[0][0]
__________________________________________________________________________________
masking_dense_14 (MaskingDense) (100, 5)            3430  lambda_16[0][0]
                                                          dense_5[0][0]
__________________________________________________________________________________
activation_20 (Activation)      (100, 5)            0     masking_dense_15[0][0]
__________________________________________________________________________________
lambda_17 (Lambda)              (100, 5)            0     lambda_16[0][0]
                                                          masking_dense_14[0][0]
                                                          activation_20[0][0]
__________________________________________________________________________________
enc_dense_7 (Dense)             (100, 100)          600   lambda_17[0][0]
__________________________________________________________________________________
enc_bn_7 (BatchNormalization)   (100, 100)          400   enc_dense_7[0][0]
__________________________________________________________________________________
```

```
activation_21 (Activation)      (100, 100)           0       enc_bn_7[0][0]
__________________________________________________________________________
dropout_13 (Dropout)            (100, 100)           0       activation_21[0][0]
__________________________________________________________________________
x_decoded (Dense)               (100, 100)           10100   dropout_13[0][0]
==========================================================================
Total params: 61,055
Trainable params: 60,455
Non-trainable params: 600
```

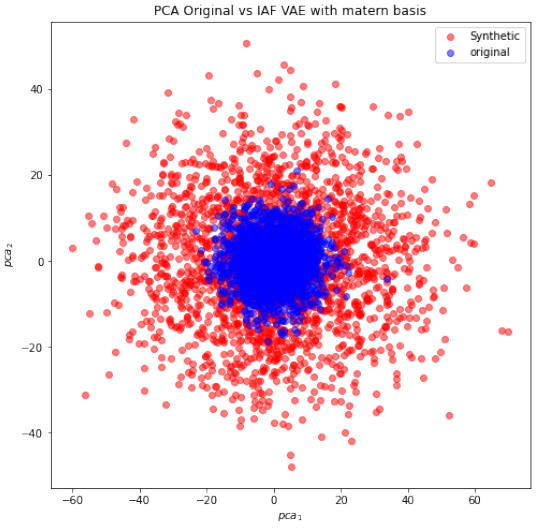

Figure B.18: Synthetic samples from VAE with IAF with Matérn kernel based basis and test data set for simulated GP samples projected into 2-D via FPCA.

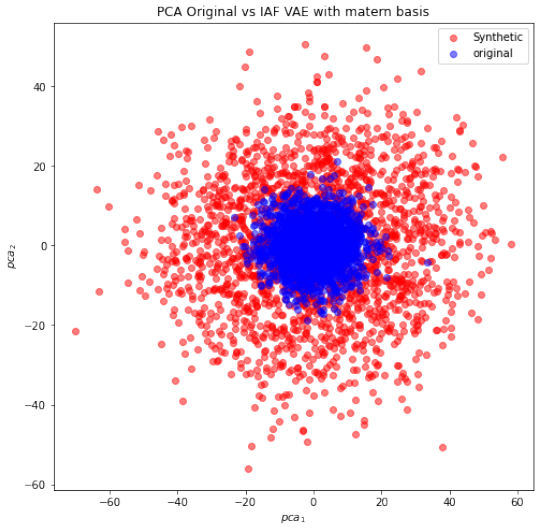

Figure B.19: Synthetic samples from VAE with IAF with FPCA based basis and test data set for simulated GP samples projected into 2-D via FPCA.

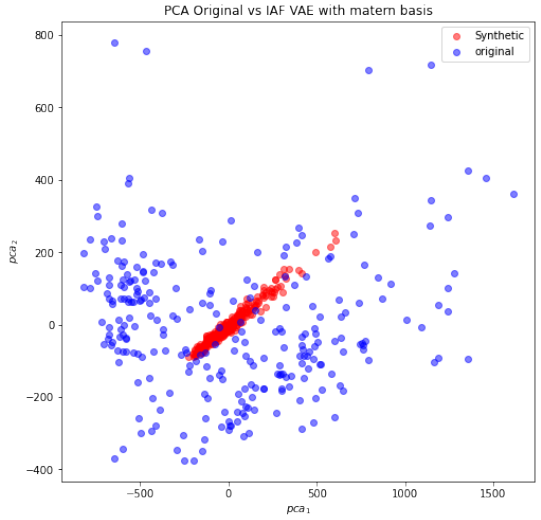

Figure B.20: Synthetic samples from VAE with IAF with Matérn kernel based basis and sugar spectra data set projected into 2-D via FPCA.

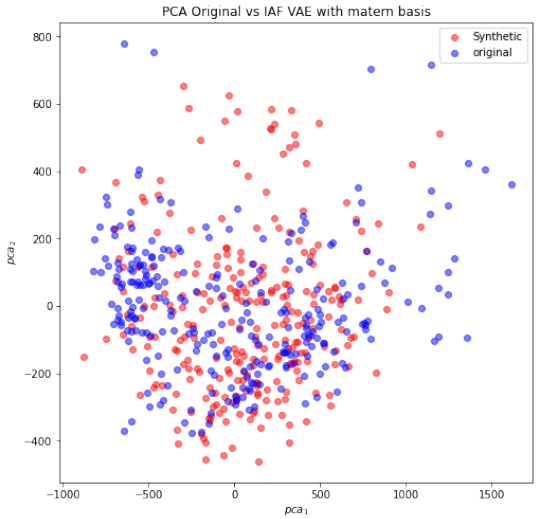

Figure B.21: Synthetic samples from VAE with IAF with FPCA based basis and sugar spectra data set projected into 2-D via FPCA.

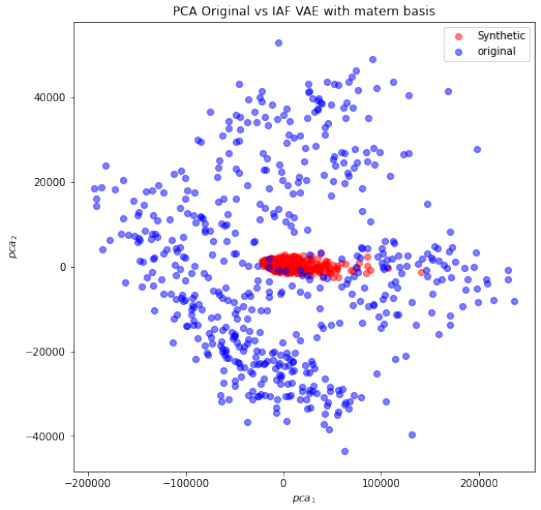

Figure B.22: Synthetic samples from VAE with IAF with Matérn kernel based basis and Gridwatch data set projected into 2-D via FPCA.

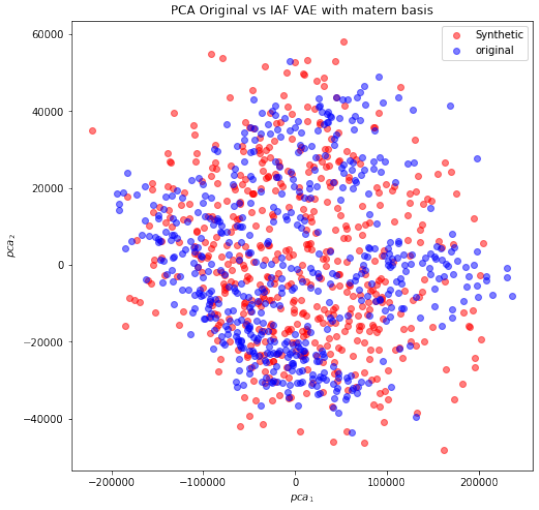

Figure B.23: Synthetic samples from VAE with IAF with FPCA based basis and Gridwatch data set projected into 2-D via FPCA.