FACULTY OF HEALTH AND MEDICAL SCIENCES
UNIVERSITY OF COPENHAGEN

Statistical methods for analyzing ECG signals using deep learning and functional data analysis

Vi Thanh Pham

This thesis has been submitted to the Graduate School of Health and Medical Sciences, University of Copenhagen on November 30, 2024.

Vi Thanh Pham

Statistical methods for analyzing ECG signals using deep learning and functional data analysis

ACADEMIC ADVISORS

Andreas Kryger Jensen
Section of Biostatistics, University of Copenhagen, Denmark

Thomas Alexander Gerds
Section of Biostatistics, University of Copenhagen, Denmark

ASSESSMENT COMMITTEE

Esben Budtz-Jørgensen
Section of Biostatistics, University of Copenhagen, Denmark

Benoit Liquet-Weiland
School of Mathematical and Physical Sciences, Macquarie University, Australia

Morten Mørup
Section for Cognitive Systems, Technical University of Denmark, Denmark

Acknowledgements

The research presented in this thesis was conducted at the Section of Biostatistics, University of Copenhagen, under the guidance of Andreas Kryger Jensen and Thomas Alexander Gerds, with a short time visiting Marvin N. Wright at the Leibniz Institute for Prevention Research and Epidemiology – BIPS in Bremen, Germany.

I want to thank everyone at the Section of Biostatistics for their support, without which this work would not have been possible. Thanks to my supervisors, Andreas, for encouraging me to trust the process, and Thomas, for reminding me to trust myself. Thanks to Sanne for ensuring we never missed a cake in the kitchen and for *te og godt selskab*. Thanks to Zehao for being an exceptional officemate and, more importantly, an invaluable friend.

Lastly, a very special thanks to my husband, Jeppe, with whom I can discuss anything and everything. You have celebrated all the ups and supported me through all the downs of my PhD journey. Thank you for all the adventures we have experienced, and I look forward to many more.

Summary

This PhD thesis develops statistical methods that integrate deep neural networks in statistical models to predict various health outcomes, with a focus on electrocardiogram (ECG) data as predictors. Traditional ECG signal analysis relies mostly on fixed algorithms, which can be affected by patient motion during observation, leading to inaccurate readings, but also being specifically tailored to predetermined outcomes through non-data-driven feature extraction methods.

This research aims to eliminate such artifacts and bypass the pre-specified feature selection step entirely by associating full ECG signals directly to clinical outcomes of interest using methods from functional data analysis. Additionally, the project seeks to evaluate ECG measurements to predict short-term and long-term risks of various cardiovascular diseases, mortality, and other conditions by adapting existing methodologies and developing new algorithms.

The analysis of ECG data presents significant challenges due to inherent noise, high dimensionality, and changes over time. The primary objectives of this PhD thesis include a method for alignment of multivariate functional data in time and the development of neural networks capable of processing functional inputs and time-to-event outcomes while adjusting for the time warping. Proper alignment of ECG data in time ensures accurate diagnosis, comparability, noise reduction, and data integration. Combining functional data analysis and neural networks into a hybrid model offers robust models that are less prone to overfitting. Furthermore, the application of survival analysis to ECG data allows for the prediction and understanding of critical cardiac events, enabling timely interventions at the patient level.

Resumé

Denne ph.d.-afhandling fokuserer på udviklingen af statistiske metoder, der integrerer dybe, neurale netværk i statistiske modeller, der kan bruges til at forudsige forskellige helbredsudfald, med fokus på elektrokardiogram (EKG) data som prædiktorer. Traditionel analyse af EKG-signaler er for det meste baseret på faste algoritmer, som kan påvirkes af patientbevægelser under observation, hvilket fører til unøjagtige aflæsninger, men de er også specifikt skræddersyet til forudbestemte resultater gennem ikke-datadrevne ekstraktionsmetoder.

Forskningen i denne afhandling sigter mod at eliminere sådanne artefakter og omgå det forudspecificerede ekstraktionstrin ved at knytte fulde EKG-signaler direkte til kliniske udfald af interesse ved hjælp af metoder fra funktionel dataanalyse. Derudover søger projektet at evaluere EKG-målinger til at forudsige kort- og langsigtede risici for forskellige kardiovaskulære sygdomme, dødelighed og andre tilstande ved at tilpasse eksisterende metoder og udvikle nye algoritmer.

Analyse af EKG-data har betydelige udfordringer på grund af iboende støj, høj dimensionalitet og ændringer over tid. De primære mål for denne ph.d.-afhandling omfatter en metode til opretning af multivariate funktionelle data i tid og udvikling af neurale netværk, der er i stand til at behandle funktionelle data som input og tid-til-hændelsesudfald, mens der justeres for tidsforskydninger. Korrekt justering for tidsforskydninger af EKG-data sikrer nøjagtig diagnose, sammenlignelighed, støjreduktion og dataintegration. Kombinationen af funktionel dataanalyse og neurale netværk i en hybrid model giver robuste modeller, der er mindre tilbøjelige til at overtilpasse. Desuden giver anvendelsen af overlevelsesanalyse til EKG-data mulighed for forudsigelse og forståelse af kritiske hjerterelaterede hændelser, hvilket muliggør rettidige indgreb på patientniveau.

Contents

Synopsis

1. Introduction

The primary objective of this research project is to develop advanced neural networks capable of providing clinically relevant predictions of various health outcomes based on electrocardiogram (ECG, Chen, 2018) data. The ultimate aim is to create a deep learning algorithm to deliver predicted risk probabilities for different health outcomes based on ECG measurements.

The study population consists of participants from the Copenhagen General Population Study (CGPS),[1] a large-scale ongoing study. It aims to investigate prevalent health conditions within the Danish population and establish correlations among lifestyle factors, genetics, and various diseases. Participants, aged between 20 and 100 years, are randomly selected from the Danish Civil Registration System.[2] In 2010, the Danish National Committee on Health Research Ethics authorized the inclusion of cardiac CT scans[3] for participants aged 40 and above in the CGPS[4]. Cardiac CT scans were conducted at Rigshospitalet, with over 10,000 participants from the study undergoing these examinations. Over the years, cardiac CT imaging has proven instrumental in identifying novel cardiovascular biomarkers. CT scans can help identify and diagnose conditions such as bone features, internal injuries, tumors, infections, and blood clots, and are often used to guide further medical procedures, such as biopsies, surgeries, and radiation therapy.

ECGs offer several advantages over CT scans, particularly in cardiac diagnostics. ECGs are quick and provide immediate results, making them ideal for initial assessments and routine check-ups. They are also more cost-effective and do not expose patients to ionizing radiation, which benefits those requiring frequent monitoring. Additionally, ECG machines are portable and can be used in various settings, including emergency rooms and ambulances. However, ECGs cannot show the heart's structure or the condition of coronary arteries (Goldberger et al., 2018, chap. 24). Given the widespread availability of ECGs, beyond their use in detecting arrhythmias, heart attacks, and heart damage, we aim to demonstrate their potential in screening for anatomical heart diseases.

Currently, ECG signal interpretation relies on a predetermined set of algorithms that assess heart rate, detect irregularities in heart rhythm, and identify other patterns within

[1] Copenhagen University Hospitals, *Organization page of the Copenhagen General Population Study* [website], https://research.regionh.dk/en/organisations/copenhagen-general-population-study, accessed November 29, 2024

[2] The National Centre for Register-based Research, *The Danish Civil Registration System (CPR)* [website], https://ncrr.au.dk/danish-registers/the-danish-civil-registration-system-cpr, accessed November 29, 2024

[3] National Health Service, *CT Scan* [website], https://www.nhs.uk/conditions/ct-scan/, accessed November 29, 2024

[4] Cardiovascular CT Research Unit, Rigshospitalet, *Ongoing Cardiovascular CT Projects* [website], https://rh-ct.org/projects/, accessed November 29, 2024

the signal waves, aiding in diagnosing and monitoring cardiac conditions. However, ECG signals are often affected by patient motion, which can be either rhythmic or irregular. Using these predefined algorithms, signal distortions may be misinterpreted as cardiac arrhythmias, leading to inaccurate readings and prognoses. One possible approach to this issue is to circumvent the diagnostic step altogether by directly associating ECG signals with clinical outcomes.

The project aims to assess ECG measurements and predict the short-term and long-term risks of various cardiovascular diseases, mortality, and other conditions by refining existing methodologies and designing new algorithms.

1.1. OBJECTIVES

The analysis of ECG data presents significant challenges due to inherent noise and the high dimensionality of the dataset. This section delineates the three primary objectives and the underlying motivations that will facilitate the analysis of ECG data and its association with clinical outcomes. The objectives include the alignment of multivariate functional data and the development of neural networks capable of processing functional inputs and time-to-event outcomes.

1.1.1. *Alignment of functional data*

Alignment of functional data is imperative for automated analysis, particularly when employing neural networks, due to several critical factors. Firstly, it ensures consistency and comparability, allowing data points to be accurately positioned relative to each other, which is essential for effective pattern recognition and reliable conclusions. Secondly, alignment minimizes noise and artifacts, resulting in cleaner data inputs that enhance the learning and predictive capabilities of neural networks. Additionally, well-aligned data improves the performance of neural network architectures, such as recurrent neural networks and convolutional neural networks, by enabling them to better capture underlying patterns and relationships.

Alignment of ECG data is essential for ensuring accurate diagnosis, comparability, noise reduction, and data integration. Proper alignment guarantees that the electrical activity of the heart is correctly represented, thereby minimizing the risk of misdiagnosis or inappropriate treatment. It also facilitates consistent comparisons between different recordings, which is crucial for monitoring changes in a patient's heart condition over time or comparing results across different patients. Additionally, alignment helps reduce noise and artifacts caused by external factors such as body movements or muscle activity, leading to clearer and more reliable ECG readings.

We aim to implement neural network models capable of aligning multivariate quasi-periodic functional data, such as ECG, and this objective is achieved in Manuscript I. The motivation for Manuscript I is further detailed in Section 2.

1.1.2. *Functional neural network*

ECG signals are continuous, and each heartbeat can be represented as a curve. Functional data analysis allows for the study of these curves as whole entities rather than discrete

points, capturing the data's inherent smoothness and continuity. Functional data analysis provides a more comprehensive understanding of the heart's electrical activity, which can lead to better identification of cardiac abnormalities and more robust biomarkers.

Neural networks are particularly useful for analyzing functional data for several reasons. They can adaptively learn the best representation of the functional data, which is especially useful when the data is high-dimensional and complex. Unlike traditional methods that require separate steps for feature extraction and classification, neural networks can perform these tasks simultaneously. This end-to-end learning approach can improve the efficiency and accuracy of the analysis.

We aim to combine functional data analysis and neural networks into a hybrid model that offers significant benefits of both approaches – functional data analysis ensures that the data is well-structured while neural networks can handle large datasets. This combination thus leads to robust models that are less prone to overfitting. We achieve the integration of functional data analysis into neural networks in Manuscript II. Section 4 provides the foundation for functional neural networks.

1.1.3. *Time-to-event outcome*

Applying survival analysis to ECG data is important because it allows for the prediction and understanding of critical cardiac events, such as sudden cardiac death or recurrent cardiovascular incidents. ECG signals provide continuous data that can be analyzed to identify which features are most predictive of adverse outcomes. By modeling the time until these events occur, survival analysis helps in risk stratification and patient management, enabling timely interventions. This approach is particularly useful in handling censored data, where the event of interest has not occurred for some patients during the study period.

Using neural networks for survival analysis offers several benefits compared to standard semi-parametric models, one of them being their ability to handle high-dimensional data. Furthermore, the relationship between predictors and time-to-event is often non-linear and complex. Neural networks can capture these non-linearities, which is crucial for understanding the underlying risk factors and their interactions.

We achieve the implementation of neural networks for time-to-event outcomes in Manuscript III. An extended background for Manuscript III is described in Section 3.

1.2. USE OF GENERATIVE AI

Microsoft Copilot (Microsoft, 2024) was utilized during the preparation of this thesis to assist with grammar correction and to enhance the clarity and coherence of the writing. The contributions of Microsoft Copilot were strictly limited to linguistic improvements and did not influence the substantive content or the originality of the presented research.

2. *Alignment of quasi-periodic functional data*

ECG data is often misaligned due to several factors that can significantly impact the accuracy and reliability of the readings. One of the reasons for misalignment is the incorrect placement of electrodes. Proper electrode placement is crucial for obtaining accurate ECG readings, since misplacement can lead to significant errors (Goldberger et al., 2018, chap. 25). Patient movement during the ECG recording is another major factor contributing to misalignment. Even slight movements can disrupt the signal, causing artifacts and inaccuracies in the ECG data. Physiological variability, such as breathing, heart rate variability, and other bodily movements, can cause fluctuations in the ECG signal, contributing to misalignment. These natural physiological variations are inherent in all patients and can complicate the alignment process. Figure 2.1 represents the noise in the ECG data. This figure is intended solely for illustrative purposes and does not capture the full range of variability present in ECG recordings. This data was simulated using the `NeuroKit2` Python module (Makowski et al., 2021) and will be used throughout the thesis for demonstrations.

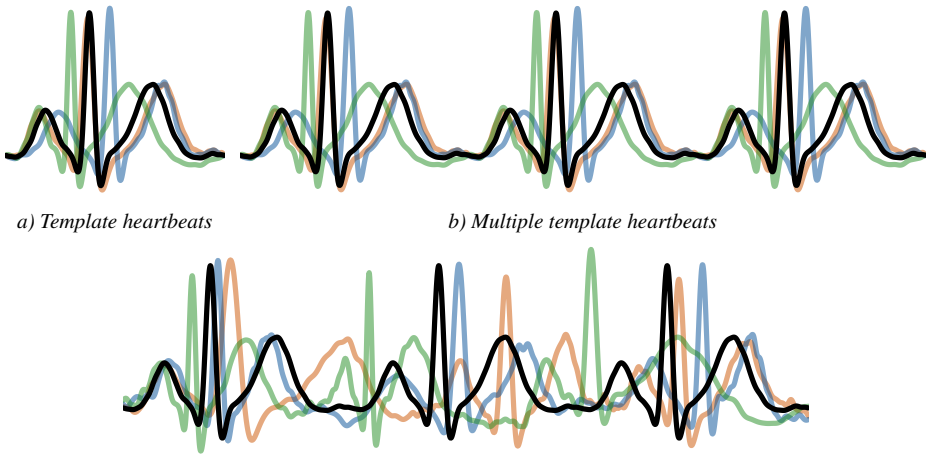


FIGURE 2.1. *Illustration of the observed misalignment in the ECG data. Figure a) shows the common (black) and subject-specific template heartbeats, while Figure b) shows multiple consecutive common (black) and subject-specific template heartbeats. Figure c) shows the observed data, which stems from Figure b) with additional noise in the time domain and amplitude.*

2.1. JOINT ALIGNMENT OF FUNCTIONS

First, we focus on the joint alignment of individual heartbeats, as represented in Figure 2.1a). In general, our objective is to identify warping functions that align the ECG biomarkers. We denote the set of N observed functions as $\{f_i\}$, which is a shorthand for $\{f_i\}_{i=1}^N = \{f_i \in \mathcal{F}_I \mid i = 1, \ldots, N\}$, where $\mathcal{F}_I$ is the set of absolutely continuous functions defined on the interval I. Without loss of generality, we consider $I = [0, 1]$. Given a functional template $\mu \in \mathcal{F}_I$, we aim to determine warping functions $\{\gamma_i\}$, a shorthand for $\{\gamma_i\}_{i=1}^N = \{\gamma_i \in \Gamma_I \mid i = 1, \ldots, N\}$, that align them to this template function μ. The set Γ_I is the set of *boundary-preserving* ($\gamma(0) = 0$ and $\gamma(1) = 1$) *diffeomorphisms* (γ and γ^{-1} are differentiable). Note also that $\gamma'(t) > 0$, for all $t \in I$. Mathematically, aligning functions $\mu, f \in \mathcal{F}_I$ equals finding γ as a solution to

$$\gamma = \arg\min_{\gamma^* \in \Gamma_I} d(\mu, f \circ \gamma^*),$$

where $\circ$ denotes function composition, and $d(\cdot, \cdot)$ is some distance measure.

2.1.1. *Fisher-Rao metric*

Srivastava and Klassen (2016, sec. 4.4) argue that a good distance measure for aligning functions $f, g \in \mathcal{F}_I$ should satisfy the *invariance property*, that is

$$d(f, g) = d(f \circ \gamma, g \circ \gamma), \quad \text{for all } \gamma \in \Gamma_I. \tag{2.1}$$

Intuitively, the invariance property is desirable, because if functions f and g are aligned, then if both of them are warped in the same way, they should still be aligned, thus their "distance" should be the same.

However, under the $\mathbb{L}^2$ norm, which has been widely used for functional registration, the action of Γ_I on $\mathcal{F}_I$ is not by *isometries* (distance-preserving transformation), making it inappropriate for direct use in functional alignment, as illustrated by the following example. Given general functions $f, g \in \mathbb{L}^2$ and $\gamma \in \Gamma_I$, we have that

$$\|f \circ \gamma - g \circ \gamma\| = \int_0^1 [f(\gamma(t)) - g(\gamma(t))]^2 dt = \int_0^1 [f(s) - g(s)]^2 \frac{1}{\gamma'(\gamma^{-1}(s))} ds,$$

where $s = \gamma(t)$. In general, $\gamma'(\gamma^{-1}(s)) \neq 1$, then $\|f \circ \gamma - g \circ \gamma\| \neq \|f - g\|$, and the distance between f and g under this norm depends greatly on the choice of γ (Srivastava & Klassen, 2016, sec. 4.5).

This motivates the use of a metric that satisfies the invariance property. We first define a Square-Root Slope Function representation of functions (SRSF, Srivastava & Klassen, 2016, def. 4.2), denoted q here, as

$$\text{SRSF}(f) = q(t) \coloneqq \text{sign}(f'(t))\sqrt{|f'(t)|}. \tag{2.2}$$

The original function can be retrieved by $f(t) = f(0) + \int_0^t q(s)|q(s)|ds$. Note that the SRSF representation of any warping function $\gamma \in \Gamma_I$ is simply $\sqrt{\gamma'}$. Furthermore, for any

$f \in \mathcal{F}_I$, its SRSF q, and $\gamma \in \Gamma_I$, the SRSF of $f \circ \gamma$ has the form

$$\begin{aligned}\tilde{q}(t) &= \operatorname{sign}((f\circ\gamma)'(t))\sqrt{|(f\circ\gamma)'(t)|} = \operatorname{sign}(f'(\gamma(t))\gamma'(t))\sqrt{|f'(\gamma(t))\gamma'(t)|}\\ &= \operatorname{sign}(f'(\gamma(t)))\operatorname{sign}(\gamma'(t))\sqrt{|f'(\gamma(t))|}\sqrt{|\gamma'(t)|}\\ &= \operatorname{sign}(f'(\gamma(t)))\sqrt{|f'(\gamma(t))|}\sqrt{\gamma'(t)} = q(\gamma(t))\sqrt{\gamma'(t)} =: (q,\gamma)(t).\end{aligned}$$

Using this representation, the action of Γ_I on $\mathcal{F}_I$ under the standard $\mathbb{L}^2$ metric is a distance-preserving transformation, and we call the $\mathbb{L}^2$ metric on the SRSF representation space the *Fisher-Rao metric*. The technical details are formally described by Srivastava and Klassen (2016, chap. 4).

2.1.2. *Karcher mean*

With the definition of the Fisher-Rao metric d_{FR}, given $\{f_i\}$, we aim to find $\{\gamma_i\}$ such that

$$\gamma_i = \arg\min_{\gamma\in\Gamma_I} d_{\mathrm{FR}}(\mu, f_i\circ\gamma) = \arg\min_{\gamma\in\Gamma_I} \|\mu_q - (q_i,\gamma)\|, \tag{2.3}$$

where μ_q is the SRSF representation of μ, and q_i is the SRSF representation of f_i.

However, the template μ is often unknown and needs to be estimated. Consider misaligned data as in Figure 2.1a). Even within this limited example, a basic statistical measure such as the cross-sectional mean is inadequate in capturing the intrinsic characteristics of the heartbeat, as demonstrated in Figure 2.2. Instead, an estimate of the template μ can be found iteratively using Algorithm 1, with an example represented by Figure 2.3.

a) Template heartbeats and cross-sectional mean *b) Cross-sectional mean and common template*

FIGURE 2.2. *Comparison of the cross-sectional mean and the common template. Figure a) shows the cross-sectional mean (black) of the misaligned data, while Figure b) shows the comparison to the common template (gray).*

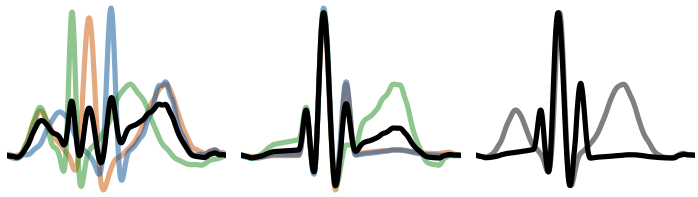
The cross-sectional mean of the aligned functions in Figure 2.3 becomes progressively more similar to the common template regarding the number of peaks, and although the cross-sectional mean does capture the size of the amplitudes of the heartbeats, it fails to extract their positions and thus differs from the true common template. This is a direct consequence of the invariance property, which can be demonstrated as follows. Denote the true common template as μ, the cross-sectional mean of the aligned data as $\hat{\mu}$, and let

Algorithm 1 Iterative template calculation

Input: Functions $\{f_i\}_{i=1}^N$, stopping criterion*
Output: Mean template μ, warping functions $\{\gamma_i\}_{i=1}^N$

$q_i \leftarrow \text{SRSF}(f_i)$ ▷ Calculate the SRSF representation of f_i using Equation (2.2)
$\gamma_i \leftarrow \gamma_{\text{id}}$ ▷ Initialize γ_i as identity warp
while stopping criterion not satisfied **do**
  $\mu \leftarrow N^{-1} \sum_{i=1}^N f_i \circ \gamma_i$ ▷ Calculate the cross-sectional mean of $\{f_i \circ \gamma_i\}_{i=1}^N$
  $\mu_q \leftarrow \text{SRSF}(\mu)$ ▷ Calculate the SRSF representation of μ using Equation (2.2)
  $\gamma_i \leftarrow \arg\min_{\gamma \in \Gamma_I} \|\mu_q - (q_i, \gamma)\|$ ▷ Find γ_i as a solution to Equation (2.3)
end while

*The choice of stopping criterion can be very general, for example, the distance between two consecutive μ.



a) Observed data *b) First iteration* *c) Templates*

FIGURE 2.3. *Example of the iterative template calculation, as described in Algorithm 1. The black lines represent the cross-sectional mean in each step, and the gray line in Figure c) represents the true common template compared to the cross-sectional mean after multiple iterations (black).*

$\hat{\gamma}$ be the warping function for which $\mu = \hat{\mu} \circ \hat{\gamma}$. Then,

$$\tilde{\gamma}_i = \arg\min_{\gamma^* \in \Gamma_I} d_{\text{FR}}(\mu, f_i \circ \gamma^*) = \arg\min_{\gamma^* \in \Gamma_I} d_{\text{FR}}(\hat{\mu} \circ \hat{\gamma}, f_i \circ \gamma^*)$$
$$\overset{(2.1)}{=} \arg\min_{\gamma^* \in \Gamma_I} d_{\text{FR}}(\hat{\mu}, f_i \circ \gamma^* \circ \hat{\gamma}^{-1}).$$

If γ_i is a solution to

$$\gamma_i = \arg\min_{\gamma^* \in \Gamma_I} d_{\text{FR}}(\hat{\mu}, f_i \circ \gamma^*),$$

then $\tilde{\gamma}_i = \gamma_i \circ \hat{\gamma}$. In other words, Algorithm 1 reconstructs the template up to a warp. In the example of Figure 2.3, the distortion is quite substantial, and depending on the misalignment of the observed data, there is no guarantee that the resulting template would have the shape of the true template.

This prompts an alternative concept of the *amplitude* of f. Let q be the SRSF representation of f, the amplitude of f is given by the *orbit*

$$[q] = \text{closure}\{(q, \gamma) \mid \gamma \in \Gamma_I\}.$$

Srivastava and Klassen (2016, def. 4.9) define the *amplitude distance* d_a as

$$d_a([q_1],[q_2]) = \inf_{\gamma_1,\gamma_2\in\tilde{\Gamma}_I} (\|(q_1,\gamma_1)-(q_2,\gamma_2)\|),$$

where $q_1, q_2 \in \mathbb{L}^2$ are the SRSF representations of $f_1, f_2 \in \mathcal{F}_I$, respectively, and $\tilde{\Gamma}_I$ is the set of absolutely continuous boundary-preserving weakly increasing functions. With the help of this distance measure, we can define the *mean amplitude*, also known as the Karcher mean (Srivastava & Klassen, 2016, def. 8.1).

DEFINITION 1 (Karcher mean in $\mathcal{A}$). The Karcher mean $[\mu_q]$ of the amplitudes $\{[q_i]\}_{i=1}^N$ of given functions $\{f_i\}_{i=1}^N$ with SRSFs $\{q_i\}_{i=1}^N$ is the solution to

$$[\mu_q] = \arg\inf_{[q]\in\mathcal{A}} \sum_{i=1}^{N} d_a^2([q],[q_i]),$$

where $\mathcal{A}$ is the quotient space $\mathcal{A} = \mathcal{F}_I/\tilde{\Gamma}_I$.

2.1.3. *Center of an orbit*

Algorithm 1 provides an element of the amplitude of μ. However, to find an element that is the best representative of a set of given functions $\{f_i\}$, we need to determine the *center* of the orbit $[\mu_q]$ with respect to the SRSFs $\{q_i\}$ (Srivastava & Klassen, 2016, def. 8.2).

DEFINITION 2 (Center of an orbit). For a given set of SRSFs $\{q_i\}_{i=1}^N$ and q, an element $\tilde{q}$ of $[q]$ is the center of $[q]$ with respect to the set $\{q_i\}_{i=1}^N$ if the relative phases $\{\gamma_i\}_{i=1}^N$, where $\gamma_i = \arg\inf_{\gamma\in\Gamma_I} \|\tilde{q}-(q_i,\gamma)\|$, have their sample Karcher mean as identity, $\gamma_{\mathrm{id}}(t) = t$.

Given functions $\{f_i\}$ and their SRSF representations $\{q_i\}$, let $[\mu_q]$ be the Karcher mean of amplitudes $\{[q_i]\}$, let q be an element of $[\mu_q]$, and let γ_i be the solution to

$$\gamma_i = \arg\inf_{\gamma\in\Gamma_I} \|q-(q_i,\gamma)\|,$$

then the Karcher mean of warping functions $\{\gamma_i\}$ can be found as $\mu_\gamma(t) = \int_0^t \mu_\psi^2(s)ds$, where μ_ψ is the Karcher mean of the SRSF representations of $\{\gamma_i\}$. The center of the orbit $[\mu_q]$ with respect to $\{q_i\}$ is then given by

$$\tilde{q} = (q, \mu_\gamma^{-1}).$$

Finally, to find the Karcher mean of the SRSF representations of $\{\gamma_i\}$, we use a convenient property of these SRSF representations. Since $\gamma' > 0$ for any warping function γ, its SRSF representation is simply $\psi = \sqrt{\gamma'}$, and furthermore, the set of representations ψ is the positive orthant of the unit sphere, $\mathbb{S}_+^\infty$, in the Hilbert space $\mathbb{L}^2$, because

$$\|\psi\|^2 = \int_0^1 \psi^2(t)dt = \int_0^1 \gamma'(t)dt = \gamma(1)-\gamma(0) = 1.$$

The Karcher mean on $\mathbb{S}_+^\infty$ is defined as follows (Srivastava & Klassen, 2016, def. 8.3)

DEFINITION 3. For a given set of points $\psi_1, \ldots, \psi_N \in \mathbb{S}_+^\infty$, their Karcher mean in $\mathbb{S}_+^\infty$ is defined to be a local minimum of the cost function $\psi \mapsto \sum_{i=1}^N d_\psi^2(\psi, \psi_i)$, where $d_\psi(\psi_1, \psi_2) = \cos^{-1}\left(\int_0^1 \psi_1(t)\psi_2(t)dt\right)$.

The Karcher mean can be calculated using Algorithm 2 (Srivastava & Klassen, 2016, alg. 24) which is based on a fixed-point algorithm, see Bhattacharya and Bhattacharya (2012, chap. 5). Furthermore, this algorithm uses projections to the tangent space $T_\psi(\mathbb{S}_+^\infty)$, which is defined as $T_\psi(\mathbb{S}_+^\infty) = \left\{v \in \mathbb{L}^2 \;\middle|\; \int_0^1 \psi(t)v(t)dt = 0\right\}$. To project points between $\mathbb{S}_+^\infty$ and $T_\psi(\mathbb{S}_+^\infty)$, we utilize the exponential map $\exp_\psi : T_\psi(\mathbb{S}_+^\infty) \to \mathbb{S}_+^\infty$

$$\exp_\psi(v) = \cos(\|v\|)\psi + \sin(\|v\|)\frac{v}{\|v\|}, \tag{2.4}$$

with an inverse $\exp_\psi^{-1} : \mathbb{S}_+^\infty \to T_\psi(\mathbb{S}_+^\infty)$

$$\exp_\psi^{-1}\left(\tilde{\psi}\right) = \frac{\theta}{\sin(\theta)}\left(\tilde{\psi} - \cos(\theta)\psi\right), \quad \text{where } \theta = \cos^{-1}\left(\int_0^1 \psi(t)\tilde{\psi}(t)dt\right), \tag{2.5}$$

see Srivastava and Klassen (2016, sec. 3.2). We refer to Figure 2.4 for an illustration of Algorithm 2. Algorithm 3 shows the iterative calculation of the common template as the center of the mean orbit, and Figure 2.5 illustrates this process.

Algorithm 2 Karcher mean on $\mathbb{S}_+^\infty$

Input: Functions $\{\psi_i\}_{i=1}^N$ on $\mathbb{S}_+^\infty$, stopping criterion ε, step size λ
Output: Karcher mean μ_ψ

$\mu_\psi \leftarrow N^{-1}\sum_{i=1}^N \psi_i$ ▷ Initialize μ_ψ as a cross-sectional mean of $\{\psi_i\}_{i=1}^N$
$\mu_\psi \leftarrow \|\mu_\psi\|^{-1}\mu_\psi$ ▷ Normalize μ_ψ, so it lies on $\mathbb{S}_+^\infty$
$\mu_v \leftarrow \infty$ ▷ Initialize mean μ_v in the tangent space $T_{\mu_\psi}(\mathbb{S}_+^\infty)$
while $\|\mu_v\| \geq \varepsilon$ **do**
  $v_i \leftarrow \exp_{\mu_\psi}^{-1}(\psi_i)$ ▷ Project ψ_i to the tangent space $T_{\mu_\psi}(\mathbb{S}_+^\infty)$ using Equation (2.5)
  $\mu_v \leftarrow N^{-1}\sum_{i=1}^n v_i$ ▷ Calculate the cross-sectional mean of $\{v_i\}$
  $\mu_\psi \leftarrow \exp_{\mu_\psi}(\lambda\mu_v)$ ▷ Project $\lambda\mu_v$ to $\mathbb{S}_+^\infty$ using Equation (2.4)
end while

2.2. QUASI-PERIODIC DATA

Algorithm 3 enables the alignment of individual heartbeats. However, ECGs typically comprise multiple heartbeats, as illustrated in Figure 2.1. We refer to data with this structure as quasi-periodic and conjecture that they stem from a *multiscale warping model*, see Figure 2.6. The multiscale warping model presumes that the underlying data is strictly periodic, yet the observed data is affected by noise in both the time domain and the amplitude, thus the strict periodicity is lost.

Mathematically, the multiscale warping model can be described as follows. Given the common template μ on the interval $[0, \tau]$, the set of subject-specific templates $\{\mu_i\}$ on the

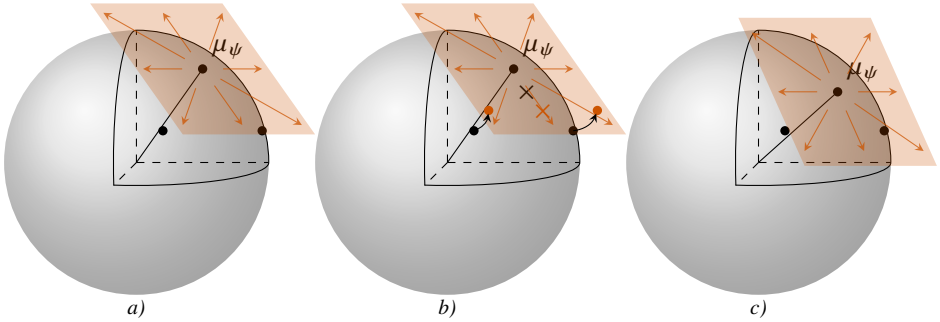


FIGURE 2.4. *Example of one step in Algorithm 2. Figure a) shows points ψ_i on $\mathbb{S}_+^\infty$ and initial guess of μ_ψ (black dots) together with the tangent plane $T_{\mu_\psi}(\mathbb{S}_+^\infty)$ (orange). Figure b) shows the projections v_i (orange dots) of ψ_i to the tangent space $T_{\mu_\psi}(\mathbb{S}_+^\infty)$, cross-sectional mean μ_v of $\{v_i\}$ (orange cross) and the new updated mean μ_ψ (black cross) on $\mathbb{S}_+^\infty$. Finally, Figure c) shows the updated μ_ψ with the new tangent plane.*

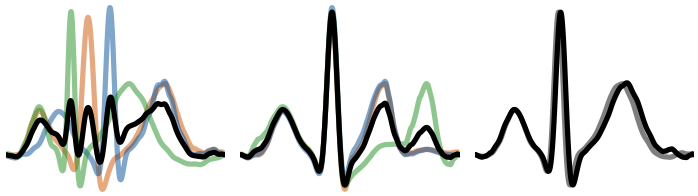


FIGURE 2.5. *Example of the iterative template calculation as the center of the mean orbit, as described in Algorithm 3. The black lines represent the resulting center of the mean orbit in each step, and the gray line in Figure c) represents the true common template compared to the iterative template after multiple iterations (black).*

interval $[0, \tau]$ is generated as

$$\mu_i(t) = \left(\mu \circ \gamma_i^l\right)(t), \tag{2.6}$$

where $\gamma_i^l \in \Gamma_{[0,\tau]}$ are the *local warping functions* as in Figure 2.6d), and the set of functions $\{f_i\}$ on the interval $[0, K\tau]$ is generated as

$$f_i(t) = \left(\text{ext}_{\mathbb{L}^2}^K(\mu_i) \circ \gamma_i^g\right)(t), \tag{2.7}$$

where $\gamma_i^g \in \Gamma_{[0,K\tau]}$ are the *global warping functions* as in Figure 2.6e), K is the (known) number of *quasi-periods*, and $\text{ext}_{\mathbb{L}^2}^K(\mu)$ is the periodic extension of μ to K periods. A periodic extension of a function $f \in \mathcal{F}_{[0,\tau]}$ to K periods is defined on the interval $[0, K\tau]$ as

$$\text{ext}_{\mathbb{L}^2}^K(f)(t + k\tau) \coloneqq f(t) \quad \text{and} \quad \text{ext}_{\mathbb{L}^2}^K(f)(K\tau) \coloneqq f(t) \tag{2.8}$$

for $t \in [0, \tau]$ and $k = 0, \ldots, K-1$. Similarly, we can define a periodic extension of a

Algorithm 3 Iterative template calculation as the center of the mean orbit

Input: Functions $\{f_i\}_{i=1}^N$, stopping criterions c and ε, step size λ
Output: Mean template μ, warping functions $\{\gamma_i\}_{i=1}^N$

$q_i \leftarrow \mathrm{SRSF}(f_i)$ ▷ Calculate the SRSF representation of f_i using Equation (2.2)
$\gamma_i = \gamma_{\mathrm{id}}$ ▷ Initialize γ_i as identity warp
while stopping criterion c not satisfied **do**
  $\mu \leftarrow \frac{1}{N}\sum_{i=1}^N f_i \circ \gamma_i$ ▷ Calculate the cross-sectional mean of $\{f_i \circ \gamma_i\}_{i=1}^N$
  $\mu_q \leftarrow \mathrm{SRSF}(\mu)$ ▷ Calculate the SRSF representation of μ using Equation (2.2)
  $\gamma_i \leftarrow \arg\min_{\gamma\in\Gamma_I} \|\mu_q - (q_i, \gamma)\|$ ▷ Find γ_i as a solution to Equation (2.3)
  $\psi_i \leftarrow \mathrm{SRSF}(\gamma_i)$ ▷ Calculate the SRSF representation of γ_i using Equation (2.2)
  Algorithm 2 with
    Input: Functions $\{\psi_i\}_{i=1}^N$ on $\mathbb{S}_+^\infty$, stopping criterion ε, step size λ
    Output: Karcher mean μ_ψ
  $\mu_\gamma \leftarrow \int_0^1 \mu_\psi^2(t)dt$ ▷ Calculate μ_γ in Γ_I
  $\gamma_i \leftarrow \gamma_i \circ \mu_\gamma^{-1}$ ▷ Warp γ_i by the inverse of their Karcher mean, μ_γ^{-1}
end while
$\mu \leftarrow \frac{1}{N}\sum_{i=1}^N f_i \circ \gamma_i$ ▷ Calculate the cross-sectional mean of $\{f_i \circ \gamma_i\}_{i=1}^N$

warping function $\gamma \in \Gamma_{[0,\tau]}$ to K periods on the interval $[0, K\tau]$ as

$$\mathrm{ext}_\Gamma^K(\gamma)(t + k\tau) \coloneqq \gamma(t) + k\tau \quad \text{and} \quad \mathrm{ext}_\Gamma^K(\gamma)(K\tau) \coloneqq K\tau, \tag{2.9}$$

for $t \in [0, \tau]$ and $k = 0, \ldots, K-1$. We can thus combine Equations (2.6) and (2.7) in a multiscale warping model as

$$\begin{aligned} f_i(t) &= \left(\mathrm{ext}_{\mathbb{L}^2}^K(\mu_i) \circ \gamma_i^g\right)(t) = \left(\mathrm{ext}_{\mathbb{L}^2}^K(\mu \circ \gamma_i^l) \circ \gamma_i^g\right)(t) = \left(\mathrm{ext}_{\mathbb{L}^2}^K(\mu) \circ \mathrm{ext}_\Gamma^K\left(\gamma_i^l\right) \circ \gamma_i^g\right)(t) \\ &= \left(\mathrm{ext}_{\mathbb{L}^2}^K(\mu) \circ \gamma_i^t\right)(t), \end{aligned}$$

where $\gamma_i^t \in \Gamma_{[0,K\tau]}$ are the *total warping functions* as in Figure 2.6f) that are defined as $\gamma_i^t = \mathrm{ext}_\Gamma^K(\gamma_i^l) \circ \gamma_i^g$.

Aligning quasi-periodic data requires special attention, as estimating the common template μ corresponding to a single period is not straightforward. The obvious idea of directly estimating the periodic extension $\mathrm{ext}_{\mathbb{L}^2}^K(\mu)$ as an output of Algorithm 3 does not provide a strictly periodic function, especially when the misalignment is substantial. Hence, it is not possible to extract the single-period common template. Another idea is splitting the functions $f \in \mathcal{F}_{[0,K\tau]}$ into K periods for $t \in [0, \tau]$ by

$$\begin{aligned} \mathrm{split}^K(f)(t) &= (f(t), \ldots, f(t + (k-1)\tau), \ldots, f(t + (K-1)\tau)) \\ &=: (f_1(t), \ldots, f_k(t), \ldots, f_K(t)), \end{aligned} \tag{2.10}$$

and we denote $\mathrm{spl}^K(f) \coloneqq \{f_k \in \mathcal{F}_{[0,\tau]} \mid k = 1, \ldots, K\} = \{f_k\}_{k=1}^K$. Then we estimate the common template as the center of the mean orbit with respect to the amplitudes of $\{f_{i,k}\}_{(i,k)\in(N\times K)}$. Again, in the case of considerable misalignment, the resulting estimate

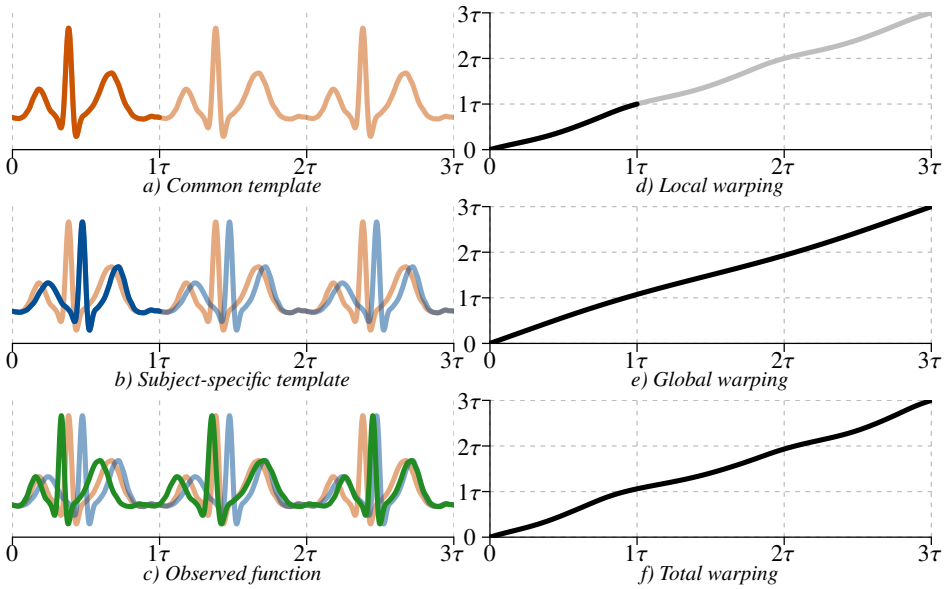


FIGURE 2.6. *Multiscale warping model. Figure a) shows the common template (bright orange) and its periodic extension (pale orange), where τ is the length of one period. Figure b) shows the subject-specific template (bright blue) and its periodic extension (pale blue), as well as the periodic extension of the common template (pale orange). Figure c) shows the observed function (green), together with the periodic extensions of the common and subject-specific templates (pale orange and blue, respectively). Figure d) shows the local warping (black) and its periodic extension (grey), which warps the time domain of the common template in Figure a) to obtain the subject-specific template in Figure b). Figure e) shows the global warping that warps the time domain of the subject-specific template in Figure b) to obtain the observed function in Figure c). Figure f) shows the total warping, which is the composition of the periodic extension of the local warping in Figure d) and the global warping in Figure e).*

differs from the true common template. See Figure 2.7 for an example of these two simple estimates.

Instead, we use a combination of the two simple methods. We still align the whole signal, but estimate the common template as the center of the Karcher mean of single heartbeats. Thus, we extend Algorithm 3 with the following modifications. In Step 4, rather than calculating the cross-sectional mean of the warped functions, we compute the cross-sectional mean of the split of the warped functions as defined in Equation (2.10), $\mu = N^{-1}K^{-1}\sum_{i=1}^{N}\sum_{k=1}^{K}(f_i \circ \gamma_i)_k$. More importantly, in Steps 8 to 10 of Algorithm 3, we calculate the Karcher mean of the split of warping functions as defined in Equation (2.10), $\{\gamma_{i,k}\}_{(i,k)\in(N\times K)}$. Note that when splitting warping functions, they must be normalized to the same interval, which we chose here to be $[0, 1]$, in both their domain and range to apply the exponential and inverse exponential maps as described in Equations (2.4) and (2.5). Henceforth, we will assume that split warping functions are normalized. The detailed procedure of aligning quasi-periodic functional data is formally outlined in Algorithm 4 and illustrated in Figure 2.8.

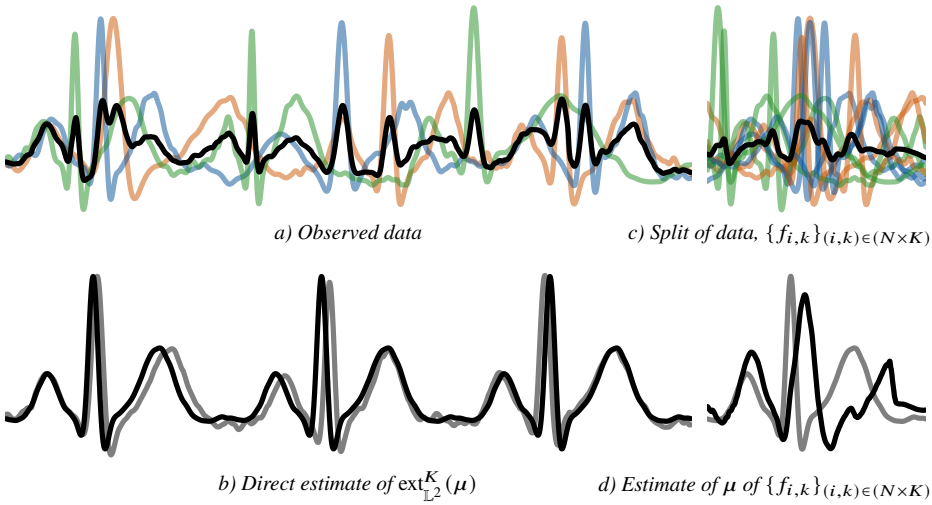


FIGURE 2.7. *Example of simple estimates of the common template of quasi-periodic data. Figure a) shows the observed data and the cross-sectional mean (black). Figure b) shows the center of the orbit (black) of amplitudes of* $\{f_i\}_{i=1}^N$ *resulting from Algorithm 3 and the true common template (grey). Figure c) shows the split of data into three periods using Equation* (2.10) *and the cross-sectional mean (black). Figure d) compares the center of the mean orbit (black) of amplitudes of* $\{f_{i,k}\}_{(i,k)\in(N\times K)}$ *to the true common template (grey).*

2.2.1. *Varying number of quasi-periods*

The number of heartbeats recorded in a standard ECG over a 10-second interval can vary significantly among different subjects due to various factors influencing heart rate. Heart rate variability is affected by individual differences in age, fitness level, and overall health, with athletes typically exhibiting lower resting heart rates compared to non-athletes. Emotional states such as stress, anxiety, or excitement can elevate heart rate, while relaxation can lower it. Additionally, medical conditions like arrhythmias can impact heart rate, as can medications. These factors collectively contribute to the natural variability in the number of heartbeats observed in ECG recordings across different individuals (Billman et al., 2015). Figure 2.9 shows an example of ECG data with varying number of heartbeats.

To handle this kind of data, we modify Algorithm 4 in the following way. Denote the number of heartbeats of each subject as K_i. Then the calculation of the mean in Step 5 is replaced by $\mu = N^{-1}\sum_{i=1}^N {K_i}^{-1}\sum_{k=1}^{K_i}(f_i\circ\gamma_i)_k$, and the Karcher mean in Steps 11 to 13 is calculated in two phases. First, we calculate the Karcher mean of warping functions separately for each subject, and then we compute the Karcher mean of the resulting warping functions.

2.2.2. *Multivariate functional data*

Up to this point, we have focused on univariate functional data. In this section, we expand our approach to address multivariate functional data, which refers to data that consists of multiple functions or curves measured over time on the same subject. Examples of such

Algorithm 4 Iterative template calculation for quasi-periodic functional data

Input: Functions $\{f_i\}_{i=1}^N$, stopping criterions c and ε, step size λ
Output: Mean template μ, warping functions $\{\gamma_i\}_{i=1}^N$

$q_i \leftarrow \mathrm{SRSF}(f_i)$ ▷ Calculate the SRSF representation of f_i using Equation (2.2)
$\gamma_i = \gamma_{\mathrm{id}}$ ▷ Initialize γ_i as identity warp
while stopping criterion c not satisfied **do**
  $\{(f_i \circ \gamma_i)_k\}_{(i,k)\in(N\times K)}$ ▷ Calculate the split of $\{f_i \circ \gamma_i\}_{i=1}^N$ using Equation (2.10)
  $\mu \leftarrow N^{-1}K^{-1}\sum_{i=1}^N \sum_{k=1}^K (f_i \circ \gamma_i)_k$ ▷ Calculate the mean template
  $\mu \leftarrow \mathrm{ext}_{\mathbb{L}}^K(\mu)$ ▷ Calculate the periodic extension of μ using Equation (2.8)
  $\mu_q = \mathrm{SRSF}(\mu)$ ▷ Calculate the SRSF representation of μ using Equation (2.2)
  $\gamma_i = \arg\min_{\gamma\in\Gamma_I} \|\mu_q - (q_i, \gamma)\|$ ▷ Find γ_i as a solution to Equation (2.3)
  $\psi_i \leftarrow \mathrm{SRSF}(\gamma_i)$ ▷ Calculate the SRSF representation of γ_i using Equation (2.2)
  $\{\psi_{i,k}\}_{(i,k)\in(N\times K)}$ ▷ Calculate the split of $\{\psi_i\}_{i=1}^N$ using Equation (2.10)
  Algorithm 2 with
    Input: Functions $\{\psi_{i,k}\}_{(i,k)\in(N\times K)}$ on $\mathbb{S}_+^\infty$, stopping criterion ε, step size λ
    Output: Karcher mean μ_ψ
  $\mu_\gamma \leftarrow \int_0^1 \mu_\psi^2(t)dt$ ▷ Calculate μ_γ in Γ_I
  $\mu_\gamma \leftarrow \mathrm{ext}_\Gamma^K \mu_\gamma$ ▷ Calculate the periodic extension of μ_γ using Equation (2.9)
  $\gamma_i \leftarrow \gamma_i \circ \mu_\gamma^{-1}$ ▷ Warp γ_i by the inverse of their Karcher mean, μ_γ^{-1}
end while
$\{(f_i \circ \gamma_i)_k\}_{(i,k)\in(N\times K)}$ ▷ Calculate the split of $\{f_i \circ \gamma_i\}_{i=1}^N$ using Equation (2.10)
$\mu \leftarrow N^{-1}K^{-1}\sum_{i=1}^N \sum_{k=1}^K (f_i \circ \gamma_i)_k$ ▷ Calculate the mean template

data include gait analysis, where various variables like joint angles are recorded (Baker, 2006), and weather data, which involves measuring multiple factors such as temperature, humidity, wind speed, and precipitation (Hosseini-Nasab & Sharghi, 2024). Another example from the medical sector is electroencephalography,[1] which measures brain activity, and multi-lead ECG.

In multivariate functional data, dimensions often represent different aspects of the same underlying phenomenon. Each function in this type of data should be warped using the same warping function to ensure consistency and to maintain temporal alignment across dimensions, thereby preserving the relationships between them. Applying the same warping function also improves interpretability. For instance, in medical data such as ECG, using a uniform warping function across different leads helps compare and understand the overall heart activity more clearly. This uniformity makes it easier to draw meaningful conclusions from the data.

We denote multivariate functional data as $f_i = (f_{i1}, \ldots, f_{iJ}) \colon I \to \mathbb{R}^J$, and we consider the relevant operations described previously to be element-wise. As an example, the element-wise warping is defined as $(f_i \circ \gamma_i)(t) = ((f_{i1} \circ \gamma_i)(t), \ldots, (f_{iJ} \circ \gamma_i)(t))$, the element-wise SRSF is defined as $q_i(t) = (q_{i1}(t), \ldots, q_{iJ}(t))$, and so on. The template μ is a multivariate function as well, $\mu = (\mu_{\cdot 1}, \ldots, \mu_{\cdot J})$, and the multivariate quasi-periodic

[1] National Health Service. *Electroencephalogram (EEG)* [website], https://www.nhs.uk/conditions/electroencephalogram/, accessed November 30, 2024

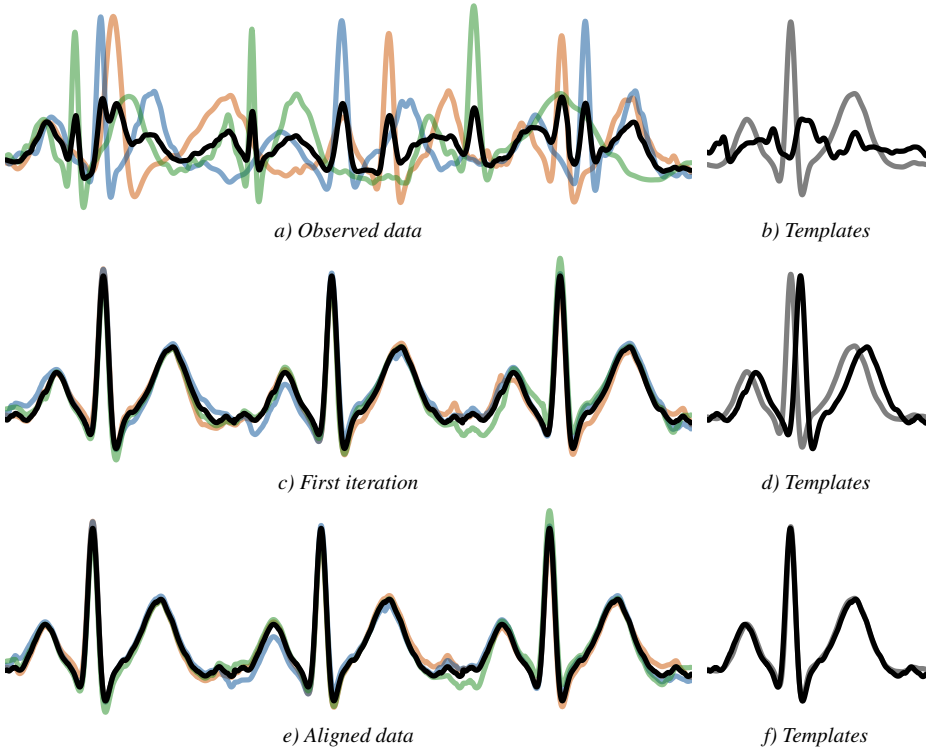
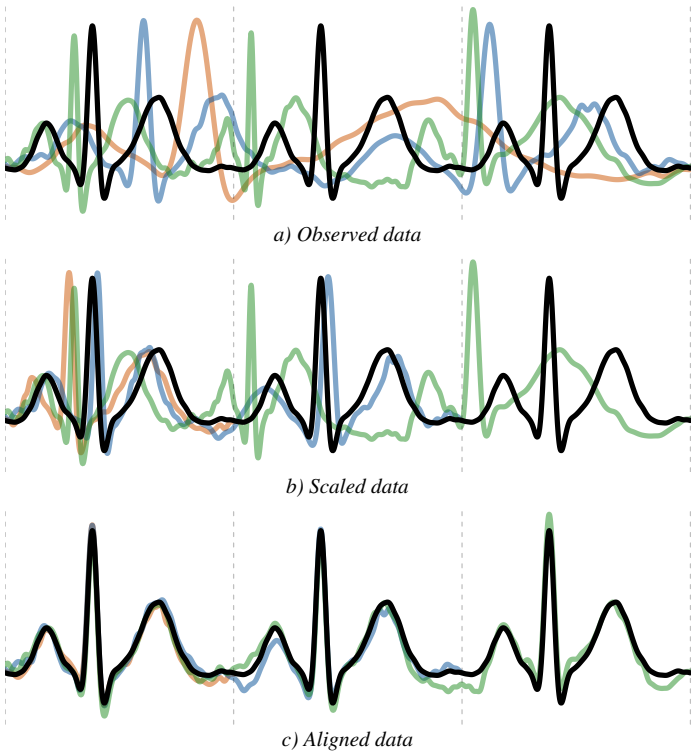


FIGURE 2.8. *Iterative template calculation for quasi-periodic data as the center of the mean orbit of amplitudes* $\{(f_i \circ \gamma_i)_k\}_{(i,k)\in(N\times K)}$*, as described in Algorithm 4. Figure a) shows the observed data with the cross-sectional mean (black). Figure b) shows the cross-sectional mean of* $\{(f_i \circ \gamma_i)_k\}_{(i,k)\in(N\times K)}$ *(black) and the true common template (grey). Figure d) shows the estimated template (black), Figure c) shows the aligned data to the periodic extension of this template after one iteration. Figures f) and e) show the same, but after multiple iterations of Algorithm 4.*

data is generated with

$$\mu_{ij}(t) = \left(\mu_{\cdot j} \circ \gamma_i^l\right)(t),$$
$$f_{ij}(t) = \left(\mathrm{ext}^K_{\mathbb{L}^2}(\mu_{ij}) \circ \gamma_i^g\right)(t).$$



a) Observed data

b) Scaled data

c) Aligned data

FIGURE 2.9. *Example of data with varying number of periods. The vertical dashed lines represent isometric periods. Figure 2.9a) shows the observed data with one, two, and three heartbeats (orange, blue and green lines, respectively). Figure 2.9b) scales the observed data so that the heartbeats span the correct number of periods (orange – one period, blue – two periods, green – three periods). Figure 2.9c) shows the aligned data. The black line is the periodic extension of the true common template.*

3. Time-to-event outcome

Survival analysis, often referred to as *time-to-event analysis*, is a branch of statistics dedicated to understanding the time until a specific event occurs. The event could involve a variety of outcomes, such as the recurrence of a disease or the time until remission. Unlike standard regression techniques, survival analysis focuses on *censored data*, where the full information about the timing of the event is only partially observed.

Key concepts in survival analysis include the survival function, hazard function, and cumulative distribution function. The survival function, $S(t)$, represents the probability of the event occurring after time t, $S(t) = \mathbb{P}(T > t)$, where T is the event time. The hazard function, $h(t)$, or the *instantaneous failure rate*, expresses the rate at which events occur at a specific time, given survival up to that point. It is essentially a measure of risk over time and is given by $h(t) = \lim_{\triangle t \to 0} (\triangle t)^{-1} \mathbb{P}(t \leq T < t + \triangle t \mid T \geq t)$. The cumulative hazard function, $H(t)$, aggregates the hazard over time and is defined as $H(t) = \int_0^t h(s)ds$.

There are several approaches to estimating survival and hazard functions from right-censored data, each suited to different types of data and study designs. The Kaplan-Meier estimator (Kaplan & Meier, 1958) is a widely used non-parametric method for estimating the survival function. The Cox proportional hazards model (Cox, 1972) is a semi-parametric model that relates the hazard function to a set of covariates, allowing for an analysis of factors influencing the hazard rate. The Cox model assumes that hazard ratios are constant over time, thus the name *proportional hazards* model. Additionally, parametric approaches assume a specific probability distribution for survival times, e.g., exponential, Weibull, or log-normal distributions. When these assumptions hold, parametric models can yield more efficient estimates and smoother survival curves.

The individuals may often be at risk for multiple types of events, or they may move through different states over time, such as different stages of disease progression. *Multi-state models* extend survival analysis by accounting for situations where more than one type of event could occur (Andersen & Keiding, 2002). The *competing risk* model is a special case of a multi-state model (Andersen et al., 2002).

Let $(\tilde{T}, \tilde{D}, \boldsymbol{X})$ be the right-censored data with the observed time $\tilde{T} = \min\{T, C\}$, the minimum of the event time T and the censoring time C, with the observed event type $\tilde{D} = \Delta D \in \{0, 1, \ldots, J\}$, where 0 represents the censoring events, $D \in \{1, \ldots, J\}$ represents one of the J (competing) events, and Δ is the censoring indicator, $\Delta = \mathbb{1}(T < C)$, and with the set of scalar and functional covariates $\boldsymbol{X}$.

Further, conditional on covariates $\boldsymbol{x}$, let $S(t \mid \boldsymbol{x}) = \mathbb{P}(T > t \mid \boldsymbol{X} = \boldsymbol{x})$ be the survival function of the event time, $F_j(t \mid \boldsymbol{x}) = \mathbb{P}(T \leq t, D = j \mid \boldsymbol{X} = \boldsymbol{x})$ the cause-specific distribution

function of the event time and

$$h_j(t \mid \boldsymbol{x}) = \lim_{\triangle t \to 0} \frac{1}{\triangle t} \mathbb{P}(t \leq T < t + \triangle t, D = j \mid T \geq t, \boldsymbol{X} = \boldsymbol{x}), \quad j = 1, \ldots, J$$

the cause-specific hazard. Similarly, we denote by $\tilde{S}(t \mid \boldsymbol{x}) = \mathbb{P}(\tilde{T} > t \mid \boldsymbol{X} = \boldsymbol{x})$ the event-free survival function of the observed time, $\tilde{F}_j(t \mid \boldsymbol{x}) = \mathbb{P}(\tilde{T} \leq t, \tilde{D} = j \mid \boldsymbol{X} = \boldsymbol{x})$ the cause-specific distribution function of observed time and

$$\tilde{h}_j(t \mid \boldsymbol{x}) = \lim_{\triangle t \to 0} \frac{1}{\triangle t} \mathbb{P}(t \leq \tilde{T} < t + \triangle t, \tilde{D} = j \mid \tilde{T} \geq t, \boldsymbol{X} = \boldsymbol{x}), \quad j = 0, 1, \ldots, J$$

the cause-specific hazard. Additionally, we let $S_0(t \mid \boldsymbol{x}) = \mathbb{P}(C > t \mid \boldsymbol{X} = \boldsymbol{x})$ be the survival function of censoring time and $G(t \mid \boldsymbol{x}) = \mathbb{P}(C \leq t \mid \boldsymbol{X} = \boldsymbol{x})$ its distribution function. Furthermore, we assume that the event time T and the censoring time C are conditionally independent given $\boldsymbol{X}$, i.e. $T \perp C \mid \boldsymbol{X}$.

The likelihood of the observed data $(\tilde{T}, \tilde{D}, \boldsymbol{X})$ with individual observations $(\tilde{t}_i, \tilde{d}_i, \boldsymbol{x}_i)$, $i = 1, \ldots, N$, is defined as

$$\begin{aligned}
\mathcal{L}(\tilde{t}, \tilde{d}, \boldsymbol{x}; \boldsymbol{\theta}) &= \prod_{i=1}^{N} \mathcal{L}(\tilde{t}_i, \tilde{d}_i, \boldsymbol{x}_i; \boldsymbol{\theta}) = \prod_{i=1}^{N} \mu(\boldsymbol{x}_i; \boldsymbol{\theta}) \tilde{S}(\tilde{t}_i - \mid \boldsymbol{x}_i; \boldsymbol{\theta}) \prod_{j=0}^{J} \tilde{h}_j(\tilde{t}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})^{\mathbb{1}(\tilde{d}_i = j)} \\
&= \prod_{i=1}^{N} \mu(\boldsymbol{x}_i; \boldsymbol{\theta}) \exp\left(- \int_0^{\tilde{t}_i -} \sum_{j=0}^{J} \tilde{h}_j(t \mid \boldsymbol{x}_i; \boldsymbol{\theta}) dt \right) \prod_{j=0}^{J} \tilde{h}_j(\tilde{t}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})^{\mathbb{1}(\tilde{d}_i = j)} \\
&= \prod_{i=1}^{N} \mu(\boldsymbol{x}_i; \boldsymbol{\theta}) \prod_{j=0}^{J} \exp\left(- \int_0^{\tilde{t}_i -} \tilde{h}_j(t \mid \boldsymbol{x}_i; \boldsymbol{\theta}) dt \right) \tilde{h}_j(\tilde{t}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})^{\mathbb{1}(\tilde{d}_i = j)},
\end{aligned}$$

where $\mu(\boldsymbol{x}_i) = \mathbb{P}(\boldsymbol{X} = \boldsymbol{x}_i)$ is the density of $\boldsymbol{X}$ and $\boldsymbol{\theta}$ is the set of additional parameters of the distribution (for example, the shape and scale parameters of a Weibull distribution, Weibull, 1951). The negative log-likelihood is given by

$$\ell(\tilde{t}, \tilde{d}, \boldsymbol{x}; \boldsymbol{\theta}) = \sum_{i=1}^{N} \sum_{j=0}^{J} \left(\int_0^{\tilde{t}_i -} \tilde{h}_j(t \mid \boldsymbol{x}_i; \boldsymbol{\theta}) dt - \mathbb{1}(\tilde{d}_i = j) \log(\tilde{h}_j(\tilde{t}_i \mid \boldsymbol{x}_i; \boldsymbol{\theta})) \right). \tag{3.1}$$

Minimizing the negative log-likelihood for the parameters $\boldsymbol{\theta}$ in Equation (3.1) yields maximum-likelihood estimators of the cause-specific hazards of observed time. Moreover, given the following equalities,

$$\tilde{h}_j(t \mid \boldsymbol{x}; \boldsymbol{\theta}) = \frac{d\tilde{F}_j(t \mid \boldsymbol{x}; \boldsymbol{\theta})/dt}{\tilde{S}(t - \mid \boldsymbol{x}; \boldsymbol{\theta})} \overset{T \perp C \mid X}{=} \frac{dF_j(t \mid \boldsymbol{x}; \boldsymbol{\theta})/dt}{S(t - \mid \boldsymbol{x}; \boldsymbol{\theta})} = h_j(t \mid \boldsymbol{x}; \boldsymbol{\theta})$$

and

$$\tilde{h}_0(t \mid \boldsymbol{x}; \boldsymbol{\theta}) = \frac{d\tilde{F}_0(t \mid \boldsymbol{x}; \boldsymbol{\theta})/dt}{\tilde{S}(t - \mid \boldsymbol{x}; \boldsymbol{\theta})} \overset{T \perp C \mid X}{=} \frac{dG(t \mid \boldsymbol{x}; \boldsymbol{\theta})/dt}{S_0(t - \mid \boldsymbol{x}; \boldsymbol{\theta})} = h_0(t \mid \boldsymbol{x}; \boldsymbol{\theta}),$$

we can reconstruct the cause-specific distribution functions of the underlying event time by

$$F_j(t \mid \boldsymbol{x}; \boldsymbol{\theta}) = \int_0^t \exp\left(-\sum_{j=1}^{J} H_j(s \mid \boldsymbol{x}; \boldsymbol{\theta})\right) h_j(s \mid \boldsymbol{x}; \boldsymbol{\theta}) ds, \quad j = 1, \ldots, J. \tag{3.2}$$

Depending on the underlying distribution of the event times, as well as the observed and censoring times, it is possible to derive explicit formulas for the hazard rate, cumulative hazard function, and the distribution functions as presented in Equation 3.2.

For the Weibull distribution (Weibull, 1951), these functions can be expressed in terms of the *shape* parameter, $k > 0$, and the *scale* parameter, $\lambda > 0$. Here we assume that both these parameters can depend on the covariates $\boldsymbol{X}$ and some additional parameters $\boldsymbol{\theta}$, thus $k \coloneqq k(\boldsymbol{x}; \boldsymbol{\theta})$ and $\lambda \coloneqq \lambda(\boldsymbol{x}; \boldsymbol{\theta})$. The hazards and the cumulative hazards in Equation (3.2) can be expressed as

$$h(t \mid \boldsymbol{x}; \boldsymbol{\theta}) = \frac{k(\boldsymbol{x}; \boldsymbol{\theta})}{\lambda(\boldsymbol{x}; \boldsymbol{\theta})} \left(\frac{t}{\lambda(\boldsymbol{x}; \boldsymbol{\theta})}\right)^{k(\boldsymbol{x};\boldsymbol{\theta})-1} \quad \text{and} \quad H(t \mid \boldsymbol{x}; \boldsymbol{\theta}) = \left(\frac{t}{\lambda(\boldsymbol{x}; \boldsymbol{\theta})}\right)^{k(\boldsymbol{x};\boldsymbol{\theta})}.$$

In the case of a piecewise constant hazard function, we can obtain a closed-form expression for the distribution function. Let $\mathcal{T}$ be an ordered set of evaluation times, $\mathcal{T} = \{t_1 = 0, t_2, \ldots, t_P \mid l < m \Leftrightarrow t_l < t_m\}$, and let $h(t \mid \boldsymbol{x}; \boldsymbol{\theta}) =: h_p$ be the constant hazard for $t_p \leq t < t_{p+1}$. The cumulative hazard can be calculated iteratively as

$$H(t \mid \boldsymbol{x}; \boldsymbol{\theta}) = H(t_p \mid \boldsymbol{x}; \boldsymbol{\theta}) + (t - t_p) h_p, \tag{3.3}$$

where $H(0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 0$. Similarly, the survival function of event time can be derived as

$$\begin{aligned} S(t \mid \boldsymbol{x}; \boldsymbol{\theta}) &= \exp\left(-\sum_{j=1}^{J} H_j(t \mid \boldsymbol{x}; \boldsymbol{\theta})\right) \overset{(3.3)}{=} \exp\left(-\sum_{j=1}^{J} H_j(t_p \mid \boldsymbol{x}; \boldsymbol{\theta}) - (t - t_p) \sum_{j=1}^{J} h_{pj}\right) \\ &= S(t_p \mid \boldsymbol{x}; \boldsymbol{\theta}) \exp\left(-(t - t_p) \sum_{j=1}^{J} h_{pj}\right), \end{aligned} \tag{3.4}$$

where $S(0 \mid \boldsymbol{x}; \boldsymbol{\theta}) = 1$. And finally, the cause-specific risk can also be calculated iteratively

by

$$
\begin{aligned}
F_j(t \mid \boldsymbol{x};\boldsymbol{\theta}) &= \int_0^t S(s \mid \boldsymbol{x};\boldsymbol{\theta}) h_j(s \mid \boldsymbol{x};\boldsymbol{\theta}) ds \\
&= \int_0^{t_p} S(s \mid \boldsymbol{x};\boldsymbol{\theta}) h_j(s \mid \boldsymbol{x};\boldsymbol{\theta}) ds + \int_{t_p}^t S(s \mid \boldsymbol{x};\boldsymbol{\theta}) h_j(s \mid \boldsymbol{x};\boldsymbol{\theta}) ds \\
&= F_j(t_p \mid \boldsymbol{x};\boldsymbol{\theta}) + h_{pj} \int_{t_p}^t S(s \mid \boldsymbol{x};\boldsymbol{\theta}) ds \\
&\overset{(3.4)}{=} F_j(t_p \mid \boldsymbol{x};\boldsymbol{\theta}) + h_{pj} S(t_p \mid \boldsymbol{x};\boldsymbol{\theta}) \int_{t_p}^t \exp\left(-(s - t_p) \sum_{j=1}^J h_{pj}\right) ds \\
&= F_j(t_p \mid \boldsymbol{x};\boldsymbol{\theta}) + \frac{h_{pj}}{\sum_{j=1}^J h_{pj}} S(t_p \mid \boldsymbol{x};\boldsymbol{\theta}) \left(1 - \exp\left(-(t - t_p) \sum_{j=1}^J h_{pj}\right)\right),
\end{aligned}
$$

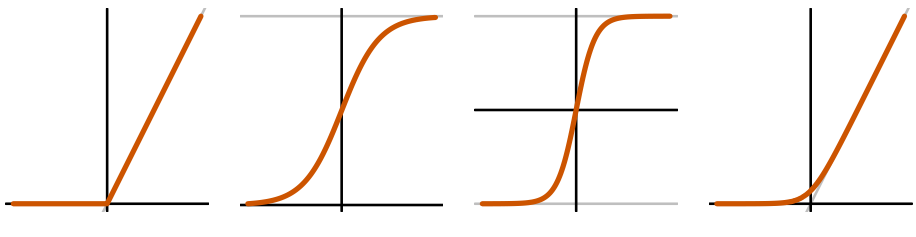
where $F(0 \mid \boldsymbol{x};\boldsymbol{\theta}) = 0$.

4. Neural networks

In this chapter, we explore the fundamental concepts of neural networks and deep learning, aiming to provide a clear understanding. We then consider specific applications tailored to address distinct challenges in various problem domains, such as the analysis of high-dimensional data, including multivariate time series.

4.1. FULLY CONNECTED FEEDFORWARD NEURAL NETWORKS

A fully connected feedforward neural network is one of the most fundamental architectures in the field of neural networks and serves as a building block for more complex neural network models that are widely used in various applications, including image recognition, natural language processing, and predictive analysis. It consists of layers labeled by $l = 0, 1 \ldots, L$, where $l = 0$ denotes the *input layer*, $l = L$ denotes the *output layer*, and the layers in between, $0 < l < L$, are called the *hidden layers*. We use a superscript (l) to refer to a particular layer. Each layer has a *dimension* $d^{(l)}$ specifying the number of *nodes* in this layer, labeled as $1, \ldots, d^{(l)}$. Layers $l < L$ contain a special node called the *bias node*, labeled as 0, which is always set to have the value 1. Each layer $l > 0$ has an input signal $\boldsymbol{s}^{(l)}$ and each layer $l < L$ has an output signal $\boldsymbol{x}^{(l)} = \left(1, \sigma^{(l)}(\boldsymbol{s}^{(l)})^T\right)^T$. Here $\sigma^{(l)}$ represents the *activation function* and is typically applied element-wise. Figure 4.1 illustrates examples of commonly used activation functions.



a) ReLU $f(x) = \max(0, x)$

b) sigmoid $f(x) = \dfrac{1}{1 + \exp(-x)}$

c) tanh $f(x) = \dfrac{1 - \exp(-2x)}{1 + \exp(-2x)}$

d) softplus $f(x) = \log(1 + \exp(x))$

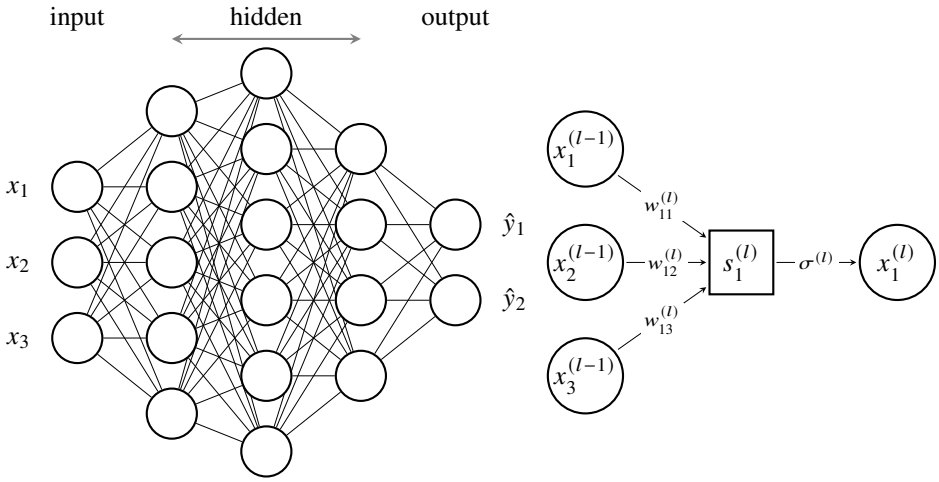
FIGURE 4.1. *Commonly used activation functions. The black lines are the lines $x = 0$, and $y = 0$, the diagonal gray lines represent $y = x$, and the horizontal gray lines represent $y = 1$ or $y = -1$.*

The output of the neural network is calculated through *forward propagation*. Let $\boldsymbol{x}$ be the neural network input, let $\boldsymbol{y}$ be the true outcome, and let $\hat{\boldsymbol{y}} = \boldsymbol{x}^{(L)}$ be the neural network

output. Then

$$\boldsymbol{s}^{(l)} = \boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)} \quad \text{and} \quad \boldsymbol{x}^{(l)} = \begin{pmatrix} 1 \\ \sigma^{(l)}(\boldsymbol{s}^{(l)}) \end{pmatrix}, \tag{4.1}$$

where $\boldsymbol{W}^{(l)}$ is the $d^{(l)} \times (d^{(l-1)} + 1)$ dimensional matrix of *weights* with components w_{jk}, the input is $\boldsymbol{x}^{(0)} = \boldsymbol{x}$ and the output is $\boldsymbol{x}^{(L)} = \sigma^{(L)}(\boldsymbol{s}^{(L)}) = \hat{\boldsymbol{y}}$. We denote $\boldsymbol{w}$ the set of all the weight matrices, $\boldsymbol{w} = \{\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(L)}\}$. The neural network and the forward propagation are typically represented by diagrams as in Figure 4.2, where the direction of the arrows illustrates the iterative dependency between the layers.



a) Neural network diagram *b) Forward propagation*

FIGURE 4.2. *An example of a fully connected feedforward neural network. In Figure a), the input is* $\boldsymbol{x} = (x_1, x_2, x_3)^T$ *and the output is* $\boldsymbol{x}^{(L)} = \hat{\boldsymbol{y}} = (\hat{y}_1, \hat{y}_2)^T$. *The circles represent the nodes of the individual layers, and the lines show the dependency between them. Each line corresponds to an element of the weight matrices* $\boldsymbol{W}^{(l)}$, $0 < l$. *The bias nodes in each layer* $l < L$ *are usually not depicted in these diagrams. Figure b) shows the forward propagation as described in Equation* (4.1).

4.1.1. *Backpropagation*

The weights of the neural network are the learnable coefficients of the model that need to be estimated. The main component in estimating these weights or *training the model*, is the *loss function*, which is the objective function measuring how well the neural network's predictions match the actual target values. Common loss functions are the *mean squared error* (MSE) for regression tasks,

$$\ell(y, \boldsymbol{x}; \boldsymbol{w}) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2,$$

binary cross-entropy for classification tasks,

$$\ell(y, \boldsymbol{x}; \boldsymbol{w}) = -\frac{1}{N} \sum_{i=1}^{N} \left[y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i) \right],$$

and *categorical cross-entropy* for multi-class classification tasks,

$$\ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w}) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log(\hat{y}_{ic}),$$

where N is the number of observations and C is the number of classes. The weights $\boldsymbol{w}$ are then updated iteratively by utilizing the *(batch) gradient descent* as

$$\boldsymbol{w} \leftarrow \boldsymbol{w} - \eta \nabla \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w}),$$

where ∇ is the gradient of the loss function ℓ and η is the step size, also called the *learning rate*. The gradient $\nabla \ell$ is calculated through iterative application of the chain rule, termed the *backpropagation* (Abu-Mostafa et al., 2012). To ease the notation, we let $\sigma^{(l)} = \sigma$, $l = 1, \ldots, L$. The *sensitivity vector* for one observation $(\boldsymbol{x}, \boldsymbol{y}) = (\boldsymbol{x}_i, \boldsymbol{y}_i)$ is defined as

$$\delta^{(l)} = \frac{\partial \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})}{\partial \boldsymbol{s}^{(l)}}.$$

The loss $\ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})$ only depends on $\boldsymbol{s}^{(l)}$ through $\boldsymbol{x}^{(l)}$, thus

$$\delta_j^{(l)} = \frac{\partial \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})}{\partial s_j^{(l)}} = \frac{\partial \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})}{\partial x_j^{(l)}} \frac{\partial x_j^{(l)}}{\partial s_j^{(l)}} = \sigma' \left(s_j^{(l)} \right) \frac{\partial \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})}{\partial x_j^{(l)}}. \tag{4.2}$$

Additionally, the loss $\ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})$ depends on $\boldsymbol{x}^{(l)}$ through all the components of $\boldsymbol{s}^{(l+1)}$, thus

$$\frac{\partial \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})}{\partial x_j^{(l)}} = \sum_{k=1}^{d^{(l+1)}} \underbrace{\frac{\partial \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})}{\partial s_k^{(l+1)}}}_{(4.2)} \underbrace{\frac{\partial s_k^{(l+1)}}{\partial x_j^{(l)}}}_{(4.1)} = \sum_{k=1}^{d^{(l+1)}} \delta_k^{(l+1)} w_{jk}^{(l+1)},$$

yielding iterative formula for the sensitivity

$$\delta_j^{(l)} = \sigma' \left(s_j^{(l)} \right) \sum_{k=1}^{d^{(l+1)}} \delta_k^{(l+1)} w_{jk}^{(l+1)}. \tag{4.3}$$

Finally, since $\ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})$ only depends on $w_{jk}^{(l)}$ through $s_j^{(l)}$, the gradient of $\ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})$ with respect to the weight parameters can be written as

$$\frac{\partial \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})}{\partial w_{jk}^{(l)}} = \underbrace{\frac{\partial \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w})}{\partial s_j^{(l)}}}_{(4.2)} \underbrace{\frac{\partial s_j^{(l)}}{\partial w_{jk}^{(l)}}}_{(4.1)} = \delta_j^{(l)} x_k^{(l)}. \tag{4.4}$$

The backpropagation algorithm is described in Algorithm 5.

Algorithm 5 Backpropagation (Abu-Mostafa et al., 2012)

Input: Data $\{(\boldsymbol{x}_i, \boldsymbol{y}_i)\}_{i=1}^N$, weights $\boldsymbol{w} = \{\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(L)}\}$, activation functions $\sigma^{(l)}$, $l = 1, \ldots, L$

Output: Gradient $\nabla \ell(\boldsymbol{y}, \boldsymbol{x}; \boldsymbol{w}) = \{G^{(1)}, \ldots, G^{(L)}\}$

1: $G^{(l)} \leftarrow 0$ ▷ Initialize gradient
2: **for** $i = 1$ **to** N **do**
3: $x^{(0)} \leftarrow \boldsymbol{x}_i$ ▷ Initialize input
4: Compute $x^{(l)}$, $l = 1, \ldots, L$ ▷ Forward propagation using Equation (4.1)
5: Compute $\delta^{(l)}$, $l = L, \ldots, 1$ ▷ Calculate sensitivities using Equation (4.3)
6: $G^{(l)} \leftarrow G^{(l)} + N^{-1} \left[\boldsymbol{x}^{(l-1)} (\delta^{(l)})^T\right]$, $l = 1, \ldots, L$ ▷ Update gradient using Equation (4.4)
7: **end for**

Stochastic and mini-batch gradient descent. Algorithm 5 describes the batch gradient descent, which computes the gradient of the loss function with respect to the weight parameters using the entire dataset. The weight parameters are only updated after considering all training examples, ensuring high accuracy. This means that the batch gradient descent can be very slow and computationally expensive. Additionally, in the case of large datasets, it can require a significant amount of memory.

Stochastic gradient descent (SGD, Adigun & Yinka-Banjo, 2022), on the other hand, updates the parameters for each training example individually. Instead of computing the gradient over the entire dataset, SGD computes the gradient for a single training example and updates the parameters immediately. This results in much faster iterations and can lead to quicker convergence, especially for large datasets. However, the updates can be noisy because they are based on individual examples, which can cause the loss function to fluctuate rather than decrease smoothly. Nevertheless, the noise can help the algorithm escape local minima and potentially find a better global minimum.

Mini-batch gradient descent combines the properties of batch gradient descent and SGD. It splits the training dataset into small batches and computes the gradient for each mini-batch. The parameters are then updated after processing each mini-batch. This approach combines the advantages of both batch gradient descent and SGD. It reduces the variance of the parameter updates, leading to more stable convergence compared to SGD, while still being more efficient and faster than batch gradient descent.

4.2. CONVOLUTIONAL NEURAL NETWORK

Convolutional Neural Networks (CNN, Goodfellow et al., 2016, chap. 9) are a class of deep neural networks that have proven effective in areas such as image recognition, medical image analysis, time series, and natural language processing. The main architectural difference between CNNs and fully connected neural networks is the use of *convolutional layers*, which is designed to automatically and adaptively learn spatial/temporal relationships from the input data. Each node in a convolutional layer is only connected to a small, localized region of the previous layer. The nodes in a layer l can be calculated by "sliding" a *kernel/filter* across the nodes in the previous layer. Additionally, each convolutional layer consists of one or more channels, which we can think of as an additional dimension of the data.

Let the nodes in each layer l be a 3-dimensional array $\boldsymbol{x}^{(l)}$, where the last index represents the channels. Let $P^{(l)}$ be the number rows, let $J^{(l)}$ be the number of columns, and let $K^{(l)}$ be the number of channels of the data in layer l. Let $\boldsymbol{W}^{(l),k} = (w_{mn\ell}^{(l),k})$, $k = 1, \ldots, K^{(l)}$, be filters of size $\varphi^{(l)}$, yielding $\boldsymbol{W}^{(l),k} \in \mathbb{R}^{\varphi^{(l)} \times \varphi^{(l)} \times K^{(l-1)}}$. Then the input signal $\boldsymbol{s}^{(l)}$ is calculated using *cross-correlation* as

$$s_{pjk}^{(l)} = w_0^{(l),k} + \sum_{\ell=1}^{K^{(l-1)}} \sum_{m=1}^{\varphi^{(l)}} \sum_{n=1}^{\varphi^{(l)}} w_{mn\ell}^{(l),k} x_{p+m-1,j+n-1,\ell}^{(l-1)}, \tag{4.5}$$

where $p = 1 \ldots, P^{(l-1)} - \varphi^{(l)} + 1$, $j = 1 \ldots, J^{(l-1)} - \varphi^{(l)} + 1$, and $w_0^{(l),k}$ is the weight parameter corresponding to a bias node. For illustration, refer to Figure 4.3.

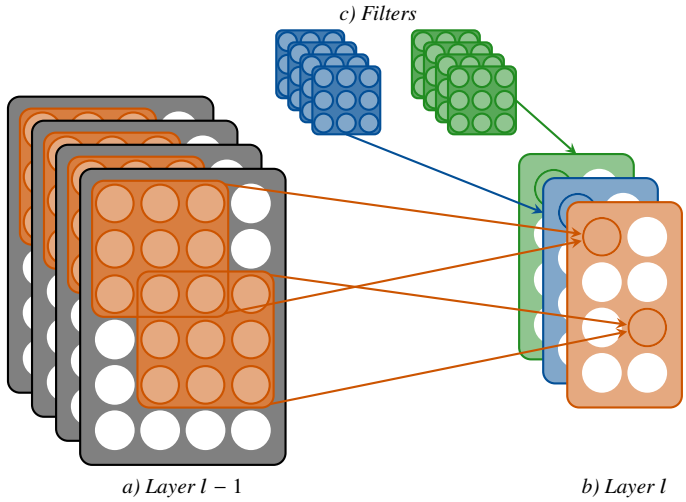


FIGURE 4.3. *Convolutional layer. Figure a) represents the layer* $l-1$ *with* 4 *channels of* 6×4 *units. The orange squares represent a filter of size 3, corresponding to dimensions* $3 \times 3 \times 4$ *(filter size* $\times$ *filter size* $\times$ *number of channels in layer* $l-1$*), at two different spatial locations starting at the points* $(1, 1)$ *and* $(3, 2)$*, producing the first (orange) channel in Figure b) of size* 4×2 *using Equation* (4.5)*. The rest of the channels in Figure b) are produced by different filters, represented in Figure c). Note that all filters have the size* $3 \times 3 \times 4$*.*

It is clear from Equation (4.5) and Figure 4.3 that using cross-correlation in this manner results in a reduction of layer dimensions for $\varphi > 1$. To control the spatial dimensions of the data as it passes through the convolutional layers, a technique called *padding* is employed. The most common form is the *same padding*, also known as *zero-padding*, which involves adding extra zero-valued nodes around the borders of the output signals in each layer, ultimately preventing the data dimensions from shrinking.

The output from the last convolutional layer, which is usually a multidimensional tensor, is often flattened into a one-dimensional vector. This vector then serves as an input to a subsequent fully connected neural network.

One-dimensional convolution. One-dimensional (1D) convolution is a fundamental operation in signal processing, particularly suited for analyzing multivariate time-series data, such as multi-lead ECG. When dealing with multivariate data, each time step in the input sequence consists of multiple values, one for each feature, or lead in the case of ECG. The input to a 1D convolutional layer for multivariate data is typically a 2-dimensional array, where the first dimension represents the time steps and the second dimension represents the different features. Let $P^{(l)}$ be the temporal dimension of layer l and let $K^{(l)}$ be the number of channels in layer l. Let $\boldsymbol{W}^{(l),k} = (w_{m\ell}^{(l),k}) \in \mathbb{R}^{\varphi^{(l)} \times K^{(l-1)}}$, $k = 1, \ldots, K^{(l)}$, be the filters of size $\varphi^{(l)}$. The cross-correlation in Equation (4.5) reduces to

$$s_{pk}^{(l)} = w_0^{(l),k} + \sum_{\ell=1}^{K^{(l-1)}} \sum_{m=1}^{\varphi^{(l)}} w_{m\ell}^{(l),k} x_{p+m-1,\ell}^{(l-1)} = w_0^{(l),k} + \boldsymbol{e}^T \left(\boldsymbol{W}^{(l),k} \odot \boldsymbol{x}_{p:\varphi^{(l)}}^{(l-1)} \right) \boldsymbol{e}, \tag{4.6}$$

where $\odot$ is element-wise multiplication, $\boldsymbol{e}$ is a vector of ones, and $\boldsymbol{x}_{p:\varphi^{(l)}}^{(l-1)} = (x_{ij}^{(l-1)})$ with $p \le i < p+\varphi^{(l)}$, $1 \le j \le K^{(l-1)}$. Refer to Figure 4.4 for an illustration of 1D convolution.

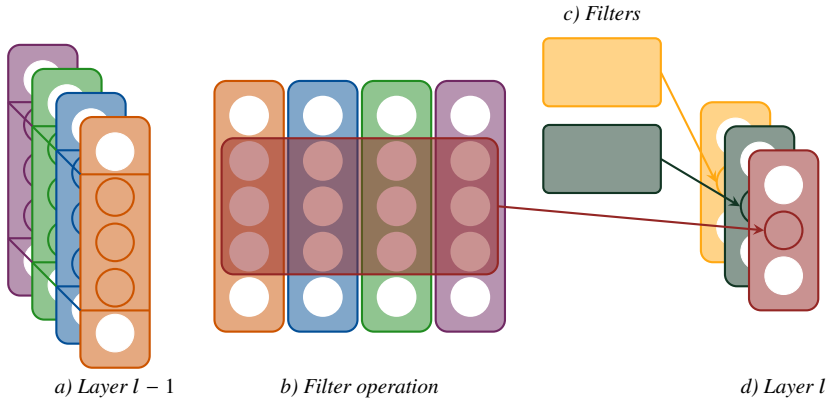


FIGURE 4.4. *1D convolutional layer. Figure a) represents the layer $l-1$ with 4 channels of 5 units. Figure b) shows the operation of the filter on this layer at the temporal location 2, producing the second node of the first channel in layer l in Figure d) using Equation* (4.6)*. The other two channels of layer l are produced by applying the filters in Figure c).*

Equivalence between 1D CNN with a filter of size one and fully connected NN. Consider a fully connected NN and 1D CNN with a filter of size one with layers $l = 0, \ldots, L$,

and a functional/time series input of length $P, \boldsymbol{x} \in \mathbb{R}^P$. In the case of a fully connected neural network, the input would be the values of the functional input at time points $t_1, \ldots, t_P$, separately, with a forward propagation for layers $l < L$ given by

$$\boldsymbol{x}_p^{(0)} = (1, x_p)^T, \quad \boldsymbol{s}_p^{(l)} = \boldsymbol{W}^{(l)} \boldsymbol{x}_p^{(l-1)}, \quad \boldsymbol{x}_p^{(l)} = (1, \sigma^{(l)}(\boldsymbol{s}_p^{(l)})^T)^T, \tag{4.7}$$

where x_p is the value of the time series at time point t_p, and $\boldsymbol{x}_p^{(L)} = \sigma^{(l)}(\boldsymbol{s}_p^{(L)})$. The subscript p in $\boldsymbol{s}_p^{(l)}$ and $\boldsymbol{x}_p^{(l)}$ denote the input signal and the output signal of layer l corresponding to the input x_p. Note that $\boldsymbol{s}_p^{(l)}$ and $\boldsymbol{x}_p^{(l)}$ are vectors of dimension $d^{(l)}$ and $d^{(l)} + 1$, respectively.

In the case of 1D CNN with filters of size one and time series input $\boldsymbol{x} \in \mathbb{R}^P$, we have the following. Since the filter size is one, each node in the convolutional layers depends only on the nodes in the previous layer with the same temporal index, p, see Figure 4.5a). Let $\bar{\boldsymbol{W}}^{(l),k} = (w_\ell^{(l),k}) \in \mathbb{R}^{K^{(l-1)}}$, $k = 1, \ldots, K^{(l)}$, be the filters in layer l. Note that the weights $\bar{\boldsymbol{W}}^{(l),k}$ are now one-dimensional arrays/vectors. The cross-correlation in Equation (4.6) reduces to

$$s_{pk}^{(l)} = w_0^{(l),k} + \sum_{\ell=1}^{K^{(l-1)}} w_\ell^{(l),k} x_{p,\ell}^{(l-1)} = w_0^{(l),k} + (\bar{\boldsymbol{W}}^{(l),k})^T \bar{\boldsymbol{x}}_p^{(l-1)} = (\boldsymbol{W}^{(l),k})^T \boldsymbol{x}_p^{(l-1)},$$

where $\bar{\boldsymbol{x}}_p^{(l-1)}$ is the "slice" of layer $l-1$ at the temporal location $p, \boldsymbol{x}_p^{(l-1)} = (1, (\bar{\boldsymbol{x}}_p^{(l-1)})^T)^T$, and $\boldsymbol{W}^{(l),k} = (w_0^{(l),k}, (\bar{\boldsymbol{W}}^{(l),k})^T)^T$. Furthermore, we can collect the elements $s_{pk}^{(l)}$ in vectors that can be calculated by

$$\boldsymbol{s}_p^{(l)} = \boldsymbol{W}^{(l)} \boldsymbol{x}_p^{(l-1)},$$

where $\boldsymbol{W}^{(l)} = (\boldsymbol{W}^{(l),1}, \ldots, \boldsymbol{W}^{(l),K^{(l)}})^T$ is a matrix of all the weights between layers $l-1$ and l. Finally, the vectors $\boldsymbol{s}_p^{(l)}$ and $\boldsymbol{x}_p^{(l)}$ can be collected in matrices and with the forward propagation for layers $l < L$ given by

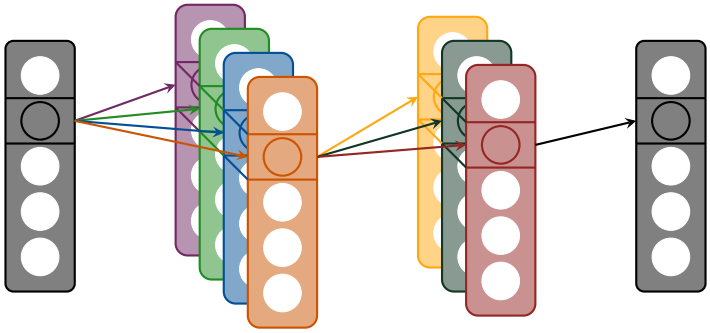
$$\begin{aligned} \boldsymbol{x}^{(0)} &= (1, x_1, \ldots, x_P)^T, \\ \boldsymbol{s}^{(l)} &= (\boldsymbol{s}_1^{(l)}, \ldots, \boldsymbol{s}_P^{(l)})^T = (\boldsymbol{W}^{(l)} \boldsymbol{x}_1^{(l-1)}, \ldots, \boldsymbol{W}^{(l)} \boldsymbol{x}_P^{(l-1)})^T, \\ \boldsymbol{x}_p^{(l)} &= (1, \sigma^{(l)}(\boldsymbol{s}_p^{(l)})^T)^T, \\ \boldsymbol{x}^{(l)} &= (\boldsymbol{x}_1^{(l)}, \ldots, \boldsymbol{x}_P^{(l)})^T, \end{aligned} \tag{4.8}$$

with the output layer $\boldsymbol{x}^{(L)} = (\sigma^{(L)}(\boldsymbol{x}_1^{(L)}), \ldots, \sigma^{(L)}(\boldsymbol{x}_P^{(L)}))^T$.

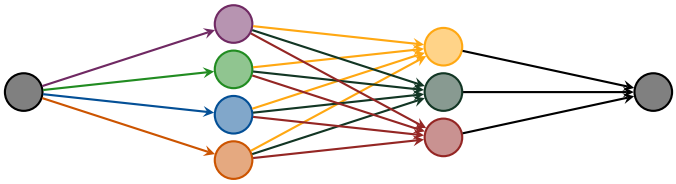
From Equations (4.7) and (4.8), we can see that if the number of filters in the 1D convolutional neural network corresponds to the number of units in the fully connected NN, then backpropagation estimates the same weight parameters $\boldsymbol{W}^{(l)}$. Refer to Figure 4.5 for an illustration of the equivalence.

4.3. ALIGNMENT

We leverage 1D CNN for the alignment of multivariate quasi-periodic functional data as outlined in Chapter 2 and Algorithm 4. Specifically, we use a neural network as the opti-



a) 1D CNN with a filter of size one



b) Fully connected neural network

FIGURE 4.5. *Equivalence between 1D CNN with a filter of size one and a fully connected neural network. Figure a) shows the 1D CNN with four layers and 4, 3, and 1 filters, respectively. Since the size of all the filters is one, each node in each layer only depends on the nodes in the previous layer with the same "temporal" index (2 in the example of Figure a)), which is illustrated by the arrows between the layers. Considering only one temporal location, all the connections can be represented using the diagram in Figure b), which is equivalent to a diagram of a fully connected NN. This is formally shown in Equations* (4.7) *and* (4.8)*.*

mizer in Step 8, thus we use the Fisher-Rao metric as the loss function. Given the multivariate quasi-periodic data $\{f_i = (f_{i1}, \ldots, f_{iJ})\}$ observed at P time points and their SRSF representations $\{q_i\}$, the loss function has the following form

$$\ell(f; \boldsymbol{w}) = \frac{1}{NJ} \sum_{i=1}^{N} \sum_{j=1}^{J} \|\mu_{q_j} - (q_{ij}, \gamma_i)\|,$$

where μ_{q_j} is an element of the current Karcher mean of the amplitudes given by $\{q_{i_j}\}$ and $\{\gamma_i\}$ are the output of the 1D CNN.

Let $\boldsymbol{s}^{(L)} \in \mathbb{R}^P$ be the input signal to the output layer. We use a *simplex* activation function based on the unit simplex inverse transform (Stan Development Team, 2024) to ensure, that the resulting warping functions are boundary-preserving diffeomorphisms as follows. First, we apply a modified sigmoid activation function with an offset to obtain new coordinates $\boldsymbol{z} \in \mathbb{R}^P$

$$z_p = \begin{cases} 0, & p = 1, \\ \text{sigmoid}(s_p^{(L)} - \log(P - p)), & 1 < p < P. \\ 1, & p = P. \end{cases} \tag{4.9}$$

This vector is then used to determine the unit simplex $\boldsymbol{x} \in \mathbb{R}^P$ as

$$x_1 = 0 \quad \text{and} \quad x_p = \left(1 - \sum_{p'=1}^{p-1} x_{p'}\right) z_p.$$

This transformation ensures that $x_p \geq 0$, $p = 1, \ldots, P$, and $\sum_{p=1}^{P} x_p = 1$. Finally, we transform this unit simplex to a discretized warping function $\boldsymbol{\gamma} \in \mathbb{R}^P$ by

$$\gamma_p = \sum_{p'}^{p} x_{p'}.$$

The offset $-\log(P-p)$ in Equation (4.9) ensures that a zero vector $\boldsymbol{s}_p^{(L)}$ is mapped to the simplex $\boldsymbol{x} = (0, (P-1),^{-1}, \ldots, (P-1)^{-1})$, which is then mapped to the identity warp.

4.4. FUNCTIONAL INPUT

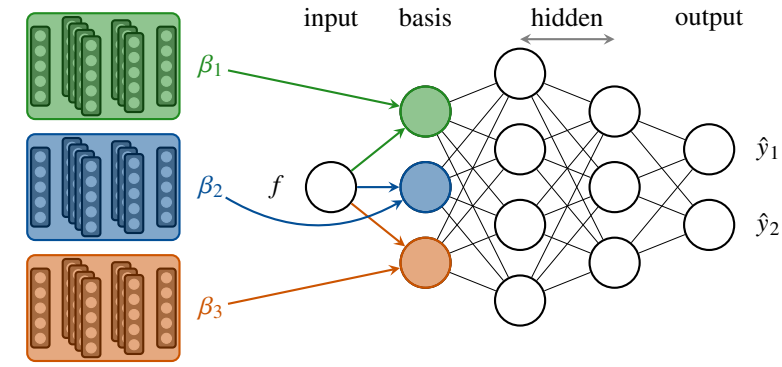
The conventional fully connected neural network processes scalar variables as inputs, thus some modifications are required to accommodate functional inputs, $f \in \mathcal{F}_I$, that are observed on a time grid t_p, $p = 1, \ldots, P$. This is achieved through the implementation of a *basis layer*, introduced by Yao et al. (2021), which serves as the initial hidden layer in the network. Let d_b denote the number of nodes in the basis layer. Each node x_b, $b = 1, \ldots, d_b$ is computed as a projection of the function f onto some basis function β_b as

$$x_b = \int_I \beta_b(t) f(t) dt. \tag{4.10}$$

These basis functions are estimated adaptively using 1D CNNs with filters of size one that take the observed time points t_p, $p = 1, \ldots, P$, and output the value $\beta(t_p)$, see Figure 4.6.

Note that the output signal of the nodes in the basis layer is a scalar, and therefore, the rest of the network after the basis layer is a standard fully connected neural network. Note also that the weight functions are a function of time t and not functions of individual f_i, thus are shared by all the observations. In the case of multivariate data $f = (f_1, \ldots, f_J)$, we propose estimating a separate basis layer for each of the dimensions $j = 1, \ldots, J$, concatenating these basis layers and together with additional tabular scalar covariates, $\boldsymbol{z}$, propagating through the subsequent NN. We define the concatenated layer $\boldsymbol{x}^{(1)}$ using

$$\boldsymbol{x}_j = \left(\int_I \beta_{1j}(s) f_j(s) ds, \ldots, \int_I \beta_{d_b j}(s) f_j(s) ds\right)^T \quad \text{and} \quad \boldsymbol{x}^{(1)} = \begin{pmatrix} \boldsymbol{x}_1 \\ \vdots \\ \boldsymbol{x}_J \\ \boldsymbol{z} \end{pmatrix}.$$



a) Networks in network *b) Functional NN with adaptive basis layer*

FIGURE 4.6. *Functional NN with adaptive basis layer. Figure a) shows the 1D CNNs that parameterize the weight functions β_i. Figure b) shows the neural network with a functional input and with a basis layer. The nodes in the basis layer are obtained as a projection on the weight functions using Equation* (4.10)*.*

4.4.1. *Instantaneous contribution to the total probability*

Neural networks are often referred to as "black box" models because their internal workings are not easily interpretable by humans. This arises from the complex structure consisting of numerous interconnected layers of units. Understanding how the predictions of the neural networks are made in critical applications like healthcare is essential. We introduce an explainability method for binary classification tasks based on the functional weights, and we call it the Instantaneous Contribution to the total Probability (ICP). It is defined as follows.

Let $f = (f_1, \ldots, f_J)$ be the multivariate functional input to the neural network defined on the interval $I = [t_1, t_2]$, let β_{bj} be the functional weights, let d_{bj} be the dimension of the basis layer for the j-th dimension of f, f_j, let $\boldsymbol{z}$ be the additional scalar covariates, let $\boldsymbol{w}$ be all the weights of the functional neural network including the neural networks that parametrize the β_{bj}, and let $\hat{y}$ be the predicted probability of the positive class.

Furthermore, we denote by η the functional neural network after the basis layer $\boldsymbol{x}^{(1)}$, and we denote by $\pi(t_2, f, \boldsymbol{z}; \boldsymbol{w})$ the *total probability*, which is defined as

$$\pi(t_2, f, \boldsymbol{z}; \boldsymbol{w}) \coloneqq \hat{y} = \boldsymbol{x}^{(L)} = \eta\left(\int_{t_1}^{t_2} \beta_{11}(s) f_1(s) ds, \ldots, \int_{t_1}^{t_2} \beta_{d_{bJ}J}(s) f_J(s) ds, \boldsymbol{z}; \boldsymbol{w}\right).$$

For $t \in I$, we define the *contribution to the total probability* as

$$\pi(t, f, \boldsymbol{z}; \boldsymbol{w}) = \eta\left(\int_{t_1}^{t} \beta_{11}(s) f_1(s) ds, \ldots, \int_{t_1}^{t} \beta_{d_{bJ}J}(s) f_J(s) ds, \boldsymbol{z}; \boldsymbol{w}\right).$$

The total probability can be decomposed as

$$\begin{aligned}\pi(t_2, f, z; \boldsymbol{w}) &= \pi(t_1, f, z; \boldsymbol{w}) + \int_{t_1}^{t_2} \pi'(t, f, z; \boldsymbol{w}) dt \\ &= \int_{t_1}^{t_2} \left(\frac{\pi(t_1, f, z; \boldsymbol{w})}{t_2 - t_1} + \pi'(t, f, z; \boldsymbol{w}) \right) dt,\end{aligned}$$

where the last integrand is the ICP. Comparing the cross-sectional means of the ICP of the two classes facilitates the understanding of important regions in the functional input for the classification problem.

In the context of ECG, the integration of this explainability technique significantly enhances model transparency. When a definitive correlation between a specific cardiac disease exists, ICP assists in validating the model by enabling the verification of whether the neural network's output is influenced by relevant regions in the ECG data. Conversely, in scenarios where no established correlation exists between the ECG and the cardiac condition under investigation, ICP can identify critical areas within the ECG measurements.

4.5. TIME-TO-EVENT OUTCOME

To accommodate time-to-event outcomes in continuous time in the presence of competing risks and censoring, we exploit the loss function defined as the negative log-likelihood in Equation (3.1)

$$\ell(\tilde{t}, \tilde{d}, \boldsymbol{x}; \boldsymbol{w}) = \sum_{i=1}^{N} \sum_{j=0}^{J} \left(\int_0^{\tilde{t}_i -} \hat{h}_j(t \mid \boldsymbol{x}_i; \boldsymbol{w}) dt - \mathbb{1}(\tilde{d}_i = j) \log(\hat{h}_j(\tilde{t}_i \mid \boldsymbol{x}_i; \boldsymbol{w})) \right),$$

where $\hat{h}_j$ is the estimated hazard of cause/censoring $j = 0, ..., J$, and $\boldsymbol{x}$ is the set of functional and scalar covariates. We estimate the hazard function in two ways, as follows.

4.5.1. *Weibull distribution*

The first method assumes the Weibull distribution of the event and censoring times. We use a fully connected neural network that takes covariates $\boldsymbol{x}$ as input and outputs the shape and scale parameters, k_j and λ_j, respectively. Let $\boldsymbol{s}^{(L)}$ be the input signal to the output layer of the neural network. The dimension is $d^{(L)} = 2(J + 1)$, since we use two nodes per cause and two nodes for censoring to represent the shape and scale of the underlying distributions. For clarity, we reshape the vector $\boldsymbol{s}^{(L)} \in \mathbb{R}^{2(J+1)}$ to a matrix $\boldsymbol{Y} \in \mathbb{R}^{2\times(J+1)}$, the first row corresponding to the shape parameters and the second row corresponding to the scale parameters.

However, we need to constrain the values of $\boldsymbol{Y}$ because both the scale and the shape have to be positive, $k_j, \lambda_j > 0$. We do this by using activation functions such as the softplus or the exponential. The output of the neural network is then

$$\eta(\boldsymbol{x}; \boldsymbol{w}) = \begin{pmatrix} k_0 & k_1 & \cdots & k_J \\ \lambda_0 & \lambda_1 & \cdots & \lambda_J \end{pmatrix},$$

where η represents the neural network. The cause- and censoring-specific hazards are calculated as

$$h_j(t \mid \boldsymbol{x}; \boldsymbol{w}) = \frac{k_j}{\lambda_j} \left(\frac{t}{\lambda_j} \right)^{k_j - 1}.$$

4.5.2. *Piecewise constant hazards*

The second method approximates the unknown hazard function by a piecewise constant hazard function. Given an ordered set $\mathcal{T} = \{t_1 = 0, t_2, \ldots, t_P \mid l < m \Leftrightarrow t_l < t_m\}$ of evaluation times, the input signal to the output layer of the neural network has the shape $\boldsymbol{s}^L \in \mathbb{R}^{P(J+1)}$. Again, for clarity, we reshape $\boldsymbol{s}^{(L)}$ to a matrix, but this time $\boldsymbol{Y} \in \mathbb{R}^{P \times (J+1)}$, each row p corresponding to a time point t_p and each column corresponding to one cause or censoring j.

Again, we have to constrain the values of $\boldsymbol{Y}$, because hazards are non-negative, thus we use an activation function with this property. The output of the neural network is then

$$\eta(\boldsymbol{x}; \boldsymbol{w}) = \begin{pmatrix} h_{1,0} & h_{1,1} & \cdots & h_{1,J} \\ h_{2,0} & h_{2,1} & \cdots & h_{2,J} \\ \vdots & \vdots & \ddots & \vdots \\ h_{P,0} & h_{P,1} & \cdots & h_{P,J} \end{pmatrix},$$

and the hazard rates can be retrieved for $t_p \leq t < t_{p+1}$ as

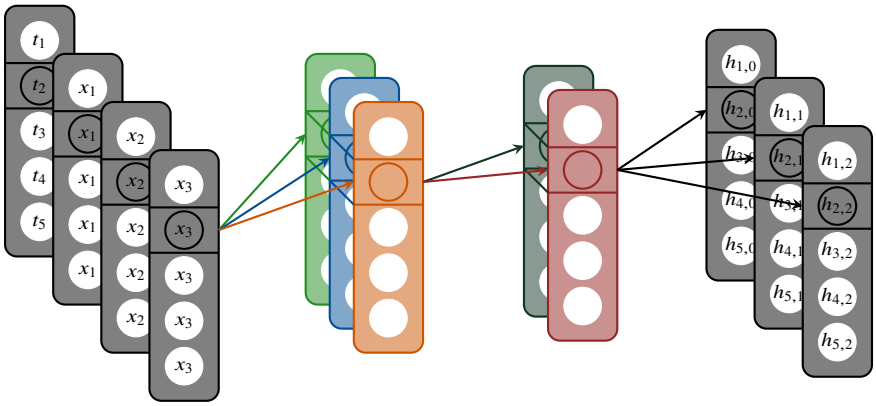
$$h_j(t \mid \boldsymbol{x}; \boldsymbol{w}) = h_{p,j}.$$

We use the 1D CNN with the filter of size one as the neural network model for this task, and we use the equivalence between this model and a fully connected neural network, see example in Figure 4.7.

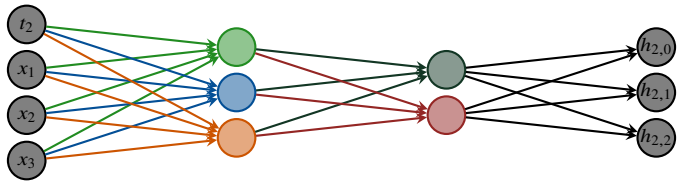
4.6. HYPERPARAMETER TUNING AND PERFORMANCE EVALUATION

The performance of neural networks is highly dependent on the correct setting of hyperparameters, which are the parameters defining the network architecture and the learning process, and they must be set before the model training. Proper tuning of hyperparameters, such as learning rate, number of layers, number of nodes, and activation functions, is crucial for achieving optimal model performance and generalization. Methods like *grid search*, *random search*, and *Bayesian optimization* are commonly used to systematically explore the hyperparameter space (Roy et al., 2023).

Grid search is a method for hyperparameter tuning that involves defining a set of values for each hyperparameter and then evaluating all possible combinations of these values to find the best set based on a performance metric. This exhaustive approach ensures that every potential combination is considered, but it can be very time-consuming and computationally expensive, especially with a large number of hyperparameters. In contrast, random search randomly samples hyperparameter combinations from specified distributions, which allows it to explore the hyperparameter space more broadly and often more efficiently. While random search may not guarantee that the optimal combination will



a) 1D CNN with a filter of size one



b) Fully connected neural network

FIGURE 4.7. *Example of a 1D CNN that parametrizes the hazard function. Figure a) shows the 1D CNN with a filter of size one, and Figure b) shows the equivalent fully connected neural network. In this example, we have 5 evaluation times* t_p*,* $p = 1, \ldots, 5$*, and 3 covariates* x_q*,* $q = 1, \ldots, 3$*. At each time point* t_p*, the neural network outputs the cause-specific hazards and the hazard of the censoring.*

be found, it typically requires fewer evaluations and can handle larger hyperparameter spaces more effectively. Both methods have their advantages and limitations, with grid search being more thorough but less efficient, and random search being more scalable but potentially less precise. Still, Bergstra and Bengio (2012) provide empirical and theoretical evidence showing that random search can outperform grid search, especially when the number of hyperparameters is large and only a few of them significantly impact the model's performance.

4.6.1. *Bayesian optimization*

Bayesian optimization offers significant advantages over grid search and random search in terms of efficiency and effectiveness in finding optimal hyperparameters, particularly in complex and high-dimensional spaces (Turner et al., 2021). It uses a *surrogate* model $\mathcal{M}$, typically a Gaussian process, to approximate the objective function, which is generally the loss function or other evaluation metric of the neural network and is usually "expensive" to evaluate directly. The surrogate function provides a prediction of the objective function's value for any given set of hyperparameters, along with an estimate of the uncertainty for that prediction.

We write the Gaussian process as

$$g(\boldsymbol{x}) \sim \mathcal{GP}(m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}')),$$

where $m\colon \mathcal{X} \to \mathbb{R}$ is the mean function, $k\colon \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the covariance function, and $\mathcal{X}$ is the set of possible inputs (Williams & Rasmussen, 2006, chap. 2). Consider a realization of g at $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathcal{X}$. Given a new point $\boldsymbol{x}$, the posterior of $g(\boldsymbol{x})$ given $\boldsymbol{\psi} = (g(\boldsymbol{x}_1), \ldots, g(\boldsymbol{x}_N))^T$ is

$$g(x) \mid \boldsymbol{\psi} \sim \mathcal{N}(\boldsymbol{k}^T \boldsymbol{K}^{-1}(\boldsymbol{\psi} - \boldsymbol{\mu}) + m(\boldsymbol{x}), k(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{k}^T \boldsymbol{K}^{-1} \boldsymbol{k}),$$

where $\boldsymbol{\mu} = (m(\boldsymbol{x}_1), \ldots, m(\boldsymbol{x}_N))^T$, $K_{ij} = k(\boldsymbol{x}_i, \boldsymbol{x}_j)$ and $\boldsymbol{k} = (k(\boldsymbol{x}, \boldsymbol{x}_1), \ldots, k(\boldsymbol{x}, \boldsymbol{x}_N))^T$ (H. Wang & Yang, 2023, chap. 3).

The next set of hyperparameters is selected using an *acquisition* function $\mathcal{A}$, that balances *exploration* (trying out hyperparameters in regions with high uncertainty) and *exploitation* (focusing on regions where the surrogate function predicts high performance). Common acquisition functions include Expected Improvement (EI), probability of improvement, and upper confidence (Snoek et al., 2012). These acquisition functions depend on the surrogate model estimated from the previous observations and hyperparameters of the Gaussian process, and this dependence is denoted by $\mathcal{A}(\boldsymbol{x}; \mathcal{M}, \boldsymbol{\theta})$. The expected improvement is defined as

$$\mathcal{A}_{\text{EI}}(\boldsymbol{x}; \mathcal{M}, \boldsymbol{\theta}) = [f^* - \mu(\boldsymbol{x}; \mathcal{M}, \boldsymbol{\theta})]\Phi(Z) + \sigma(\boldsymbol{x}; \mathcal{M}, \boldsymbol{\theta})\phi(Z)$$

for $Z = [f^* - \mu(\boldsymbol{x}; \mathcal{M}, \boldsymbol{\theta})]/\sigma(\boldsymbol{x}; \mathcal{M}, \boldsymbol{\theta})$, where f^* is the current best evaluated objective function, $\mu(\boldsymbol{x}; \mathcal{M}, \boldsymbol{\theta})$ and $\sigma^2(\boldsymbol{x}; \mathcal{M}, \boldsymbol{\theta})$ are the predictive mean function and the predictive variance function under the Gaussian process prior, respectively, $\Phi(\cdot)$ is the cumulative distribution function and $\phi(\cdot)$ is the density function of the standard normal distribution, respectively. The process of Bayesian optimization is described in Algorithm 6 and illustrated in Figure 4.8.

Algorithm 6 Bayesian optimization (adapted from H. Wang & Yang, 2023, alg. 10.1)

Input: Objective function f, surrogate model $\mathcal{M}$, acquisition function $\mathcal{A}$, parameters $\boldsymbol{\theta}$ of $\mathcal{A}$, search space $\mathcal{X}$, number of initial evaluation points N, stopping criterion

$X \leftarrow \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\} \subset \mathcal{X}$ ▷ Generate the initial evaluation points
$Y \leftarrow \{f(\boldsymbol{x}_1), \ldots, f(\boldsymbol{x}_N)\}$ ▷ Evaluate the objective function f at X
estimate surrogate model $\mathcal{M}$ on (X, Y)
while stopping criterion not satisfied **do**
    $\boldsymbol{x} \leftarrow \arg\max_{\boldsymbol{x}^* \in \mathcal{X}} \mathcal{A}(\boldsymbol{x}^*; \mathcal{M}, \boldsymbol{\theta})$ ▷ Find $\boldsymbol{x}$ that maximizes the acquisition function
    $X \leftarrow X \cup \{\boldsymbol{x}\}$ ▷ Add the optimal $\boldsymbol{x}$ to X
    $Y \leftarrow Y \cup \{f(\boldsymbol{x})\}$ ▷ Add the evaluated $f(\boldsymbol{x})$ to Y
    estimate surrogate model $\mathcal{M}$ on (X, Y)
end while

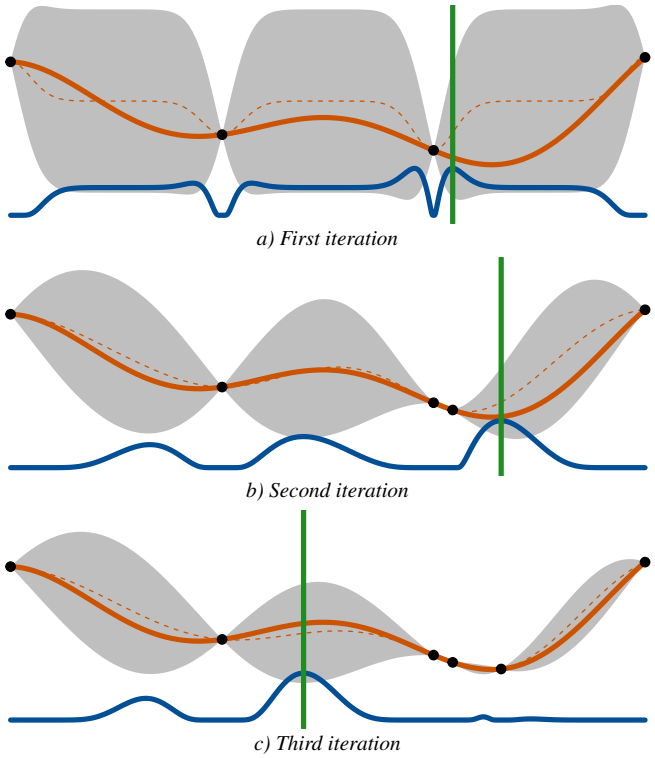


FIGURE 4.8. *First three iterations of Bayesian optimization for a univariate search space $\mathcal{X}$ as defined in Algorithm 6. The orange line is the objective function, with evaluated points in black, the dashed line is the surrogate function, with the 95% confidence interval (grey area). The blue line is the (scaled) acquisition function, and the vertical green line shows the point of the next evaluation.*

4.6.2. *Performance evaluation*

Deep neural networks are characterized by a large number of parameters, enabling them to learn complex patterns within the training data, including the noise inherent in the dataset. Consequently, while these models may exhibit outstanding performance on the training data, they often struggle to generalize to new, unseen data.

Evaluating the performance of neural networks using the training data is unreliable. The performance metrics derived from the training data tend to be overly optimistic. This is because the model has already been exposed to this data during training, leading to high accuracy and low error rates that do not accurately reflect the model's ability to generalize. These metrics can create a false sense of confidence in the model's performance, which can be misleading when the model is applied to new data.

To obtain a realistic estimate of a neural network's performance, it is crucial to use separate validation and test sets. The validation set is employed during the training process to fine-tune hyperparameters and make decisions regarding the model architecture. The test set, on the other hand, is used to evaluate the final model's performance after all training and tuning have been completed.

Cross-validation is often necessary to ensure a more robust and reliable evaluation of a neural network's performance. One of the primary reasons for this is the variability in the data. A single training-validation-test split can lead to performance metrics that are highly dependent on the specific partitioning of the data. Cross-validation averages the performance over multiple splits, providing a more comprehensive assessment of the model's generalization capabilities.

Another important reason for using cross-validation is to maximize the use of available data. In a single training-validation-test split, a portion of the data is reserved for testing, and is not used during training. This can be particularly problematic when the dataset is small. Cross-validation allows each data point to be used for both training and testing across different folds. This ensures that the model is trained and evaluated on all available data, leading to more reliable and stable performance metrics.

K-fold cross-validation. The K-fold cross-validation is a technique for assessing the model performance and optimizing the hyperparameters (Allgaier & Pryss, 2024). It involves partitioning the dataset into K equally sized subsets, that are often termed *folds*. The model is then trained on $K - 1$ folds, with the performance being estimated on the remaining fold. This is repeated K times in a *loop*, where each fold is used as the validation set exactly once. The performance metric is averaged over the validation folds.

If cross-validation is used for hyperparameter selection, then the hyperparameter combination with the best average performance across the folds is chosen. On the other hand, if cross-validation was used for performance evaluation, the average performance provides a more reliable estimate of the neural network's performance.

Nested cross-validation. Nested cross-validation is particularly useful when it comes to model selection and hyperparameter tuning (Hastie et al., 2009, sec. 7.10). The necessity of nested cross-validation arises from the need to obtain an unbiased estimate of the model's performance and to prevent overfitting during the hyperparameter optimization process.

When hyperparameters need to be tuned, the traditional cross-validation can lead to an overly optimistic estimate of the model's performance, because the same data is used both for hyperparameter tuning and evaluation of the model, causing information to "leak" from the validation set into the training process (Varma & Simon, 2006).

The process involves two loops. The outer loop serves for performance evaluation, while the inner loop is dedicated for hyperparameter tuning. In the outer loop, the dataset is divided into K folds, where again, one fold acts as a validation set while $K - 1$ folds are used for training. In each iteration of the outer loop, the $K - 1$ folds are split into J folds. The combination of hyperparameters that perform best on the inner J-fold cross-validation is selected, trained on the $K - 1$ folds, and evaluated on the validation fold. This process is repeated for each fold in the outer loop, and the performance metric is aggregated over the folds that were used for validation.

5. *Summary of manuscripts and contributions*

Manuscript I. The manuscript titled "Joint alignment of multivariate quasi-periodic functional data using deep learning" introduces a novel method for aligning multivariate quasi-periodic functions using deep neural networks. This method, named DeepJAM, addresses the limitations of traditional techniques that often overlook phase variability and focus solely on amplitude variability. By preserving both phase and amplitude variability, DeepJAM provides a comprehensive alignment of multivariate functional data. The method employs a special activation function based on the unit simplex transformation and utilizes a loss function derived from the Fisher-Rao metric. It is unsupervised and capable of generating both a common template function and subject-specific templates. The effectiveness of DeepJAM is demonstrated through simulations and data from 12-lead ECG recordings.

A key contribution of the manuscript is the introduction of a multiscale warping model to handle quasi-periodic functions. This model allows for the decomposition of variability into local and global components, facilitating more accurate alignment of multivariate quasi-periodic data. The method's applicability to real-world data and its potential for future enhancements make it a valuable contribution to the field of functional data analysis.

Manuscript II. The manuscript titled "Deep learning for multivariate functional data with built-in time warping" exercises an application of neural networks for analyzing multivariate quasi-periodic functional data, with a specific emphasis on 12-lead ECGs. We build on a neural network with an adaptive basis layer for univariate data and extend the capabilities of this approach to handle multivariate functional data alongside additional scalar variables.

We introduce the instantaneous contribution to the total probability (ICP), a novel method aimed at improving the interpretability of the neural network model's predictions. It identifies key segments of the input signal that have an impact on the predictions, offering valuable insights into the data-generating mechanisms. This interpretability is especially important in clinical settings, where understanding the factors influencing model predictions can build trust in the model's output.

Additionally, we address the challenge of handling input data with varying length and varying number of periods, a common issue in ECG recordings stemming from heart rate variability among patients.

Manuscript III. The manuscript titled "Deep learning of event risk in continuous time with competing risks" presents an innovative approach for predicting time-to-event outcomes in continuous time with competing risks using neural networks. Survival analysis, which focuses on understanding the time until a specific event occurs, is extended in this

work to incorporate high-dimensional data and competing risks.

The proposed method introduces two types of neural network models. The first type assumes that the underlying generating processes follow a Weibull distribution and outputs the shape and scale parameters of this distribution for each event type and censoring. The model leverages the flexibility of neural networks to capture complex relationships between the covariates and the parameters of the Weibull distribution, allowing for accurate prediction of time-to-event outcomes.

The second type of model directly parameterizes the hazard functions as piecewise constant over predefined time intervals. This model does not assume a specific distribution for the underlying processes. Instead, it uses a convolutional neural network to estimate the hazard rates for each event type and censoring at different time points, allowing for the estimation of the cause-specific risks.

A significant contribution of this work is its ability to simultaneously predict the risk of the event of interest, competing risks, and the probability of censoring, while also incorporating high-dimensional data.

6. *Discussion and perspectives*

The primary objective of this thesis was to develop statistical methods integrating deep learning and functional data analysis for predicting various clinical outcomes based on ECG signals. The ultimate goal was to create models that clinicians could use for early diagnosis of heart diseases, leveraging the rich data available from the Danish national health registers.

Achievements and limitations. The proposed methods, including the algorithm for joint alignment of multivariate quasi-periodic data and the neural network for multivariate functional data as input with adaptive basis layers, showed promising results in reducing variability caused by misalignment and improving prediction accuracy.

The application of these methods to 12-lead ECG recordings from the Copenhagen General Population Study highlighted their effectiveness in handling high-dimensional functional data. The introduction of the instantaneous contribution to the total probability provided insight into the model's predictions, enhancing interpretability and transparency.

However, the scope of the ECG analysis was limited to a dichotomized coronary calcification score due to the challenges in accessing and processing the clinical outcomes from the Danish patient registry. The inability to perform a more extensive analysis of various clinical outcomes related to heart diseases represents a limitation and opportunity for future work.

Future directions. Future research should focus on addressing the computational challenges and expanding the analysis to include a broader range of clinical outcomes. The following areas are suggested for further exploration.

Utilizing the extensive data available in the Danish health registers will be crucial. This will enable a more detailed analysis of short- and long-term risks of various cardiovascular diseases, providing clinicians with valuable tools for early diagnosis and interventions.

Implementing transfer learning techniques to retrain the models for different clinical outcomes can significantly enhance their applicability and reduce computational costs (J. Wang & Chen, 2023). This approach allows the models to be adapted to new datasets and outcomes, improving their generalizability and utility in clinical practice.

Extending the survival analysis methods developed in this thesis to ECG data will provide a more comprehensive understanding of the prognostic value of ECG measurements. This integration requires addressing the computational challenges associated with training the neural networks, especially the convolutional neural networks, on the research machines of Statistics Denmark.[1]

[1] Statistics Denmark, *Homepage of Statistics Denmark* [website], https://www.dst.dk/en/, accessed November 30, 2024

Validating the proposed methods on diverse datasets from different populations and clinical settings will be essential to establish their robustness and generalizability. This will also help identify potential biases and ensure that the models are applicable across different demographic groups.

Perspectives. The advancements in deep learning and functional data analysis presented in this thesis have important implications for future clinical diagnostics and personalized medicine. By leveraging ECG data and advanced statistical methods, clinicians can gain deeper insights into the health status of patients and make more informed decisions regarding their care.

The integration of these methods into clinical workflows has the potential to transform the early diagnosis and management of heart diseases. As computational resources continue to improve, the application of these techniques will likely expand, offering new opportunities for enhancing patient outcomes.

In conclusion, while this thesis laid the groundwork for using ECG data to predict clinical outcomes with deep learning, there is still much work to be done. The proposed future directions provide a roadmap for further research and development, with the ultimate goal of creating robust, adaptable, and interpretable models that can be integrated into clinical practice.

Bibliography

ABU-MOSTAFA, Y., MAGDON-ISMAIL, M., & LIN, H.-T. (2012, March). E-chapter 7: Neural networks. In *Learning from data*. https://amlbook.com/eChapters.html

ADIGUN, A. A., & YINKA-BANJO, C. (2022). Comparing stochastic gradient descent and mini-batch gradient descent algorithms in loan risk assessment. In S. MISRA, J. OLURANTI, R. DAMAŠEVIČIUS, & R. MASKELIUNAS (Eds.), *Informatics and intelligent applications* (pp. 283–296, Vol. 1547). Springer International Publishing. https://doi.org/10.1007/978-3-030-95630-1_20

ALLGAIER, J., & PRYSS, R. (2024). Cross-validation visualized: A narrative guide to advanced methods. *Machine Learning and Knowledge Extraction*, *6*(2), 1378–1388. https://doi.org/10.3390/make6020065

ANDERSEN, P. K., ABILDSTROM, S. Z., & ROSTHØJ, S. (2002). Competing risks as a multi-state model. *Statistical Methods in Medical Research*, *11*(2), 203–215. https://doi.org/10.1191/0962280202sm281ra

ANDERSEN, P. K., & KEIDING, N. (2002). Multi-state models for event history analysis. *Statistical Methods in Medical Research*, *11*(2), 91–115. https://doi.org/10.1191/0962280202SM276ra

BAKER, R. (2006). Gait analysis methods in rehabilitation. *Journal of NeuroEngineering and Rehabilitation*, *3*(1), 4. https://doi.org/10.1186/1743-0003-3-4

BERGSTRA, J., & BENGIO, Y. (2012). Random search for hyper-parameter optimization. *Journal of machine learning research*, *13*(2).

BHATTACHARYA, A., & BHATTACHARYA, R. (2012, April 5). *Nonparametric inference on manifolds: With applications to shape spaces* (1st). Cambridge University Press. https://doi.org/10.1017/CBO9781139094764

BILLMAN, G. E., HUIKURI, H. V., SACHA, J., & TRIMMEL, K. (2015). An introduction to heart rate variability: Methodological considerations and clinical applications. *Frontiers in Physiology*, *6*. https://doi.org/10.3389/fphys.2015.00055

CHEN, W. (2018). Electrocardiogram. In T. TAMURA & W. CHEN (Eds.), *Seamless healthcare monitoring* (pp. 3–44). Springer International Publishing. https://doi.org/10.1007/978-3-319-69362-0_1

COX, D. R. (1972). Regression models and life-tables. *Journal of the Royal Statistical Society. Series B (Methodological)*, *34*(2), 187–220.

GOLDBERGER, A. L., GOLDBERGER, Z. D., & SHVILKIN, A. (2018). *Goldberger's clinical electrocardiography*. Elsevier. https://doi.org/10.1016/C2014-0-03319-9

GOODFELLOW, I., BENGIO, Y., & COURVILLE, A. (2016). *Deep learning*. The MIT Press.

HASTIE, T., TIBSHIRANI, R., & FRIEDMAN, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer Science & Business Media.

HOSSEINI-NASAB, S. M. E., & SHARGHI, H. (2024). Functional data analysis: Key concepts and applications. In H. DOOSTI (Ed.), *Flexible nonparametric curve estimation* (pp. 43–80). Springer International Publishing. https://doi.org/10.1007/978-3-031-66501-1_3

KAPLAN, E. L., & MEIER, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, *53*(282), 457–481. https://doi.org/10.1080/01621459.1958.10501452

MAKOWSKI, D., PHAM, T., LAU, Z. J., BRAMMER, J. C., LESPINASSE, F., PHAM, H., SCHÖLZEL, C., & CHEN, S. H. A. (2021). NeuroKit2: A python toolbox for neurophysiological signal pro-

cessing. *Behavior Research Methods*, *53*(4), 1689–1696. https://doi.org/10.3758/s13428-020-01516-y

MICROSOFT. (2024, October). *Copilot*. https://copilot.microsoft.com

ROY, S., MEHERA, R., PAL, R. K., & BANDYOPADHYAY, S. K. (2023). Hyperparameter optimization for deep neural network models: A comprehensive study on methods and techniques. *Innovations in Systems and Software Engineering*. https://doi.org/10.1007/s11334-023-00540-3

SNOEK, J., LAROCHELLE, H., & ADAMS, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Proceedings of the Advances in Neural Information Processing Systems*, *25*.

SRIVASTAVA, A., & KLASSEN, E. P. (2016). *Functional and shape data analysis* (1st). Springer New York. https://doi.org/10.1007/978-1-4939-4020-2

STAN DEVELOPMENT TEAM. (2024). Unit simplex. *Stan Modeling Language Users Guide and Reference Manual, 2.35*. https://mc-stan.org/docs/reference-manual/transforms.html#simplex-transform.section

TURNER, R., ERIKSSON, D., MCCOURT, M., KIILI, J., LAAKSONEN, E., XU, Z., & GUYON, I. (2021). Bayesian optimization is superior to random search for machine learning hyperparameter tuning: Analysis of the black-box optimization challenge 2020. *NeurIPS 2020 Competition and Demonstration Track*, 3–26.

VARMA, S., & SIMON, R. (2006). Bias in error estimation when using cross-validation for model selection. *BMC Bioinformatics*, *7*(1), 91. https://doi.org/10.1186/1471-2105-7-91

WANG, H., & YANG, K. (2023). Bayesian optimization. In D. BROCKHOFF, M. EMMERICH, B. NAUJOKS, & R. PURSHOUSE (Eds.), *Many-criteria optimization and decision analysis* (pp. 271–297). Springer International Publishing. https://doi.org/10.1007/978-3-031-25263-1_10

WANG, J., & CHEN, Y. (2023). *Introduction to transfer learning: Algorithms and practice*. Springer Nature Singapore. https://doi.org/10.1007/978-981-19-7584-4

WEIBULL, W. (1951). A statistical distribution function of wide applicability. *Journal of applied mechanics*.

WILLIAMS, C. K., & RASMUSSEN, C. E. (2006). *Gaussian processes for machine learning* (Vol. 2). MIT press Cambridge, MA.

YAO, J., MUELLER, J., & WANG, J.-L. (2021). Deep learning for functional data analysis with adaptive basis layers. *Proceedings of the 38th International Conference on Machine Learning*, *139*, 11898–11908. http://proceedings.mlr.press/v139/yao21c/yao21c.pdf

Manuscripts

Manuscript I

JOINT ALIGNMENT OF MULTIVARIATE QUASI-PERIODIC FUNCTIONAL DATA USING DEEP LEARNING

VI THANH PHAM
Section of Biostatistics, University of Copenhagen, Denmark

JONAS BILLE NIELSEN
Department of Cardiology and Radiology, Copenhagen University Hospital, Denmark

KLAUS FUGLSANG KOFOED
Department of Cardiology and Radiology, Copenhagen University Hospital, Denmark
and
Department of Clinical Medicine, University of Copenhagen, Denmark

JØRGEN TOBIAS KÜHL
Department of Cardiology, Zealand University Hospital, Denmark

ANDREAS KRYGER JENSEN
Section of Biostatistics, University of Copenhagen, Denmark

ABSTRACT

The joint alignment of multivariate functional data plays an important role in various fields such as signal processing, neuroscience and medicine, including the statistical analysis of data from wearable devices. Traditional methods often ignore the phase variability and instead focus on the variability in the observed amplitude. We present a novel method for joint alignment of multivariate quasi-periodic functions using deep neural networks, decomposing, but retaining all the information in the data by preserving both phase and amplitude variability. Our proposed neural network uses a special activation of the output that builds on the unit simplex transformation, and we utilize a loss function based on the Fisher-Rao metric to train our model. Furthermore, our method is unsupervised and can provide an optimal common template function as well as subject-specific templates. We demonstrate our method on two simulated datasets and one real example, comprising data from 12-lead 10s electrocardiogram recordings.

Keywords: Deep Learning; Elastic Phase-Amplitude Separation; Functional Data; Joint Multivariate Alignment; Multiscale Time Warping; Quasi-Periodic Functions.

I.1. INTRODUCTION

Statistical analysis of functional data is becoming increasingly essential in the biological and medical research as technologies allow for measuring subjects over long time durations with potential high-frequency sampling times. When applying statistical inference to functional data, it would be advantageous to align the functions, so that the positions of corresponding peaks and valleys are the same across the data set. Failure to align the data correctly can lead to inefficiency of basic statistical summaries like averages, leading to poor estimates of population mean functions that are not representative of the data. In reality, many commonly used functional data analysis techniques present inferior performance when confronted with phase variation (Marron et al., 2015). Furthermore, alignment can also improve prediction accuracy when classification is the goal (Tucker et al., 2013). In practice, functional data is often misaligned and contains variability along both the x axis, called the *phase* variability, and the y axis, called the *amplitude* variability. The process of decomposing the overall variability in the data into these two components is termed elastic phase-amplitude separation and represents the joint alignment of multiple functions. In this paper, we focus on multivariate quasi-periodic functions. We call a function f multivariate if $f(t) = (f_1(t), f_2(t), \ldots, f_J(t)) \in \mathbb{R}^J$, and following Boucheham (2008), we call a function quasi-periodic if it is a concatenation of similar patterns or pseudo-periods, or in other words, if it is periodic up to a warping action.

Examples of such quasi-periodic functional data are continuous glucose monitoring (CGM) data (Klonoff, 2005), which provide information about levels of blood glucose through wearable devices (McDonnell et al., 2022), and electrocardiogram (ECG) recordings (Gregg et al., 2008), which measure the electrical activity of the heart. We aim to study ECGs as they are regularly used together with other tests to diagnose and monitor diseases affecting the cardiovascular system, as well as to inspect symptoms of possible heart conditions. ECGs can assist with detecting arrhythmias and can be used over time to monitor a person with an existing diagnosis, or a person taking medication that can have an influence on the heart. These measurements taken over time through multiple leads placed on the body can be regarded as multivariate quasi-periodic functions, where the periods are the heartbeats. We think of the underlying function or process as inherently periodic, but due to phase and amplitude variability resulting from other physiological activities at the moment, the observed data is quasi-periodic.

One can perform statistical analyses to align and compare these observations using tools such as the L^2 distance to find the cross-sectional mean and variance. In addition to these summary measures, it is of great interest to model the variability in the data. Due to the high dimensionality of the data and to attain flexible models, we would like to analyze them using neural networks. In the example of the ECG data, we might want to align the peaks and segments to extract a template heartbeat while also recording the relative positions of these features. However, application of standard methods for the analysis of functional data is not straightforward when it comes to quasi-periodic functions as we illustrate in Figure I.1. The dashed lines in Figure a) represent equal period lengths, while the dotted lines represent the observed quasi-periods. In general, we do not know the positions of the quasi-periods of the observed data, and when assuming periodic data, the cross-sectional mean in Figure b) is not representative of the underlying process – the magnitude of the peaks and valleys are attenuated. On the other hand, when first

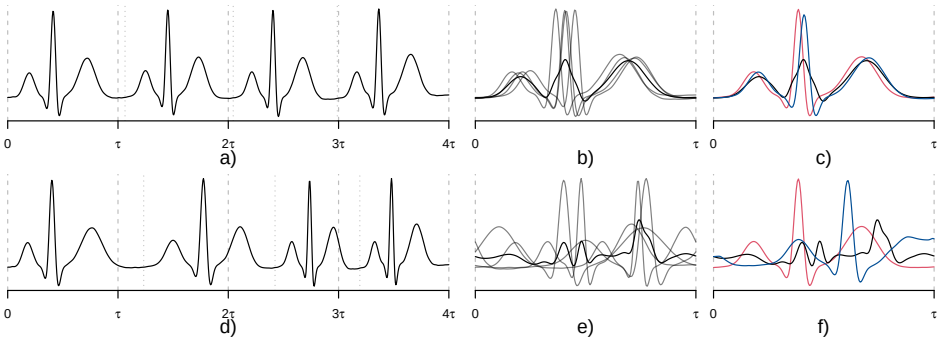


FIGURE I.1. *Figure a) shows four quasi-periods of one observed ECG lead with a length of one period being τ. The dashed lines represent periods of length τ, while the dotted lines are the observed quasi-periods. Figure b) shows the cross-sectional mean (black) of the four periods (gray), when they are treated as periodic, and Figure c) compares this cross-sectional mean to a cross-sectional mean of aligned data (red), taking into consideration the quasi-periodic nature of the observed data, as well as the standard Karcher mean using dynamic programming (blue), which does not take the quasi-periodicity of the observed data into account. The second row shows the same as the first row, but for a case of more extreme phase variability.*

aligning the data while considering the quasi-periodic nature of the observed functional data, the cross-sectional mean preserves the maxima and minima better, see Figure c). In the example of Figure I.1 (first row), the phase-variability is not substantial, hence the cross-sectional mean of the observed periods still maintains the main segments of the ECGs. Using alignment based on the elastic shape-amplitude separation algorithm utilizing dynamic programming (Srivastava & Klassen, 2016, chap. 8), the Karcher mean also preserves the true shape, but in extreme cases when the data contains considerable phase variability, the simple cross-sectional mean does not retain these key ECG segments (modality), and even when using the elastic shape-amplitude separation algorithm, the Karcher mean is distorted, see the second row of Figure I.1.

In this paper, to jointly (simultaneously) model multiple time warping functions and to align multivariate quasi-periodic functions along the x axis, we modify and extend the elastic phase-amplitude separation algorithm for univariate functions from Srivastava and Klassen (2016) and introduce the algorithm for the Joint Alignment of Multivariate quasi-periodic functional data using Deep learning (DeepJAM). Importantly, DeepJAM provides us with warping functions, one for each observed function, that contain all the information about the phase variability in the data. In the case of multivariate functional data, all the dimensions share the same warping function, because they are all observed on the same sample. Because our approach is unsupervised, we also utilize nice geometric properties of transformed warping functions to extract a unique multivariate functional template. Subsequently, we can use the DeepJAM neural network to easily align new data to this template. Furthermore, DeepJAM can be used as an integrated part of other neural networks in an end-to-end analysis as opposed to two-step modeling, where alignment is only treated as a pre-processing step.

I.1.1. *Related work*

A standard approach for alignment of functions is called landmark registration (Ramsay & Silverman, 2005, chap. 7). Landmarks are typically distinctive features of a function, such as minima, maxima, and zero-crossings of the function itself, or its derivatives. The challenge with landmark registration is that it only focuses on the alignment of specific points. An extension to the landmark registration is template registration (Srivastava & Klassen, 2016, chap. 8.2), which aligns whole functions to a specific template. Kneip and Ramsay (2008) suggest aligning functions to the functional principal components instead, with the idea that aligned functions can be represented as the sum of a mean function and a linear combination of a few principal components. This is done through an iterative process of simultaneous estimation of the mean function, warping functions, functional principal components and their scores. An alternative approach also based on functional principal component analysis (FPCA) is presented in Srivastava and Klassen (2016, chap. 8.8).

Dynamic Time Warping (DTW, Müller, 2007, chap. 4) is another well-known technique for finding an optimal alignment between two time series. Building on DTW, Boulnemour and Boucheham (2018) proposed a combination with the shape exchange algorithm, which should be more suitable for handling quasi-periodic time series. However, in applications, we often need joint alignment of multiple functions, because we are interested in patterns regarding the whole data, such as population mean and variance. This could be done, for example, by finding a representative sample in the data, or by constructing a representative template and then using DTW to align the functions to this template. Nonetheless, DTW does not provide us with this template. Srivastava and Klassen (2016) proposed a method to solve this problem using dynamic programming. Regardless of whether the objective is to achieve pairwise or group-wise alignment, the purpose of the above methods using dynamic programming is mainly to serve as a pre-processing step.

In the deep learning universe, the main focus has been on time warping invariant neural networks (Sun et al., 1992), convolutional neural networks (LeCun et al., 2015) that can be invariant to shifts in the input data, and recurrent neural networks (Tallec & Ollivier, 2018). The challenge with using neural networks that are invariant to time warping or shifts is that they mostly focus on amplitude variability. This may or may not be relevant depending on the specific task. Disregarding the phase variability could result in loss of information and inadequate generative models, while using models for both amplitude and phase can improve classification of future data (Tucker et al., 2013).

Recently, research has also been done on pairwise temporal alignment using neural networks. Nunez and Joshi (2020) proposed a convolutional neural network for the alignment of all pairs of functions in the data, which outputs estimated warping functions. However, the training data in this particular setting were constructed using DTW, the optimal warping functions from DWT were used in the loss function, and as the authors themselves expressed, their primary motivation was demonstrating reduced computational cost compared to DTW. In addition, the neural network was constructed to deal with pairwise alignment of all functional pairs, but not for joint alignment of multiple functions.

Oh et al. (2018) proposed an end-to-end classification model using a Sequence Transformer Network (STN), which transforms the input signals with parameters learned through a Sequence Transformer convolutional neural network, before proceeding with another neural network for classification. However, this STN focuses on linear transfor-

mation and does not implement more complicated time warping of the input signal.

In addition to unsupervised single-class learning, Weber et al. (2019) explored the multi-class case with semi-supervised learning. The implemented model is trained using a loss function that involves the empirical variance of the warped signals and uses a regularization term on the warping as a way to ensure identifiability.

Currently, more and more deep learning methods are developed to align multivariate functions using elastic phase-amplitude separation algorithm; see e.g., the works of Lohit et al. (2019), Nunez et al. (2021) and Chen and Srivastava (2021), all of which use a probabilistic simplex transformation for the derivative of the warping functions by calculating the elementwise product of the neural network output and dividing it by the square of its norm. This transformation is, however, not one-to-one.

I.1.2. *Proposed approach*

In this paper, we propose a non-parametric approach for joint alignment of multivariate quasi-periodic functions. We deploy neural networks that aim to extract the optimal warping functions that achieve the smallest distance between the warped functions obtained from the observed functions by warping their domain. We use a special one-to-one activation function in the output layer, so that the output of the neural network satisfies the conditions for warping functions. In addition, instead of using the L^2 metric to calculate the distance between functions directly, we use the Fisher-Rao metric and the square-root slope representation to calculate the distance between the warped functions. This metric is used in the loss function of the neural network. Furthermore, because we take advantage of the differential geometry of the space of warping functions, we can calculate the Karcher mean of the orbits to extract a functional template.

We make use of convolutional layers with multiple channels to account for multivariate functions, and as a result, the output layer returns a single warping function per subject that can be used for warping all the dimensions of the observed function simultaneously, which is desirable as they are all measured on the same subject. Finally, we incorporate a multiscale warping model to handle quasi-periodic functions.

This paper is structured as follows: In Section I.2, we present and review the mathematical formalism behind joint alignment of functional data, which we proceed to extend in Section I.2.1 to the case of multivariate quasi-periodic functional data. In the same section, we present the architecture of the deep neural warping network employed in the algorithm. In Section I.3, we apply our method to simulated univariate and multivariate functions. In Section I.4, we carry out the joint alignment on real ECG data, and finally in Section I.5, we discuss the results and limitations. An implementation of our method can be found on the first author's GitHub repository.

I.2. JOINT ALIGNMENT OF MULTIVARIATE FUNCTIONAL DATA

In this section, we present the mathematical background for joint alignment of a set of functions $\{f_i\} = \{f_i\}_{i=1}^n = \{f_i \in \mathcal{F}_I \mid i = 1, \ldots, n\}$, where $\mathcal{F}_I$ is the set of absolutely continuous functions defined on the interval I, which we without loss of generality take to be $I = [0, 1]$. Further, we consider multivariate functional data $f_i = (f_{i1}, \ldots, f_{iJ}) : I \to \mathbb{R}^J$.

To jointly align a set of functions, we need to warp the domain of the functions in a certain constrained way. The constraints for the mapping $\gamma\colon\ I \to I$ performing the domain warp are that $\gamma(0) = 0$ and $\gamma(1) = 1$ (*boundary-preserving*), that γ is invertible, and that both γ and γ^{-1} are differentiable (*diffeomorphism*). We denote the set of such functions Γ_I. Note that the derivative of γ, denoted as $\dot{\gamma}$, is always positive.

With the definition of the two sets, $\mathcal{F}_I$ and Γ_I, we can describe the two main alignment problems: pairwise and multiple alignment (Srivastava & Klassen, 2016, p. 85). In the pairwise alignment problem, given functions $f_1, f_2 \in \mathcal{F}_I$, we wish to find a warping function $\gamma \in \Gamma_I$, such that some energy term $E[f_1, f_2 \circ \gamma]$, where the symbol "$\circ$" represents function composition, i.e., $(f \circ \gamma)(t) = f(\gamma(t))$, is minimized. In the case of multivariate functions, i.e., $J \geq 2$, we calculate the function composition elementwise, namely, $f \circ \gamma = (f_1 \circ \gamma, \ldots, f_J \circ \gamma)$. The operations defined below are also elementwise, where relevant. We wish to find γ^* as a solution to $\gamma^* = \operatorname{argmin}_{\gamma \in \Gamma_I} E[f_1, f_2 \circ \gamma]$. The function f_1 is then said to be registered to $f_2 \circ \gamma^*$ for any domain value $t \in I$. The multiple alignment problem is an extension of the pairwise alignment problem, where, given a set of functions $\{f_i\}$, we wish to find a set of warping functions $\{\gamma_i\} = \{\gamma_i\}_{i=1}^n = \{\gamma_i \in \Gamma_I \mid i = 1, \ldots, n\}$, such that $f_i \circ \gamma_i$ are said to be registered to each other over i. The $\{\gamma_i\}$ are called the *phases* and $\{f_i \circ \gamma_i\}$ are representatives of their *amplitude*. Finally, the pairwise solution of the multiple alignment, in which all pairs of functions are registered to each other, can be extended to a template-based registration. Here, we first consider a template function μ, and then we align each of the functions $\{f_i\}$ to this template. These steps can be done iteratively to improve the overall alignment as well as the quality of the template (Srivastava & Klassen, 2016, p. 271). First, we calculate the average of the current versions of $\{f_i\}$ under a proper metric to construct the template μ. Then we align the functions $\{f_i\}$ to this template by calculating the optimal warping functions $\{\gamma_i\}$, and update the functions $\{f_i\}$ by $f_i \leftarrow f_i \circ \gamma_i$, and then we iterate these steps.

A natural way to calculate the average of $\{f_i\}$ for constructing the template would be to use the L^2 norm, yielding the cross-sectional mean of $\{f_i\}$. However, as argued by Srivastava and Klassen (2016, chap. 8.2), the cross-sectional mean of $\{f_i\}$ is not a good representation of the template μ, and in addition, also shown by Srivastava and Klassen (2016, pp. 88–90), the standard L^2 norm is not appropriate as the distance measure due to the lack of isometry under warping, pinching effect, and inverse inconsistency. Instead, we use a specific Riemannian metric, called the Fisher-Rao metric (Srivastava & Klassen, 2016, Definition 4.8., p. 105), together with an alternative functional representation called the square-root slope function representation (SRSF, Srivastava & Klassen, 2016, Definition 4.2., p. 91) defined as

$$q(t) = \operatorname{sign}\left(\dot{f}(t)\right)\sqrt{\left|\dot{f}(t)\right|}. \tag{I.1}$$

from which the original function can be recovered as

$$f(t) = f(0) + \int_0^t q(s)|q(s)|\,ds. \tag{I.2}$$

Under this representation, the Fisher-Rao metric becomes the standard L^2 metric (Srivastava & Klassen, 2016, Lemma 4.7., p. 105). Note also that the SRSF representation of a

warped function $f \circ \gamma$ is (Srivastava & Klassen, 2016, p. 91)

$$\tilde{q}(t) = (q \circ \gamma)(t)\sqrt{\dot{\gamma}(t)} =: (q, \gamma)(t). \tag{I.3}$$

Srivastava and Klassen (2016, chap. 8) present an alternative estimator of μ which uses the notion of the amplitude of a function in the SRSF space. Given a function $f \in \mathcal{F}_I$ and its associated SRSF representation, q in Equation (I.1), the amplitude of f in the SRSF space is defined by the orbit, which is the set of all possible domain transformations according to the group action

$$[q] = \text{closure}\left\{(q, \gamma) = (q \circ \gamma)\sqrt{\dot{\gamma}} \;\middle|\; \gamma \in \Gamma_I\right\}.$$

With the definition of the amplitude, Srivastava and Klassen (2016) show that the Karcher mean of the set of orbits $\{[q_i]\}$, as described later in this section, is an estimator of the orbit $[\mu_q]$ and that a specific element of this orbit can be used as an estimator of μ_q, which is the SRSF representation of μ. The template-based alignment problem is then reduced to finding the set $\{\gamma_i\}$ that best aligns the functions $\{q_i\}$ to the template μ_q, which can be formally written as

$$\gamma_i = \underset{\gamma \in \Gamma_I}{\text{arginf}} \|\mu_q - (q_i, \gamma)\|. \tag{I.4}$$

In order to estimate μ_q, we first have to calculate the Karcher mean of the set of orbits $\{[q_i]\}$ that is defined as (Srivastava & Klassen, 2016, Definition 8.1., p. 274)

$$[\mu_q] = \text{arginf}_{[q] \in \mathcal{A}} \sum_{i=1}^{n} \inf_{\gamma \in \Gamma_I} (\|q - (q_i, \gamma)\|)^2,$$

where $\mathcal{A}$ is a quotient space $\mathcal{F}_I / \tilde{\Gamma}_I$ and $\tilde{\Gamma}_I$ is a set of boundary preserving weakly increasing absolutely continuous functions $\gamma \colon I \to I$.

The Karcher mean of the amplitudes is again an orbit, and we have to find a particular element of this orbit, specifically its center with respect to the set $\{q_i\}$ (Srivastava & Klassen, 2016, Definition 8.2., p. 275). Such an element μ_q satisfies the property that the Karcher mean of the warping functions $\{\gamma_i\}$, which are the solutions to Equation (I.4), is the identity warp, $\gamma_{\text{id}}(t) = t$.

The algorithm for finding a center of an orbit with respect to the set $\{q_i\}$ follows Srivastava and Klassen (2016, Algorithm 33., p. 277) with some modifications. In the first step of the algorithm, we select an element $\tilde{\mu}_q$ of the orbit $[\mu_q]$, i.e., the cross-sectional mean of $\{q_i\}$, and we find $\{\tilde{\gamma}_i\}$ by solving $\tilde{\gamma}_i = \text{arginf}_{\gamma \in \Gamma_I}(\|\tilde{\mu}_q - (q_i, \gamma)\|)$. As opposed to using dynamic programming for this problem as in Srivastava and Klassen (2016), we propose to use a convolutional neural network. A second step of the algorithm is calculating the Karcher mean $\mu_{\tilde{\gamma}}$ of the phases $\{\tilde{\gamma}_i\}$ and finding the center of the orbit $[\mu_q]$ with respect to the set $\{q_i\}$ by

$$\mu_q = (\tilde{\mu}_q, \mu_{\tilde{\gamma}}^{-1}). \tag{I.5}$$

Srivastava and Klassen (2016) show that for each q_i,

$$\gamma_i = \tilde{\gamma}_i \circ \mu_{\tilde{\gamma}}^{-1} \tag{I.6}$$

minimizes $\|\mu_q - (q_i, \gamma)\|$, hence these $\{\gamma_i\}$ are a solution to the joint template-based registration problem represented by Equation (I.4). The center, μ_q, of the orbit is the estimator of the SRSF of the template function μ.

Karcher mean of warping functions.. To define the Karcher mean of a set of warping functions γ_i under the Fisher-Rao metric, we will use the differential geometry of Γ_I. Direct analysis on Γ_I is not straightforward due to it being a non-linear manifold, thus we will work with the SRSF representation of γ_i. The SRSF representation of any $\gamma \in \Gamma_I$ has the form $\psi = \sqrt{\dot{\gamma}}$, which is equivalent to Equation (I.1) because $\dot{\gamma} > 0$ for all domain values $t \in I$. An advantage of using this representation is that $\|\psi\|^2 = \int_0^1 \psi^2(t)\, dt = \int_0^1 \dot{\gamma}(t)\, dt = \gamma(1) - \gamma(0) = 1$, and thus ψ lies in the positive orthant of the unit Hilbert sphere, $\mathbb{S}_+^\infty = \left\{\psi \in \mathrm{L}^2 \;\middle|\; \|\psi\| = 1, \psi > 0\right\}$. On $\mathbb{S}_+^\infty$, any point $\tilde{\psi} \in \mathbb{S}_+^\infty$ can be projected to the tangent space $T_\psi(\mathbb{S}_+^\infty) = \left\{v \in \mathrm{L}^2 \;\middle|\; \int_0^1 \psi(t)v(t)\, dt = 0\right\}$ at ψ by the inverse exponential map (Srivastava & Klassen, 2016, p. 83) by

$$\left(\exp_\psi^{-1} \tilde{\psi}\right)(t) = \frac{\theta}{\sin(\theta)} \left(\psi(t) - \tilde{\psi}(t)\cos(\theta)\right), \quad t \in [0,1], \tag{I.7}$$

with $\theta = \cos^{-1}\left(\int_0^1 \psi(t)\tilde{\psi}(t)\, dt\right)$. Similarly, points in the tangent space can be projected back to the unit sphere $\mathbb{S}_+^\infty$ at the point ψ by the exponential map (Srivastava & Klassen, 2016, p. 83)

$$\left(\exp_\psi v\right)(t) = \cos\left(\|v\|\right)\psi(t) + \sin\left(\|v\|\right)\frac{v(t)}{\|v\|}, \quad t \in [0,1]. \tag{I.8}$$

The algorithm for finding the Karcher mean of warping functions under the Fisher-Rao metric follows Srivastava and Klassen (2016, Algorithm 24, p. 238) and is formally described in Algorithm I.1, with an initial estimate of the Karcher mean of the warping functions being the normalized cross-sectional mean of their SRSFs. This procedure is based on a fixed-point algorithm, see e.g., Bhattacharya and Bhattacharya (2012, chap. 5).

I.2.1. *Joint alignment of multivariate quasi-periodic data*

In this section, we describe the extension of the joint alignment algorithm to multivariate quasi-periodic functions. We assume that quasi-periodic functions are generated from a multiscale warping model as shown in Figure I.2. The terms describing the model are further defined below in this section.

Let a period be defined on the interval $[0, \tau]$ and let the quasi-periodic functional data have K periods. We assume that the length of the period, in this case τ, and the number of periods, K, are known. We define a periodic extension of a function $f\colon [0,\tau] \to \mathbb{R}$ as $\mathrm{ext}_{\mathrm{L}^2}^K f\colon [0, K\tau] \to \mathbb{R}$, $K \in \mathbb{N}$ with

$$\left(\mathrm{ext}_{\mathrm{L}^2}^K f\right)(t + k\tau) \coloneqq f(t) \quad \text{and} \quad \left(\mathrm{ext}_{\mathrm{L}^2}^K f\right)(K\tau) \coloneqq f(\tau), \tag{I.9}$$

Algorithm I.1 Karcher mean of warping functions

Input: Warping functions $\{\gamma_i\}_{i=1}^n$, stopping criterion c, maximum number of iterations E, step size ϵ

Output: Karcher mean μ_γ

$\psi_i \leftarrow \text{SRSF}(\gamma_i)$ ▷ Calculate SRSF representation of γ_i using (I.1)
$\mu_\psi \leftarrow \frac{1}{n}\sum_{i=1}^n \psi_i$ ▷ Initialize mean of SRSFs
$\mu_\psi \leftarrow \mu_\psi / \|\mu_\psi\|$ ▷ Normalize the mean of SRSFs
$\mu_v \leftarrow 0$ ▷ Initialize mean in the tangent space
$e \leftarrow 0$ ▷ Initialize the index of iteration
while $(\|\mu_v\| \geq c$ **OR** $e = 0)$ **AND** $e < E$ **do**
    $e \leftarrow e + 1$ ▷]Increase the index of iteration
    $\mu_\psi \leftarrow \exp_{\mu_\psi} \epsilon\mu_v$ ▷ Update mean of SRSFs using (I.8)
    $v_i \leftarrow \exp^{-1}_{\mu_\psi} \psi_i$ ▷ Project ψ_i to tangent space at μ_ψ using (I.7)
    $\mu_v \leftarrow \frac{1}{n}\sum_{i=1}^n v_i$ ▷ Calculate mean of projections $\{v_i\}$
end while
$\mu_\gamma \leftarrow \text{ToWarp}(\mu_\psi)$ ▷ Calculate mean warping function using (I.2)

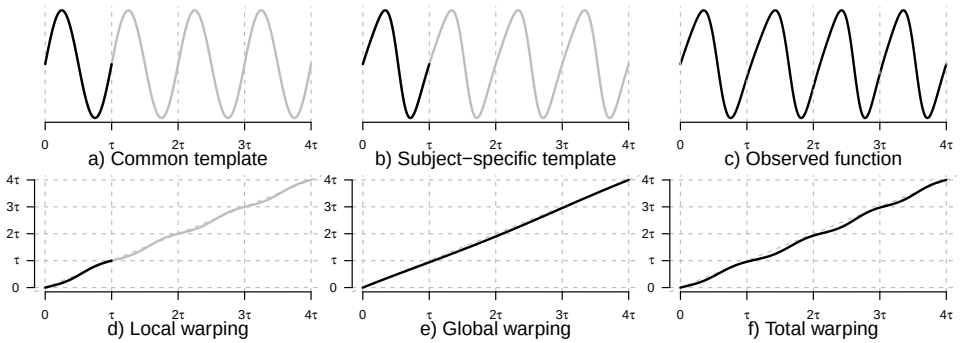


FIGURE I.2. *Multiscale warping model. The common template is represented by the black line on* $[0, \tau]$ *in Figure a), where the gray line represents the periodic extension of this template on* $[0, 4\tau]$. *The subject-specific template is shown in Figure b), which is a result of warping the common template with the subject-specific local warping function, as represented in Figure d). Figure c) shows the observed function that was obtained from the periodic extension of the subject-specific template warped with the subject-specific global warping function, as represented in Figure e). Finally, Figure f) shows the composition of the subject-specific local and the subject-specific global warping functions.*

for $t \in [0, \tau)$ and $k = 0, \ldots, K-1$. We also define a periodic extension of a warping function $\gamma\colon [0,\tau] \to [0,\tau]$ on $[0, K\tau]$ by

$$\left(\operatorname{ext}_{\Gamma}^{K} \gamma\right)(t + k\tau) \coloneqq \gamma(t) + k\tau \quad \text{and} \quad \left(\operatorname{ext}_{\Gamma}^{K} \gamma\right)(K\tau) \coloneqq K\tau, \tag{I.10}$$

for $t \in [0, \tau)$ and $k = 0, \ldots, K-1$. As a dual to the periodic extension, we define a split of a function $f\colon [0, K\tau] \to \mathbb{R}$ in its domain into K functions with $t \in [0, \tau]$ by

$$\begin{aligned}\left(\operatorname{spl}^{K} f\right)(t) &= (f(t), \ldots, f(t + (k-1)\tau), \ldots, f(t + (K-1)\tau)) \\ &\coloneqq (f_1(t), \ldots, f_k(t), \ldots, f_K(t))),\end{aligned} \tag{I.11}$$

for $k = 1, \ldots, K$. After performing the extension or the split of a warping function, we need to linearly transform the image of the resulting warping functions to the same interval as the domain in order to keep the boundary-preserving property, and furthermore, whenever the exponential and inverse exponential maps, Equations (I.8) and (I.7), respectively, need to be used, the domain and image of the warping functions need to be transformed to the interval $[0, 1]$.

The observed multivariate quasi-periodic functions $f_i\colon [0, K\tau] \to \mathbb{R}^J$, where $f_i = (f_{i1}, \ldots, f_{iJ})$ with their SRSF representations $q_i = (q_{i1}, \ldots, q_{iJ})$, are assumed to be generated from a multiscale warping model (see Figure I.2) by

$$\begin{aligned} f_{ij}(t) &= \left(\operatorname{ext}_{\mathrm{L}^2}^{K} \mu_{\cdot j} \circ \operatorname{ext}_{\Gamma}^{K} \gamma_i^l \circ \gamma_i^g\right)(t) \\ &= \left(\operatorname{ext}_{\mathrm{L}^2}^{K} \mu_{ij} \circ \gamma_i^g\right)(t) \\ &= \left(\operatorname{ext}_{\mathrm{L}^2}^{K} \mu_{\cdot j} \circ \gamma_i^t\right)(t), \end{aligned} \tag{I.12}$$

where $\gamma_i^l\colon [0,\tau] \to [0,\tau]$ is the *local* warping function (Figure I.2d, black) and $\operatorname{ext}_{\Gamma}^{K} \gamma_i^l\colon [0, K\tau] \to [0, K\tau]$ its periodic extension (Figure I.2d, black and gray), $\gamma_i^g\colon [0, K\tau] \to [0, K\tau]$ is the *global* warping function (FigureI.2e), and $\gamma_i^t\colon [0, K\tau] \to [0, K\tau]$ is the *total* warping function defined as the composition $\operatorname{ext}_{\Gamma}^{K} \gamma_i^l \circ \gamma_i^g$ (Figure I.2f). Note that the local warping function is defined on another domain than the global and total warping. In the rest of Section I.2.1, we use two subscripts for μ for clarity. The first subscript denotes individuals, and the second subscript denotes the elements of the multivariate functions. We think of $\mu_{\cdot\cdot} = (\mu_{\cdot 1}, \ldots, \mu_{\cdot J})$ as the common template (Figure I.2a, black) and $\mu_{i\cdot} = \mu_{\cdot\cdot} \circ \gamma_i^l = \left(\mu_{\cdot 1} \circ \gamma_i^l, \ldots, \mu_{\cdot J} \circ \gamma_i^l\right) = (\mu_{i1}, \ldots, \mu_{iJ})$ as the subject-specific template (Figure I.2b, black) on $[0, \tau]$.

On the other hand, given functions γ_i^l and γ_i^g, or γ_i^t, we can extract the periodic extension $\operatorname{ext}_{\mathrm{L}^2}^{K} \mu_{\cdot j}$ (Figure I.2a, black and gray) of the common template function $\mu_{\cdot j}$ (Fig-

ure I.2c, black) from an observed function f_{ij} (Figure I.2c) by

$$\begin{aligned}\left(\operatorname{ext}^K_{\mathrm{L}^2}\mu_{\cdot j}\right)(t) &= \left(f_{ij}\circ\left(\gamma_i^t\right)^{-1}\right)(t)\\ &= \left(f_{ij}\circ\left(\gamma_i^g\right)^{-1}\circ\left(\operatorname{ext}^K_{\Gamma}\gamma_i^l\right)^{-1}\right)(t)\\ &= \left(\operatorname{ext}^K_{\mathrm{L}^2}\mu_{ij}\circ\left(\operatorname{ext}^K_{\Gamma}\gamma_i^l\right)^{-1}\right)(t).\end{aligned} \tag{I.13}$$

Subject-specific template.. The warping functions $\{\gamma_i\}$ obtained by Algorithm I.2 correspond to the inverse of the total warping functions as described in Equations (I.12) and (I.13). In order to obtain subject-specific templates, we need to decompose this total warping function into the global and local warping functions, and we propose that this can be done in the following way. Consider the subject i and the corresponding warping function γ_i that aligns the observed function f_i to the periodic extension of the common template, $\operatorname{ext}^K_{\mathrm{L}^2}\mu_{\cdot\cdot}$ (see Figure I.2a). We split the domain of the warping function γ_i into K functions using Equation (I.11). We define the SRSF of the subject-specific template as the center of the orbit $[\mu_{q_i}]$ with respect to the set $\{(q_i,\gamma_i)_k\}_{k=1}^K$, the split of (q_i,γ_i). We find this by calculating the Karcher mean μ_{γ_i} of $\{\gamma_{ik}\}_{k=1}^K$ by first scaling their domain and image to the interval $[0,1]$, and then performing Algorithm I.1. We can represent the warping function γ_i as

$$\gamma_i = \gamma_i\circ\gamma_{\mathrm{id}} = \gamma_i\circ\left(\operatorname{ext}^K_{\Gamma}\mu_{\gamma_i}\right)^{-1}\circ\operatorname{ext}^K_{\Gamma}\mu_{\gamma_i}, \tag{I.14}$$

where $\operatorname{ext}^K_{\Gamma}\mu_{\gamma_i}$ is a periodic extension of μ_{γ_i}, with the domain and image scaled to the interval $[0,1]$. Furthermore, we have the following identities

$$\gamma_i = \left(\gamma_i^t\right)^{-1},\quad \gamma_i\circ\left(\operatorname{ext}^K_{\Gamma}\mu_{\gamma_i}\right)^{-1} = \left(\gamma_i^g\right)^{-1},\quad \mu_{\gamma_i} = \left(\gamma_i^l\right)^{-1}, \tag{I.15}$$

where γ_i^t, γ_i^g and γ_i^l are the total, global and local warping functions, respectively, as described in Equation (I.12).

The SRSF of the common template, μ_q, can be obtained from Algorithm I.2, and the common template $\mu_{\cdot\cdot}$ can be obtained from its SRSF using Equation (I.2) up to a constant $c\in\mathbb{R}$, $\mu_{\cdot\cdot}(t) = c + \int_0^t \mu_q(s)|\mu_q(s)|\,ds$. The subject-specific template can be extracted using identities in Equations (I.12) and (I.15) by

$$\mu_i = \mu\circ\mu^l_{\gamma_i} = \mu\circ\mu^{-1}_{\gamma_i}. \tag{I.16}$$

Quasi-periodic functions with amplitude variability.. In addition to the multiscale time warping model, the amplitude of the subject-specific template can be further susceptible to amplitude variability, which means that the amplitudes in the different periods might not be the same. Furthermore, amplitudes measured on different subjects do not have to be the same either, thus there is no longer the same relationship between $\mu_{\cdot j}$ and μ_{ij} as described in Equation (I.13), and the subject-specific template cannot be obtained by Equation (I.16).

To obtain the SRSF of the subject-specific template in the presence of amplitude vari-

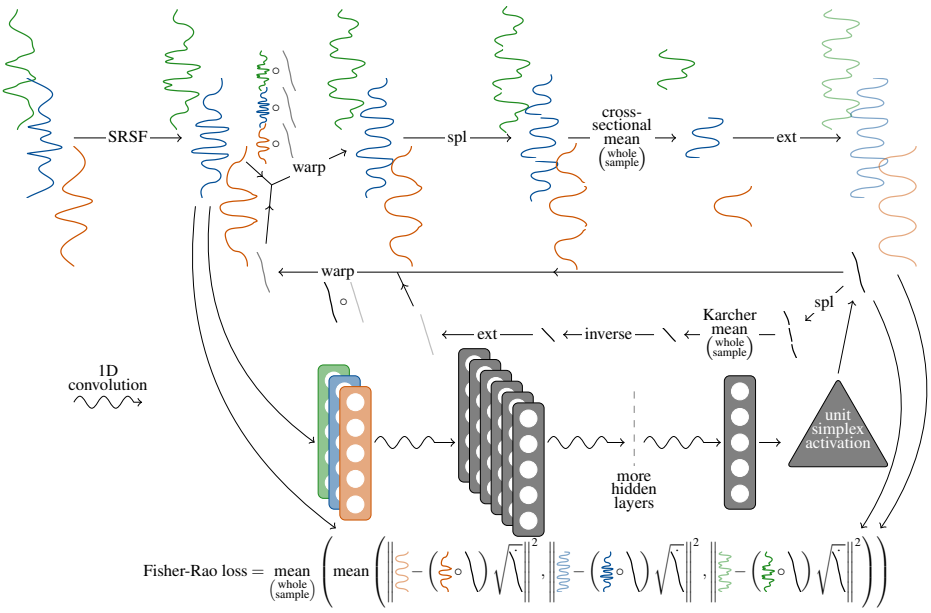


FIGURE I.3. *Diagram of the algorithm for joint alignment of multivariate quasi-periodic functions. Note that both the cross-sectional and Karcher mean are taken over the whole sample. Furthermore, the domain and image of the warping functions that are involved in the Karcher mean are first scaled to the interval* $[0, 1]$.

ability, we propose calculating the cross-sectional mean of the aligned functional data

$$\mu_{q_i}(t) = \frac{1}{K} \sum_{k=1}^{K} \left(q_i, \gamma_i'\right)_k (t),$$

where $t \in [0, 1]$, $\left\{\left(q_i, \gamma_i'\right)_k\right\}$ is the split of $\left(q_i, \gamma_i'\right)$ using Equation (I.11). Finally, the subject-specific template can be obtained up to a constant from μ_{q_i} using Equation (I.2).

I.2.2. *Algorithm for joint alignment of multivariate quasi-periodic data using deep learning*

The algorithm for adaptive template-based groupwise registration follows Srivastava and Klassen (2016, Algorithm 33., p. 277) with some modifications, and is represented in Algorithm I.2, as well as in Figure I.3. The most important modifications are replacing dynamic programming (Srivastava & Klassen, 2016, Algorithm 58., p. 437) with a convolutional neural network, and extending the algorithm to handle quasi-periodic multivariate functions.

Algorithm I.2 Joint alignment of multivariate quasi-periodic functional data using deep learning: DeepJAM

Output: Observed functions $\{f_i\}_{i=1}^n$ on the interval $I = [0,1]$, number of periods K, number of iterations E, initialized neural network NN

Input: Warping functions $\{\gamma_i\}_{i=1}^n$, SRSF μ_q of the common template

$\gamma_i \leftarrow \gamma_{\text{id}}$ ▷ Initialize γ_i to identity warp
$q_i \leftarrow \text{SRSF}(f_i)$ ▷ Calculate SRSF representation of f_i using (I.1)
for e **from** 1 **to** E **do**
  $\tilde{q}_i \leftarrow (q_i, \gamma_i)$ ▷ Warp in the SRSF space using (I.3)
  $(\tilde{q}_{i1}, \ldots, \tilde{q}_{iK}) \leftarrow \text{spl}^K \tilde{q}_i$ ▷ Split the SRSFs using (I.11)
  $\mu_{\tilde{q}} \leftarrow \frac{1}{nK} \sum_{i=1}^n \sum_{k=1}^K \tilde{q}_{ik}$ ▷ Calculate mean of $\{\tilde{q}_{ik}\}$
  $\mu_{\tilde{q}}^K \leftarrow \text{ext}_{\text{L}^2}^K \mu_{\tilde{q}}$ ▷ Calculate extension of $\mu_{\tilde{q}}$ using (I.9)
  train NN$(x = \{q_i\}, y = \mu_{\tilde{q}}^K)$ for one epoch
  $\tilde{\gamma}_i \leftarrow \text{NN}(x = q_i)$ ▷ Calculate neural network prediction
  $(\tilde{\gamma}_{i1}, \ldots, \tilde{\gamma}_{iK}) \leftarrow \text{spl}^K \tilde{\gamma}_i$ ▷ Split warping functions using (I.11)
  $\tilde{\gamma}'_{ik} \leftarrow \text{Scale}(\tilde{\gamma}_{ik})$ ▷ Scale domain and image of $\tilde{\gamma}_{ik}$ to $[0,1]$
  $\mu_{\tilde{\gamma}} \leftarrow \text{KarcherMean}(\{\tilde{\gamma}'_{ik}\})$ ▷ Find the Karcher mean of $\{\tilde{\gamma}'_{ik}\}$ as in Algorithm I.1
  $\gamma^* \leftarrow \text{ext}_{\Gamma}^K \mu_{\tilde{\gamma}}^{-1}$ ▷ Calculate extension of $\mu_{\tilde{\gamma}}^{-1}$ using (I.10)
  $\gamma' \leftarrow \text{Scale}(\gamma^*)$ ▷ Scale domain and image of γ^* to $[0,1]$
  $\gamma_i \leftarrow \tilde{\gamma}_i \circ \gamma'$ ▷ Calculate warping functions using (I.6)
end for
$\gamma' \leftarrow \text{Scale}\left(\mu_{\tilde{\gamma}}^{-1}\right)$ ▷ Scale domain and image of $\mu_{\tilde{\gamma}}^{-1}$ to [0,1/K]
$\mu_q \leftarrow (\mu_{\tilde{q}}, \gamma')$ ▷ Calculate SRSF of common template using (I.5)

*$\text{NN}(x = \{q_i\}, y = \mu_{\tilde{q}}^K)$ and $\text{NN}(x = q_i)$ represent a neural network with input $\{q_i\}$ and q_i, respectively, and outcome $\mu_{\tilde{q}}^K$.

I.2.3. *Deep learning architecture*

The general architecture of a warping neural network can be quite flexible. The most important part is that it produces an output that has similar dimension of the discretized input functions and can be transformed into a warping function. In our case, this is achieved by using convolutional layers with a padding such that the layer output has the same length as the layer input. The outputs of the neural network are the warping functions, which do not need to have exactly the same dimension as the input layer. In this case, we can use interpolation to scale the output up or down. Furthermore, to save computation time and because of the specific loss function described below, instead of inputting the original functions, we use their SRSF representation as inputs directly. We implemented the neural network using Keras (Chollet et al., 2015) and TensorFlow (Abadi et al., 2015) in the R packages `keras` (Allaire & Chollet, 2022) and `tensorflow` (Allaire & Tang, 2022).

The output of the neural network uses a special activation function based on the unit simplex transformation, so that the resulting warping functions are boundary-preserving increasing functions on the interval $[0, 1]$. For a given neural network output $y \in \mathbb{R}^P$ before any activation is applied, we calculate new coordinates $z \in \mathbb{R}^P$ by

$$z_p = \operatorname{expit}\left(y_p - \log(P - p + 1)\right), \text{ for } 1 \leq p \leq P, \tag{I.17}$$

where $\operatorname{expit}(x) = 1/(1+\exp(-x))$, ensuring that the new coordinates are all between 0 and 1. The vector z is then used to determine a vector $x \in \mathbb{R}^{P+1}$ by

$$x_p = \begin{cases} z_p, & p = 1, \\ \left(1 - \sum_{p'=1}^{p-1} x_{p'}\right) z_p, & 1 < p \leq P, \\ 1 - \sum_{p'=1}^{P} x_{p'}, & p = P + 1, \end{cases}$$

with the property that all the coordinates of x are greater than 0, and that $\sum_{p=1}^{P+1} x_p = 1$. Looking back at Equation (I.17), the offset $-\log(P-p+1)$ is added so that the zero vector y is mapped to the simplex $x = (1/(P+1), \ldots, 1/(P+1))$. Finally, we transform this vector x to a discretized warping function $\gamma \in \mathbb{R}^{P+2}$ by

$$\gamma_p = \begin{cases} 0, & p = 1, \\ \sum_{p'=1}^{p-1} x_{p'} & 1 < p \leq P + 2. \end{cases}$$

In order to obtain a vector in $\mathbb{R}^P$, we propose linearly interpolating the values of γ, so that $\gamma \in \mathbb{R}^P$.

The loss function of our warping neural network is defined as follows. Let f_i, where $f_i(t) = (f_{i1}(t), \ldots, f_{iJ}(t)) \in \mathbb{R}^J$, be the observed multivariate functions and q_{ij} the SRSF representations of f_{ij}. Let μ_{q_j} be the SRSF representation of the template μ_j, $j = 1, \ldots, J$, and let γ_i be the outputs of the neural network. The Fisher-Rao loss function is then defined as

$$\mathcal{L} = \frac{1}{n} \sum_{i=1}^{n} \frac{1}{J} \sum_{j=1}^{J} \|\mu_{q_j} - (q_{ij}, \gamma_i)\|^2.$$

The univariate function case is easily obtainable by setting $J = 1$.

We used the Adam optimizer (Kingma & Ba, 2015). The architectural hyperparameters such as the number of layers, number of filters, kernel size, and learning rate were chosen with the Bayesian optimization algorithm (Snoek et al., 2012) using the expected improvement acquisition function. Finally, the hyperbolic tangent was used as an activation function of the hidden layers.

I.3. SIMULATION STUDY

As a proof-of-concept of our neural network approach, we deploy two simulation studies: one for univariate quasi-periodic functions and one for multivariate quasi-periodic functions. In the case of univariate functions, we did not vary the amplitudes, hence the functions can be perfectly aligned, i.e., the distance between the true template and the aligned functions converges to zero with increasing sample size and complexity of the neural network. In the case of multivariate functions we varied both the inter-subject and intra-subject amplitudes, thus the peaks and valleys of the functions can be temporally aligned to the common template, but the vertical distance between them does not converge to zero.

In both cases, we generated $N = 14,000$ functions and used 8,000 of them as the training data, $2,000$ for hyperparameter tuning, $2,000$ as the validation data, and $2,000$ as the test data.

To evaluate the performance of our method, we use the decrease in the cumulative cross-sectional variance of the observed and aligned data. The cumulative cross-sectional variance is a measure of the average distance of the functions from the mean, defined as follows. Let $\{f_i(t), t \in [0,1]\}_{i=1}^n$ be a functional dataset, then the cumulative cross-sectional variance is

$$\widehat{\text{Var}}\left(\{f_i\}\right) = \frac{1}{n-1}\int_0^1 \sum_{i=1}^n \left(f_i(t) - \mu(t)\right)^2 dt.$$

However, since we do not know the true common template μ for the real application data, we will use the cross-sectional mean in the calculation instead

$$\overline{\text{Var}}\left(\{f_i\}\right) = \frac{1}{n-1}\int_0^1 \sum_{i=1}^n \left(f_i(t) - \frac{1}{n}\sum_{i=1}^n f_i(t)\right)^2 dt.$$

A decrease in the cumulative cross-sectional variance is desirable, because the cumulative cross-sectional variance of the observed data contains both the amplitude and phase variability, whereas the cumulative cross-sectional variance of the aligned data represents only the variability of the amplitude. Furthermore, we calculate the square of the L^2 norm of the distance between the cross-sectional mean and the true common template by

$$\left\|\frac{1}{n}\sum_{i=1}^n f_i - \mu\right\|^2 = \int_0^1 \left(\frac{1}{n}\sum_{i=1}^n f_i(t) - \mu(t)\right)^2 dt.$$

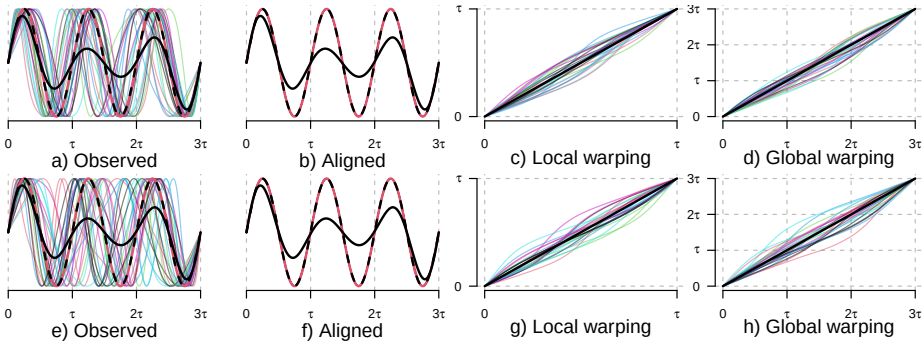


FIGURE I.4. *Result of the DeepJAM algorithm. The first row represents the training data and the second row represents the test data. Figures a) and e) show random 25 observed functions from the training and test data, respectively, while Figures b) and f) show the same functions, but aligned using the DeepJAM neural network, respectively for the training and test data. The thick red lines are the true common template extended to three periods, and the thick black full/dashed lines represent the cross-sectional mean of the full observed/aligned training and test data, separately. Figures c), d, g) and f) show the estimated local and global warping of these random 25 functions from the training and test data, where the thick black lines are the identity lines.*

I.3.1. *Scenario 1, univariate functions*

In the first study, we simulated the common template to be the sine wave on the interval $[0, 2\pi]$, $\mu_f = \sin(t)$, $t \in [0, 2\pi]$, sampled at 65 equidistant points on each of the $K = 3$ periods, adding up to $P = 193$ equidistant points. Using the described notation, the periodic extension of the template is a sine wave on the interval $[0, 6\pi]$, $\text{ext}^K_{\text{L}^2}\, \mu = \sin(t)$, $t \in [0, 6\pi]$. The domain of the functions were then scaled to the interval $[0, 1]$. We simulated $N = 14,000$ global and $N = 14,000$ local warping functions denoted γ_i^g and γ_i^l using the `fdasrvf` package (Tucker, 2022), respectively. The warping functions using this package are generated in a way, so that their Karcher mean is the identity function. However, to ensure the identifiability of the subject-specific template, we need to transform the generated warping functions using Equation (I.14) and the identities in Equation (I.15). The observed functions are then generated as $f_i(t) = \left(\text{ext}^K_{\text{L}^2}\, \mu \circ (\gamma_i^l)^K \circ \gamma_i^g\right)(t)$.

The optimized architecture consists of an input layer with 193 nodes and one channel, followed by 17 convolutional layers with a kernel of size 64. The hidden layers have 25 filters, while the output layer has only one filter. The learning rate was chosen to be $3.66 \cdot 10^{-7}$.

Figure I.4 shows the observed and aligned functions, as well as the corresponding estimated local and global warping functions. We can see that we achieved almost perfect alignment of both the training and test data. Furthermore, we can see that the cross-sectional mean of the observed data still preserves the modality in the data, but the size of the amplitudes is distorted. The cumulative cross-sectional variance of the simulated multivariate data can be found in Table I.1.

Functions			Mean		
Observed	Aligned	% ↓	Observed	Aligned	% ↓
0.475	0.001	99.81	0.148	$1.89 \cdot 10^{-5}$	99.99

TABLE I.1. *Cumulative cross-sectional variance of the observed and aligned simulated univariate data, where the columns "% ↓" show the reduction of the cumulative cross-sectional variance in %. In addition, we calculate the square of the* L^2 *distance between the cross-sectional mean and the true template.*

I.3.2. *Scenario 2, multivariate functions*

In this example, the functions to be aligned are f_i where the image of the functions at each t is $\mathbb{R}^3$. We write $f_i(t) = (f_{i1}(t), f_{i2}(t), f_{i3}(t))$, and we think of each function f_{ij}, as a univariate function that is the observed version of y_{ij} generated as follows. Let $z_{ip} \sim \mathcal{N}(1, 0.25^2)$, $p = 1, \ldots, 18$. The functions $y_{i1} \colon [0, 6\pi] \to \mathbb{R}$ were generated as

$$y_{i1}(t) = \begin{cases} z_{i1} \sin(t) & t \in [0, 2\pi), \\ z_{i2} \sin(t) & t \in [2\pi, 4\pi), \\ z_{i3} \sin(t) & t \in [4\pi, 6\pi]. \end{cases}$$

The common template is then $\mu_1(t) = \sin(t), t \in [0, 2\pi]$. The functions y_{i2} were generated as

$$y_{i2}(t) = \begin{cases} z_{i4}\,(c_1(x) - c_3(x)) + c_3(x) + z_{i5}\,(c_2(x) - c_4(x)) + c_4(x), \\ \qquad\qquad t \in [0, 6), x = t - 3, \\ z_{i6}\,(c_1(x) - c_3(x)) + c_3(x) + z_{i7}\,(c_2(x) - c_4(x)) + c_4(x), \\ \qquad\qquad t \in [6, 12), x = t - 9, \\ z_{i8}\,(c_1(x) - c_3(x)) + c_3(x) + z_{i9}\,(c_2(x) - c_4(x)) + c_4(x), \\ \qquad\qquad t \in [12, 18], x = t - 15, \end{cases}$$

where

$$c_1(x) = e^{-(x-4.5)^2/2}, \qquad c_2(x) = e^{-(x-1.5)^2/2},$$

$$c_3(x) = \frac{e^{-\frac{4.5^2}{2}} - e^{-\frac{1.5^2}{2}}}{6} x + \frac{e^{-\frac{4.5^2}{2}} + e^{-\frac{1.5^2}{2}}}{2}, \quad c_4(x) = \frac{e^{-\frac{1.5^2}{2}} - e^{-\frac{4.5^2}{2}}}{6} x + \frac{e^{-\frac{4.5^2}{2}} + e^{-\frac{1.5^2}{2}}}{2}.$$

This transformation is applied to ensure that $y_{i2}(0) = y_{i2}(6) = y_{i2}(12) = y_{i2}(18)$, allowing continuity at the limits of the periods. The common template is therefore $\mu_2(t) =$

Functions	Observed	Aligned	% ↓	Mean	Observed	Aligned	% ↓
1	0.506	0.038	92.54	1	0.148	0.001	99.03
2	0.571	0.144	74.76	2	0.194	0.003	98.23
3	0.420	0.033	92.06	3	0.126	0.002	98.75

TABLE I.2. *Cumulative cross-sectional variance of the observed and aligned simulated multivariate data, where columns "% ↓" show the reduction of the cumulative cross-sectional variance in %. In addition, we calculate the square of the* L^2 *distance between the cross-sectional means and the true templates.*

$e^{-(t-4.5)^2/2} + e^{-(t-1.5)^2/2}$, $t \in [-3, 3]$. Finally, functions y_{i3} were generated as

$$y_{i3}(t) = \begin{cases} z_{i10}p_2(x) - z_{i11}(p_1(x) - p_1(0)), & t \in [0, 0.5), x = t \\ z_{i10}p_2(x) + z_{i12}(p_3(x) - p_3(0)), & t \in [0.5, 1), x = t \\ z_{i13}p_2(x) - z_{i14}(p_1(x) - p_1(0)), & t \in [1, 1.5), x = t - 1 \\ z_{i13}p_2(x) + z_{i15}(p_3(x) - p_3(0)), & t \in [1.5, 2), x = t - 1 \\ z_{i16}p_2(x) - z_{i17}(p_1(x) - p_1(0)), & t \in [2, 2.5), x = t - 2 \\ z_{i16}p_2(x) + z_{i18}(p_3(x) - p_3(0)), & t \in [2.5, 3), x = t - 2, \end{cases}$$

where p_1 is the probability density function of $\mathcal{N}(0.25, 0.1^2)$, p_2 is the probability density function of $\mathcal{N}(0.5, 0.15^2)$ and p_3 is the probability density function of $\mathcal{N}(0.75, 0.1^2)$. The template function is then

$$\mu_3(t) = \begin{cases} p_2(t) - (p_1(t) - p_1(0)), & t \in [0, 0.5], \\ p_2(t) + (p_3(t) - p_3(0)), & t \in [0.5, 1], \end{cases}$$

In addition, functions y_{i2} and y_{i3} were further transformed with an affine transformation so that the image of the template functions μ_2 and μ_3 is in the interval $[-1, 1]$. Finally, the domains of all functions y_{ij}, as well as the extensions of the template functions, $\mathrm{ext}_{\mathrm{L}^2}^K \mu_j$, were scaled to the interval $[0, 1]$, and sampled at $P = 193$ equidistant points. The functions were then warped as $f_{ij}(t) = \left(y_{ij} \circ \mathrm{ext}_{\Gamma}^K \gamma_i^l \circ \gamma_i^g\right)(t)$.

The optimized architecture consists of an input layer with 193 nodes and three channels, followed by 14 convolutional layers with a kernel of size 60. The hidden layers have 56 filters, while the output layer has only one filter. The learning rate was chosen to be $9.35 \cdot 10^{-6}$.

Figure I.5 shows the observed and aligned test functions. We can see that we achieved nearly perfect phase alignment. The functions were simulated with amplitude variability, which is retained in the aligned data. Notice how the cross-sectional means of the observed data misrepresent the intrinsic shape of the underlying data generating model. Furthermore, the periodicity is also lost, especially for the cross-sectional mean of the functions $\{f_{i2}\}$. The cumulative cross-sectional variance of the simulated multivariate data can be found in Table I.2.

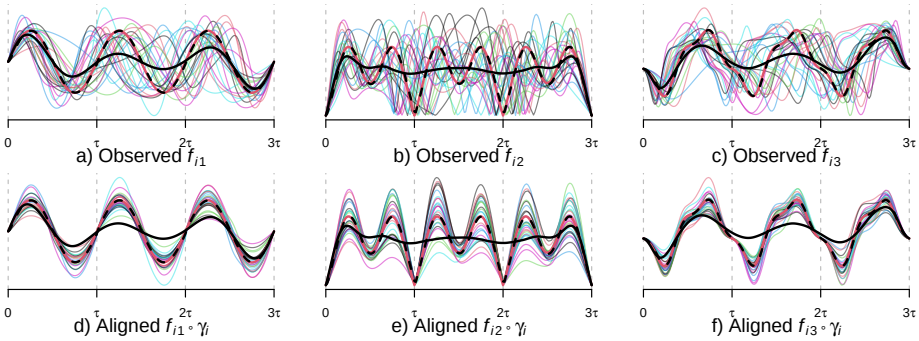


FIGURE I.5. *Result of the DeepJAM algorithm. Figures a), b) and c) show random 25 observed functions from the test data. The observed functions are the functions* f_{ij}, $i = 1, \ldots, n$, $j = 1, 2, 3$, *as described in Section I.3.2. Figures d), e) and f) show the same functions, but aligned using the DeepJAM neural network. The thick red lines are the true common templates extended to three periods, and the thick black full/dashed lines represent the cross-sectional means of the full observed/aligned test data.*

I.4. APPLICATION

For the application, we chose 12-lead 10s ECG recordings. The ECG study population consisted of participants in the Copenhagen General Population Study randomly sampled from the general population in Copenhagen, Denmark (Fuchs et al., 2023; Kühl et al., 2019). All ECGs were recorded with the study participant at rest and in supine position prior to a cardiac computed tomography scan. Written informed consent was obtained from all participants, and the study was approved by the local ethics committee (H-KF-01-144/01). The ECGs were preprocessed using the `neurokit2` python module (Makowski et al., 2021). First, the ECGs were filtered to remove noise and to improve the detection of peaks using the function `ecg_clean` with default settings for the method. Afterward, the function `ecg_peaks` was used to detect the location of R-peaks, and for the purpose of this application, we only selected a subset of the ECGs between the second and fifth R-peak, giving rise to a three-period time series, with each period between two subsequent R-peaks. The ECGs were then resampled to $P = 301$ equidistant points. Furthermore, because of the position of the electrodes, four leads can be derived from other leads (Horáček, 2010), thus we will only consider eight leads: I, II, V1–V6. We also only study a subset of the ECGs, resulting in 9,645 ECG recordings. From these, we used $5,511$ for training, $1,378$ for hyperparameter tuning, $1,378$ as the validation data and $1,378$ as the test data.

The optimized architecture consists of an input layer with 301 nodes and eight channels, followed by 17 convolutional layers with a kernel of size 101. The hidden layers have 30 filters, while the output layer has only one filter. The learning rate was chosen to be $2.84 \cdot 10^{-5}$.

Figure I.6 shows the observed and aligned ECG recordings, as well as the corresponding estimated local and global warping functions. In addition, we present a comparison of the cross-sectional means of the observed and aligned data. Table I.3 shows the cumulative cross-sectional variance of the observed and aligned data per lead. We can see that alignment using the DeepJAM neural network reduced the cumulative cross-sectional

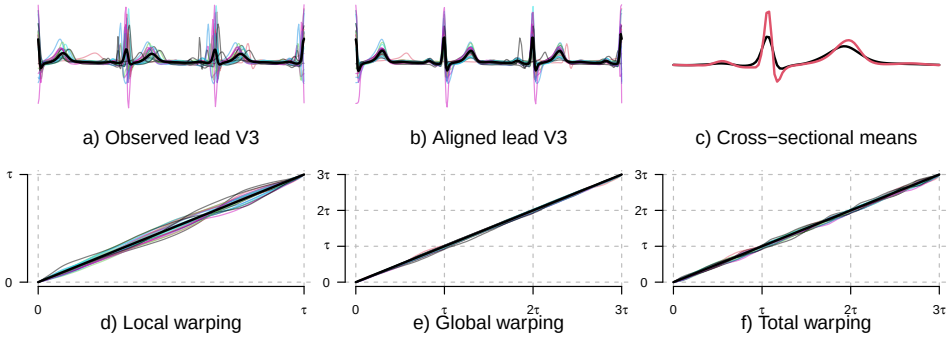


FIGURE I.6. *Result of the DeepJAM algorithm. Figures a) and b) show the observed and aligned lead V3, respectively, of random 25 test ECGs, where the thick black lines represent the cross-sectional means of the full observed and aligned data, respectively. Figure c) shows again the cross-sectional means of the full observed data (black) and the full aligned data (red), where we only present one period. Figures d), e) and f) show the local, global, and total warping of the 25 random test ECGs, with the thick black lines representing the identity line.*

Lead	Observed	Aligned	% ↓
I	0.005	0.003	32.87
II	0.017	0.010	41.08
V1	0.010	0.007	27.46
V2	0.024	0.017	29.50
V3	0.029	0.020	30.68
V4	0.026	0.015	41.23
V5	0.020	0.011	45.57
V6	0.014	0.007	46.32

TABLE I.3. *Cumulative cross-sectional variance of the observed and aligned ECG data, where column "% ↓" shows the reduction of the cumulative cross-sectional variance in %.*

variance by 27.46%–46.32%.

I.5. CONCLUSION

In this paper, we have presented a novel method for joint alignment of multivariate quasi-periodic functional data, addressing the phase variability in the data that is often overlooked by traditional approaches. We applied a convolutional neural network with a unique activation function based on the unit simplex transformation to achieve effective phase-amplitude separation. To extend our approach to multivariate functions, we incorporated a convolutional layer with multiple channels in the input layer. The output layer, on the other hand, provides a single warping function per subject, which simultaneously aligns all dimensions of the observed function. We also incorporated the multiscale warping model to accommodate quasi-periodic functions. To train our model, we used a loss function based on the Fisher-Rao distance between the square-root slope representations

of functions. Notably, our approach is unsupervised and capable of generating common and subject-specific template functions.

We conducted experiments on two simulated datasets as well as a real dataset of 12-lead 10 second electrocardiogram recordings and demonstrated that our method could accurately separate the phase variability from the amplitude variability with nearly perfect phase alignment in the simulated data and a substantial reduction in cumulative cross-sectional variance in the real dataset. We only used very simple convolutional neural networks as a proof of concept. The best models selected using Bayesian optimization were of varying complexity but yielded similar performances. The optimal model architecture depends very much on the data.

Future work could focus on exploring potential enhancements to the neural network architecture, such as including dropout layers, batch normalization layers, skip layers, as well as different optimizers, and activation functions of the hidden layers. In addition, architectures handling input of various length could be relevant for future work.

We have shown that using DeepJAM for aligning multivariate quasi-periodic functional data decrease the cumulative cross-sectional variance and yields a better representation of the salient features present in multivariate functional data. The implication of this on the analysis of ECG data in more complicated regression or prediction settings is beyond the scope of this paper. Nevertheless, the results show that DeepJAM retains the peaks and valleys of the ECG, and we thus conjecture, that incorporating our method in a larger statistical model could enhance accurate identification of particular ECG characteristics, such as the PR interval or the QRS complex. The argument behind this conjecture is that the conditional mean in any prediction model, e.g., $\mathrm{E}(Y_i \mid f_i) = Pf_i$ for some prediction functional P will be first-order biased when f_i is not properly aligned. This can be seen from a Taylor expansion around the identity warp, $\gamma_{\mathrm{id}}(t) = t$, as in $\mathrm{E}(Y_i \mid f_i \circ \gamma_i) \approx Pf_i + P[\dot{f}_i(\gamma_i - \gamma_{\mathrm{id}})]$. This bias is subject-specific and can be severe depending on the magnitude of the predictor velocities.

The contribution of our research extends beyond the specific dataset analyzed in this paper. The joint alignment of multivariate functional data with quasi-periodic characteristics has extensive applications in various fields such as signal processing, computer vision, and medical applications such as neuroscience and data from wearable devices.

BIBLIOGRAPHY

ABADI, M., AGARWAL, A., BARHAM, P., BREVDO, E., CHEN, Z., CITRO, C., et al. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. https://www.tensorflow.org/

ALLAIRE, J., & CHOLLET, F. (2022). *Keras: R interface to 'keras'* [R package version 2.10.0]. https://CRAN.R-project.org/package=keras

ALLAIRE, J., & TANG, Y. (2022). *Tensorflow: R interface to 'tensorflow'* [R package version 2.10.0]. https://CRAN.R-project.org/package=tensorflow

BHATTACHARYA, A., & BHATTACHARYA, R. (2012). *Nonparametric inference on manifolds: With applications to shape spaces* (1st). Cambridge University Press. https://doi.org/10.1017/CBO9781139094764

BOUCHEHAM, B. (2008). Matching of quasi-periodic time series patterns by exchange of block-sorting signatures. *Pattern Recognition Letters*, *29*(4), 501–514. https://doi.org/10.1016/j.patrec.2007.11.004

BOULNEMOUR, I., & BOUCHEHAM, B. (2018). QP-DTW: Upgrading dynamic time warping to handle quasi periodic time series alignment. *Journal of Information Processing Systems*, *14*(4), 851–876. https://doi.org/10.3745/JIPS.02.0090

CHEN, C., & SRIVASTAVA, A. (2021). SrvfRegNet: Elastic function registration using deep neural networks. *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 4462–4471. https://doi.org/10.1109/CVPRW53098.2021.00503

CHOLLET, F., et al. (2015). Keras. https://keras.io

FUCHS, A., KÜHL, J. T., SIGVARDSEN, P. E., AFZAL, S., KNUDSEN, A. D., MØLLER, M. B., et al. (2023). Subclinical coronary atherosclerosis and risk for myocardial infarction in a danish cohort: A prospective observational cohort study. *Annals of Internal Medicine*, *176*(4), 433–442. https://doi.org/10.7326/M22-3027

GREGG, R. E., ZHOU, S. H., LINDAUER, J. M., HELFENBEIN, E. D., & GIULIANO, K. K. (2008). What is inside the electrocardiograph? *Journal of Electrocardiology*, *41*(1), 8–14. https://doi.org/10.1016/j.jelectrocard.2007.08.059

HORÁČEK, B. M. (2010). Lead theory. In *Comprehensive electrocardiology* (2nd). Springer.

KINGMA, D. P., & BA, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*. https://doi.org/10.48550/ARXIV.1412.6980

KLONOFF, D. C. (2005). Continuous glucose monitoring: Roadmap for 21st century diabetes therapy. *Diabetes Care*, *28*(5), 1231–1239. https://doi.org/10.2337/diacare.28.5.1231

KNEIP, A., & RAMSAY, J. O. (2008). Combining registration and fitting for functional models. *Journal of the American Statistical Association*, *103*(483), 1155–1165. https://doi.org/10.1198/016214508000000517

KÜHL, J. T., NIELSEN, J. B., STISEN, Z. R., FUCHS, A., SIGVARDSEN, P. E., GRAFF, C., et al. (2019). Left ventricular hypertrophy identified by cardiac computed tomography and ECG in hypertensive individuals: A population-based study. *Journal of Hypertension*, *37*(4), 739–746. https://doi.org/10.1097/HJH.0000000000001962

LECUN, Y., BENGIO, Y., & HINTON, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

LOHIT, S., WANG, Q., & TURAGA, P. (2019). Temporal transformer networks: Joint learning of invariant and discriminative time warping. *Proceedings of the 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 12418–12427. https://doi.org/10.1109/CVPR.2019.01271

MAKOWSKI, D., PHAM, T., LAU, Z. J., BRAMMER, J. C., LESPINASSE, F., PHAM, H., SCHÖLZEL, C., & CHEN, S. H. A. (2021). NeuroKit2: A python toolbox for neurophysiological signal pro-

cessing. *Behavior Research Methods*, *53*(4), 1689–1696. https://doi.org/10.3758/s13428-020-01516-y

MARRON, J. S., RAMSAY, J. O., SANGALLI, L. M., & SRIVASTAVA, A. (2015). Functional data analysis of amplitude and phase variation. *Statistical Science*, *30*(4), 468–484. https://doi.org/10.1214/15-STS524

MCDONNELL, E. I., ZIPUNNIKOV, V., SCHRACK, J. A., GOLDSMITH, J., & WROBEL, J. (2022). Registration of 24-hour accelerometric rest-activity profiles and its application to human chronotypes. *Biological Rhythm Research*, *53*(8), 1299–1319. https://doi.org/10.1080/09291016.2021.1929673

MÜLLER, M. (2007). *Information retrieval for music and motion* (1st). Springer.

NUNEZ, E., & JOSHI, S. H. (2020). Deep learning of warping functions for shape analysis. *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 3782–3790. https://doi.org/10.1109/CVPRW50498.2020.00441

NUNEZ, E., LIZARRAGA, A., & JOSHI, S. H. (2021). SrvfNet: A generative network for unsupervised multiple diffeomorphic functional alignment. *Proceedings of the 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 4476–4484. https://doi.org/10.1109/CVPRW53098.2021.00505

OH, J., WANG, J., & WIENS, J. (2018). Learning to exploit invariances in clinical time-series data using sequence transformer networks. *Proceedings of the 3rd Machine Learning for Healthcare Conference*, *85*, 332–347.

RAMSAY, J. O., & SILVERMAN, B. W. (2005). *Functional data analysis* (2nd). Springer. https://doi.org/10.1007/b98888

SNOEK, J., LAROCHELLE, H., & ADAMS, R. P. (2012). Practical bayesian optimization of machine learning algorithms. *Proceedings of the Advances in Neural Information Processing Systems*, *25*.

SRIVASTAVA, A., & KLASSEN, E. P. (2016). *Functional and shape data analysis* (1st). Springer New York. https://doi.org/10.1007/978-1-4939-4020-2

SUN, G.-Z., CHEN, H.-H., & LEE, Y.-C. (1992). Time warping invariant neural networks. *Advances in Neural Information Processing Systems*, *5*.

TALLEC, C., & OLLIVIER, Y. (2018). Can recurrent neural networks warp time? *arXiv*. https://doi.org/10.48550/ARXIV.1804.11188

TUCKER, J. D. (2022). *fdasrvf: Elastic Functional Data Analysis*. https://CRAN.R-project.org/package=fdasrvf

TUCKER, J. D., WU, W., & SRIVASTAVA, A. (2013). Generative models for functional data using phase and amplitude separation. *Computational Statistics & Data Analysis*, *61*, 50–66. https://doi.org/10.1016/j.csda.2012.12.001

WEBER, R. S., EYAL, M., SKAFTE, N., SHRIKI, O., & FREIFELD, O. (2019). Diffeomorphic temporal alignment nets. *Advances in Neural Information Processing Systems*, *32*.

A.1. OBSERVED AND ALIGNED ECG TEST DATA

Refer to Figure A.1 to see the performance of the DeepJAM algorithm on 8 leads of the ECGs.

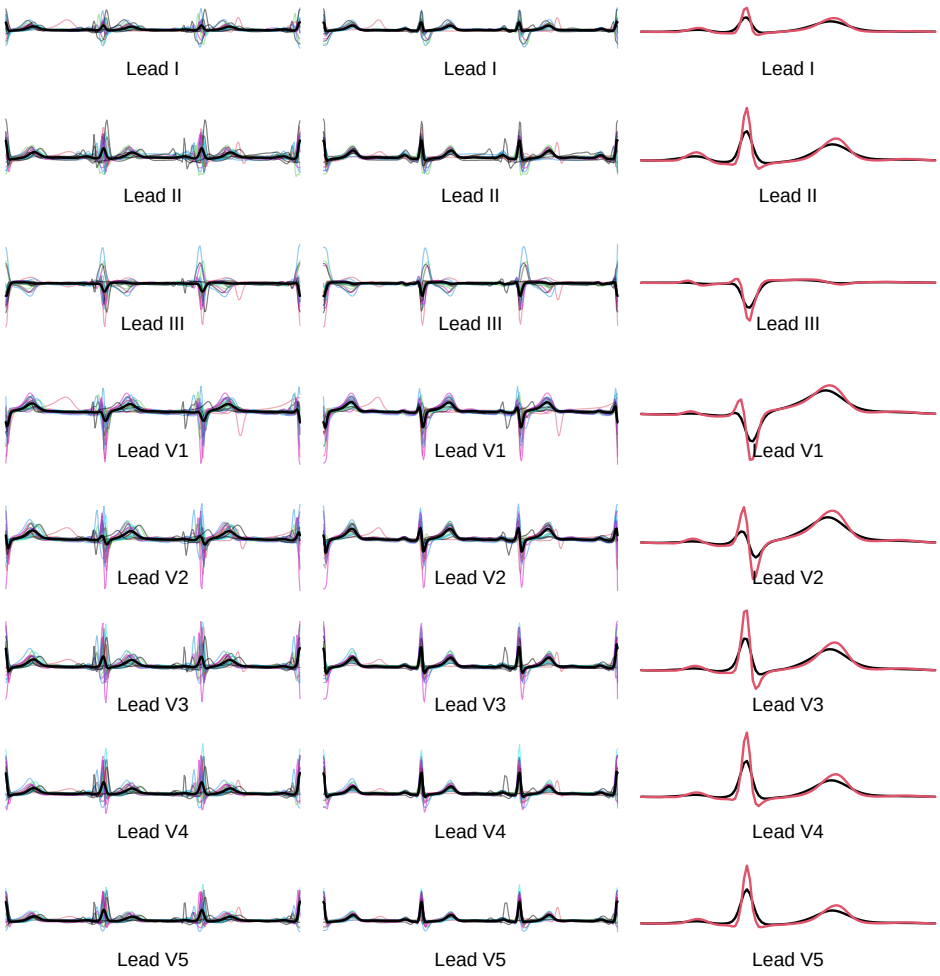


FIGURE A.1. *Result of the DeepJAM algorithm. The first column shows individual leads of random 25 ECG measurements, while the second column shows the same ECGs, but aligned. Finally, the third column shows the cross-sectional means of the full observed dataset (black) and the full aligned dataset (red), where we only show one period.*

Manuscript II

DEEP LEARNING FOR MULTIVARIATE FUNCTIONAL DATA WITH BUILT-IN TIME WARPING

VI THANH PHAM
Section of Biostatistics, University of Copenhagen, Denmark

ANDREAS KRYGER JENSEN
Section of Biostatistics, University of Copenhagen, Denmark

ABSTRACT

Functional data analysis is a powerful framework for interpreting data characterized by smooth, continuous processes. Recent advancements in deep learning have enhanced the ability to identify intricate patterns within complex datasets. This paper explores the use of functional neural networks for analyzing multivariate quasi-periodic functional data, with a focus on 12-lead electrocardiogram (ECG) recordings. We propose an adaptive basis model that extends the capabilities of existing neural networks to handle multivariate functional data and additional scalar variables. Our approach integrates a functional alignment module, enhancing the accuracy of subsequent analyses by reducing variability caused by misalignment. We demonstrate the effectiveness of our method on ECG data from the Copenhagen General Population Study and the results indicate that our model outperforms traditional convolutional neural networks. Additionally, we introduce the instantaneous contribution to the total probability for improved interpretability of the model's predictions.

Keywords: Deep Learning; Multivariate Functional Data; Time warping; Quasi-Periodic Functions.

II.1. INTRODUCTION

Functional data analysis (FDA, Ramsay & Silverman, 2005) is a powerful framework for understanding and interpreting data characterized by underlying smooth, continuous processes. From analyzing growth curves to deciphering brain imaging data, the applications of FDA span diverse scientific domains. Recent advancements in deep learning have transformed data analysis methodologies, demonstrating the capability to identify intricate patterns within complex datasets (Anbarasi et al., 2024; Archana & Jeevaraj, 2024; Li et al., 2023).

Functional data traditionally consists of samples at discrete time points of underlying random functions, typically one or more curves per individual subject in the dataset, measured on an interval. Multivariate functional data arises when multiple functions are measured simultaneously on each individual. Examples of such functions are 12-lead electrocardiogram (ECG) recordings (Gregg et al., 2008) that are typically sampled with a frequency of of 500 samples per second. ECGs are diagnostic tests that record the electrical activity of the heart over a period of time. This process usually involves attaching electrodes to the skin to monitor electrical changes that occur due to the heart muscle's depolarization pattern during each heartbeat. ECGs provide valuable information about the heart's rate and rhythm, as well as various cardiac abnormalities such as arrhythmias, ischemia, and myocardial infarction (heart attacks). It is a non-invasive and essential tool used by healthcare professionals in diagnosing and monitoring heart conditions.

In this article, we explore the possibility of using ECGs as a screening method for heart calcification using functional neural networks together with joint alignment of multivariate quasi-periodic functional data (Pham et al., 2023), and in addition, we provide a method for interpretability and explainability of the results.

Aligning functional data before analysis is important for several reasons. Measurements over time can be affected by temporal shifts or misalignment. Aligning the data ensures that similar events or features are compared at the same time point, which is essential for accurate analysis or prediction. Misaligned data can lead to incorrect conclusions because the same features may appear at different time points in different datasets, and even within the same dataset. Aligning functional data can enhance the statistical power of subsequent analyses by reducing variability caused by misalignment, leading to more accurate and reliable results. In functional data, specific features (e.g. peaks in an ECG) may correspond to salient events or patterns. Proper alignment ensures that these features are correctly matched, facilitating meaningful comparisons and interpretations. Misaligned data can introduce artifacts or distortions that compromise the integrity of the analysis. Aligning data also allows for a clearer and more intuitive interpretation of patterns and trends, ensuring that any observed differences or similarities are due to actual phenomena rather than misalignment artifacts.

While our application primarily targets ECG analysis, our method holds promise for broader applications in other domains involving time series, including data from health apps and wearable sensors, data from weather stations and environmental sensors, or utilities and energy grid data. Many of these domains contain quasi-periodic data, see (Pham et al., 2023). Quasi-periodic data arise from inherently periodic processes affected by noise, both in the temporal domain and in the amplitude.

In this paper, we use neural networks (NN) for the alignment of the data, as well as for

subsequent prediction. Neural networks excel at handling high-dimensional data, which is particularly useful for functional data. The ability to manage complexity allows NNs to model intricate, non-linear relationships effectively. Moreover, they have the flexibility to adapt to a wide range of data patterns and structures, providing robust modeling capabilities across diverse functional data scenarios. A significant strength of neural networks is their capacity for automatic feature extraction (Archana & Jeevaraj, 2024).

II.1.1. *Related work*

The use of deep neural networks for analyzing functional data and time series has grown, particularly with the introduction of convolutional neural networks (CNNs) and recurrent neural networks (RNNs) as highlighted by LeCun et al. (2015). These and many other methods use discretized functions as input.

ECGs have been analyzed using deep learning in numerous studies. Sannino and De Pietro (2018) proposed a fully connected NN to detect abnormal heartbeats, combining ECG heartbeats sampled uniformly at 50 sampling times, and four derived temporal features of each heartbeat. ECGs have also been analyzed using CNNs as in Uchiyama et al. (2022), where the signal from the individual ECG leads was converted to a grayscale image. More commonly, ECGs have been analyzed using a one-dimensional (temporal) CNN for predicting age and sex, see Attia, Friedman, et al. (2019), or for detecting atrial fibrillation, see Attia, Noseworthy, et al. (2019). An extensive overview was presented by Roopa and Harish (2017).

The above-mentioned NNs use discretized signals as inputs, while Thind et al. (2023) proposed a functional neural network for when the inputs are functions. Their methodology introduces functional weights, that are smooth functions of time. The relationship between functional weights, input functions, and neurons inside the neural network is represented by the formula $b_u = g\left(b_u + \int_{\mathcal{T}} \beta_u(t) f(t)\, dt\right)$, where b_u is the value of the neuron, g is a (usually) nonlinear activation function, $\mathcal{T}$ is the time interval under consideration, $\beta_u(t)$ is the functional weight, $f(t)$ is the univariate functional input, and u is the index of the neuron in the first hidden layer. The functional weights are represented by their basis expansion as $\beta_u(t) = \sum_{j=1}^{J} c_{uj}\phi_{uj}(t) = \boldsymbol{c}_u^T \boldsymbol{\phi}_u(t)$, where $\boldsymbol{\phi}_u(t) = (\phi_{u1}(t), \ldots, \phi_{uJ}(t))$ is a vector of pre-defined basis functions such as the B-splines or the Fourier basis functions, and $\boldsymbol{c}_u = (c_{u1}, \ldots, c_{uJ})$ is the vector of corresponding basis coefficients, that are learned by the network. With the addition of scalar covariates z_m, $m = 1, \ldots, M$, the neurons in the first hidden layer can be represented as

$$b_u = g\left(b_u + \sum_{m=1}^{M} w_{um} z_m + \sum_{p=1}^{P}\sum_{j=1}^{J} c_{upj} \int_{\mathcal{T}} \phi_{upj}(t) f_p(t)\, dt\right),$$

where P is the dimension of the multivariate input data, $f(t) = (f_1(t), \ldots, f_P(t)) \in \mathbb{R}^P$.

As opposed to pre-specifying the basis functions, Yao et al. (2021) introduced the Adaptive Basis Functional Neural Network (AdaFNN), a neural network that estimates the functional weights adaptively. AdaFNN utilizes a technique similar to the Network in Network approach (NiN) introduced in Lin et al. (2013). Unlike NiN which replaces linear convo-

lutions in Lin et al. (2013) by multilayer perceptrons, AdaFNN produces the first hidden layer (called the basis layer in AdaFNN) by implementing a basis micro network which takes a fixed time point t and outputs the value of the functional weight at this time point, $\beta(t)$. The neurons in the basis layer are then obtained by the integral $\int_{\mathcal{T}} \beta(t) f(t)\, dt$. In all the applications, the authors showcased the performance of AdaFNN on univariate functional data with no additional variables.

II.1.2. *Proposed approach*

In this paper, we propose a functional neural network, that uses the adaptive basis approach, similar to AdaFNN by Yao et al. (2021), but we extend the network to handle multivariate functional data, with the addition of scalar/tabular variables. Finally, we expand the model to account for misalignment in quasi-periodic data.

The paper is structured as follows. Section II.2 introduces the notation and extension of the AdaFNN model with a general architecture that can accommodate multivariate functional data and additional covariates. Additionally, this section describes the integration of another NiN, specifically a functional alignment module, an extension of DeepJAM (the Deep Joint Alignment of Multivariate functional data, Pham et al. (2023)). Finally, this section also presents an interpretability method: the Instantaneous Contribution to the total Probability (ICP). In Section II.3 we apply our method to 12-lead 10 second ECG data. To conclude, in Section II.4 we discuss the results of our method.

II.2. METHODS

This section details the proposed method. First, Section II.2.1 describes the notation of a standard, fully connected feed-forward neural networks, together with the algorithm for forward propagation. Section II.2.2 builds on the notation and presents the extension of the AdaFNN model for prediction, together with an interpretability tool, the ICP.

II.2.1. *Fully connected feed-forward neural network*

A neural network with L layers is a model consisting of an input layer $l = 0$, hidden layers $l = 1, \ldots, L-1$, and an output layer $l = L$, referred to by a superscript (l). Each layer has a dimension $d^{(l)}$, which denotes the $d^{(l)} + 1$ nodes in this layer, $l < L$. The nodes in each layer are labeled $0, 1, \ldots, d^{(l)}$, where the bias node (intercept) is labeled as 0. The bias node is set to have an output 1 and has no incoming signal. The nodes in each layer $l > 0$, have an incoming signal $\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)}$ from the previous layer, *activation function* $\sigma^{(l)}$, and output signal $\boldsymbol{x}^{(l)} = (1, \sigma^{(l)}(\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)}))^T$, where $\boldsymbol{W}^{(l)}$ is a $d^{(l)} \times (d^{(l-1)} + 1)$ dimensional matrix of weights (parameters) and $\boldsymbol{x}^{(0)}$ is the input including a bias node. We can collect all weight matrices to a single parameter $w = \{\boldsymbol{W}^{(1)}, \ldots, \boldsymbol{W}^{(L)}\}$. The NNs are usually depicted by a diagram as in Figure II.1. The output layer is then iteratively calculated from the previous layers and this process is termed the *forward propagation*, see Algorithm II.1.

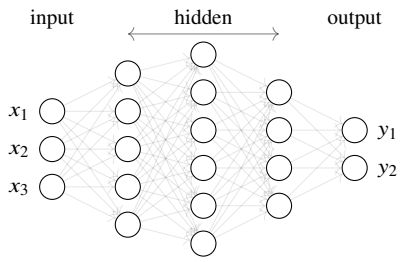


FIGURE II.1. *Example of an NN diagram, with an input layer with three nodes, three hidden layers with five, six, and four nodes, and an output layer with two nodes.*

Algorithm II.1 Forward propagation for one observation

Input: input data $\boldsymbol{x}$, number of layers L, weight matrices w, activation functions $\sigma^{(l)}$, $l = 1, \ldots, L$

Output: prediction $\widehat{\boldsymbol{y}}$

$\boldsymbol{x}^{(0)} \leftarrow \begin{pmatrix} 1 \\ \boldsymbol{x} \end{pmatrix}$

for $l = 1, \ldots, L$ **do**

$\quad \boldsymbol{x}^{(l)} = \begin{pmatrix} 1 \\ \sigma^{(l)}(\boldsymbol{W}^{(l)}\boldsymbol{x}^{(l-1)}) \end{pmatrix}$

end for

$\widehat{\boldsymbol{y}} \leftarrow \boldsymbol{x}^{(L)} = \sigma^{(L)}(\boldsymbol{W}^{(L)}\boldsymbol{x}^{(L-1)})$

II.2.2. *Adaptive basis model*

The standard fully connected feed-forward neural network takes scalar variables as input, and therefore, some adjustments are necessary to account for functional data as inputs. This is done by using a *basis layer*, which is the first hidden layer in the network, following the functional input. Let's denote the number of nodes in the basis layer as U. Each node b_u, $u = 1, \ldots, U$, is calculated as a projection of the input function f on some basis function β_u as

$$b_u = \int_{\mathcal{T}} \beta_u(t) f(t)\, dt,$$

which is in practice approximated using numerical integration. The basis functions are parametrized by an NiN model that takes a scalar input t and outputs the value of the basis function at t, $\beta_u(t) = NiN_u(t)$. Note that the output signals of the nodes in the basis layer are scalars, hence after the basis layer, the rest of the network has the same structure as the standard feed-forward neural network. An example of the model architecture is represented in Figure II.2. In this example, there is one input function f and four nodes in the basis layer. Algorithm II.2 shows the forward propagation in the adaptive basis model.

In the case of multiple functional inputs or multivariate functional data, we propose to calculate a separate basis layer for each functional input, and then we concatenate these basis layers and additional scalar covariates to obtain an input layer for the subsequent

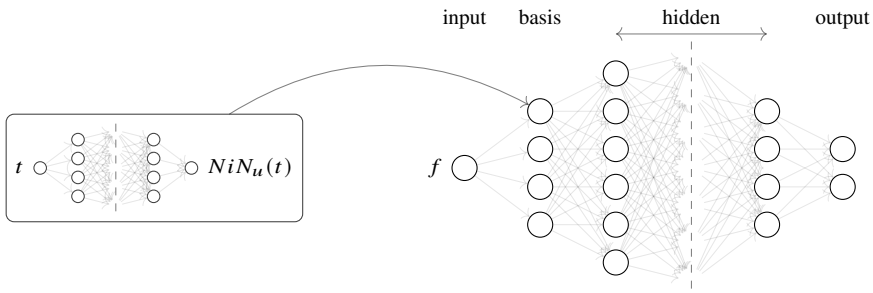


FIGURE II.2. *Example of an architecture of a functional neural network with one input function and four basis layer nodes.*

neural network. An example of such a network architecture is shown in Figure II.3, with the forward propagation represented in Algorithm II.3.

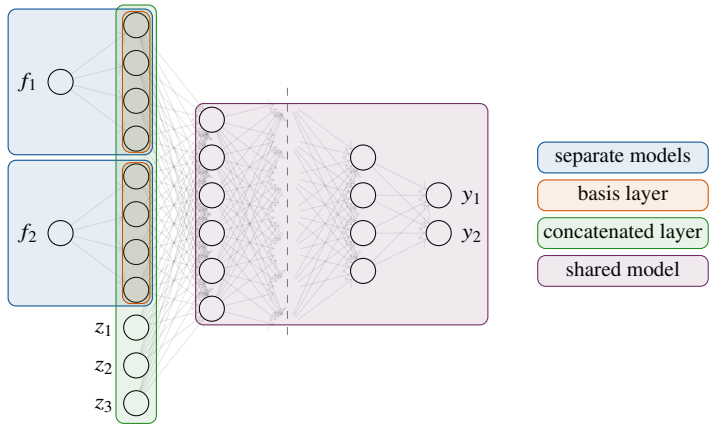


FIGURE II.3. *Example of a neural network with multivariate functional data and adaptive basis. Here, there are two functional inputs f_1 and f_2, three scalar covariates z_1, z_2 and z_3, and two outputs y_1 and y_2.*

Joint alignment of multivariate functional data - JAM_AB. Additionally, functional data is often quasi-periodic, which means that the data exhibits patterns that are almost, but not exactly periodic in time. Examples of such data are the tidal rise and fall of sea levels and seasonal temperatures, and in medicine, we can think of the ECG data or the menstrual cycle. We assume that the underlying processes generating the data are periodic, but due to noise stemming from other external processes, we only observe distorted signals, both in amplitude and time domains.

It is crucial to align the data in their temporal domain to facilitate the identification of trends and patterns over time. Alignment not only helps reduce noise and anomalies that can arise from misaligned data but also improves the validity of the results of standard statistical methods.

Various architectures can accommodate multivariate quasi-periodic functional data and

Algorithm II.2 Forward propagation for one observation in the adaptive basis model

Input: sampling times $\boldsymbol{t}$, data $f(t)$, number of basis nodes U, number of layers in the NiNs B_u, number of layers in the subsequent NN L, weight matrices in the NiNs w_u, weight matrices in the subsequent NN w, activation functions $\sigma_u^{(b_u)}$ and $\sigma^{(l)}$, $u = 1, \ldots, U$, $b_u = 1, \ldots, B_u$, $l = 1, \ldots, L$, time interval $\mathcal{T}$

Output: prediction $\widehat{\boldsymbol{y}}$

for $u = 1, \ldots, U$ **do**
  for t **in** $\boldsymbol{t}$ **do**
    Algorithm II.1 with
      Input: data t, number of layers B_u, weight matrices w_u, activation functions $\sigma_u^{(b_u)}$, $b_u = 1, \ldots, B_u$
      Output: weight functions β_u
  end for
  $b_u \leftarrow \int_{\mathcal{T}} \beta_u(t) f(t)$
end for
$\boldsymbol{x} \leftarrow (b_1, \ldots, b_U)^T$
Algorithm II.1 with
  Input: data $\boldsymbol{x}$, number of layers L, weight matrices w, activation functions $\sigma^{(l)}$ $l = 1, \ldots, L$
  Output: prediction $\widehat{\boldsymbol{y}}$

account for additional variables at the same time. A general approach used in this paper is as follows. First, we use the DeepJAM warping module to jointly align multivariate quasi-periodic functions (Pham et al., 2023). The DeepJAM module is based on an unsupervised algorithm that aligns data based on a Fisher-Rao loss and additionally provides an adaptive estimation of a common and subject-specific template. Second, we use the adaptive basis layer approach, with a micro neural network per node. Then, we concatenate this basis layer with additional tabular covariates to propagate this concatenated layer through a standard fully-connected NN. At training time, the parameters of the warping module and the parameters of the subsequent neural network are learned jointly, which is achieved by utilizing a loss that is a weighted average of the Fisher-Rao loss of the DeepJAM module and the classification stemming from the subsequent fully-connected NN. Since the Fisher-Rao loss and the classification loss can vary in magnitude, we adjust the scale of each loss during every epoch of model training to ensure that they are of comparable magnitude before computing the weighted average. For an example of such a network, see Figure II.4. We call the full model combining the Joint Alignment of Multivariate functional data and the Adaptive Basis layer as JAM_AB.

Instantaneous contribution to the total probability. To better understand the NN model predictions for classification tasks, we introduce the Instantaneous Contribution to the total Probability (ICP). This technique can help with the interpretability of the model and highlight the key areas of the input signal that were critical for the model output.

We define the ICP in the following way. Let $\pi(t)$ be the output of an NN model with an adaptive basis layer, where the input is a multivariate function $f(t) = (f_1(t), \ldots, f_P(t)) \in \mathbb{R}^P$. We can represent it as some complicated function of the input and the weight functions

Algorithm II.3 Forward propagation for one observation in the adaptive basis model with multivariate functional data

Input: sampling times $\boldsymbol{t}$, number of functions P, data $f_p(t)$, number of basis nodes U, number of layers in the NiNs B_{pu}, number of layers in the subsequent NN L, weight matrices in the NiNs w_{pu}, weight matrices in the subsequent NN w, activation functions $\sigma_{pu}^{(b_{pu})}$ and $\sigma^{(l)}$, $p = 1, \ldots, P$, $u = 1, \ldots, U$, $b_{pu} = 1, \ldots, B_{pu}$, $l = 1, \ldots, L$, time interval $\mathcal{T}$

Output: prediction $\widehat{\boldsymbol{y}}$

for $p = 1, \ldots, P$ **do**
  for $u = 1, \ldots, U$ **do**
    for t **in** $\boldsymbol{t}$ **do**
      Algorithm II.1 with
        Input: data t, number of layers B_{pu}, weight matrices w_{pu}, activation functions $\sigma_{pu}^{(b_{pu})}$, $b_{pu} = 1, \ldots, B_{pu}$
        Output: weight functions β_{pu}
    end for
    $b_{pu} \leftarrow \int_{\mathcal{T}} \beta_{pu}(t) f_p(t)$
  end for
end for
$\boldsymbol{x} \leftarrow (b_{11}, \ldots, b_{1U}, \ldots, b_{p1}, \ldots, b_{PU})^T$
Algorithm II.1 with
  Input: data $\boldsymbol{x}$, number of layers L, weight matrices w, activation functions $\sigma^{(l)}$ $l = 1, \ldots, L$
  Output: prediction $\widehat{\boldsymbol{y}}$

as

$$\pi(t) = h\left(\int_{t_1}^{t} \beta_{11}(s) f_1(s)\, ds, \ldots, \int_{t_1}^{t} \beta_{PU_P}(s) f_P(s)\, ds, \theta\right),$$

where h is a function representing the NN functional, and β_{pu}, for $p = 1, \ldots, P$ and $u = 1, \ldots, U_p$, are the functional weights included in the model. U_p is the number of basis nodes corresponding to the p-th element of the multivariate function f. The output of the NN, $\pi(t_2)$, can be decomposed as

$$\pi(t_2) = \pi(t_1) + \int_{t_1}^{t_2} \pi'(t)\, dt = \int_{t_1}^{t_2} \left(\frac{\pi(t_1)}{t_2 - t_1} + \pi'(t)\right) dt,$$

where the last integrand can be interpreted the instantaneous contribution to the total probability.

For a classification NN with a binary outcome, the output is usually a probability of the positive label. We can then interpret the ICP in the following way. First, we extract the ICP for each observation, and then we calculate the cross-sectional means for each class. Comparing the cross-sectional means of the two classes, we can on average determine which regions of the data that are the most different, and which areas contribute positively to the predicted probability.

In the context of ECG, incorporating interpretability techniques can enhance model

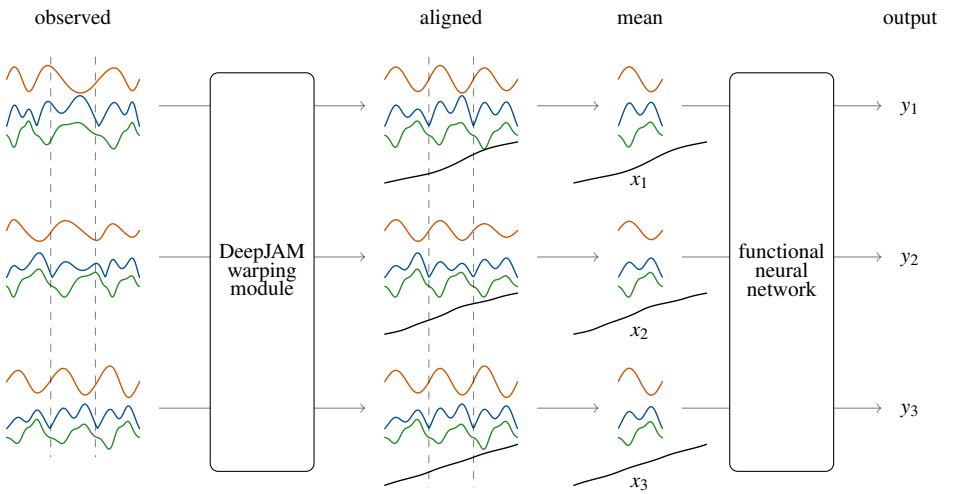


FIGURE II.4. *Example of a neural network with joint alignment of multivariate quasi-periodic functional data and adaptive basis (JAM_AB). There are 3 observed multivariate functions on the left side of the diagram that are passed through a DeepJAM warping module, which outputs aligned multivariate functions together with the corresponding warping function represented in black. Subsequently, a multivariate subject-specific mean is propagated through a functional neural network together with the warping functions and additional covariates x_i, obtaining outputs y_i. The gray dashed lines split the data into three periods.*

transparency. When a clear association exists between a specific heart disease and ECG measurements, interpretability methods, including ICP, facilitate validation. They allow us to verify whether the neural network's output is influenced by relevant regions within the ECG data. Conversely, in cases where no established association exists between the ECG and the studied heart condition, ICP can pinpoint critical areas within the ECG measurements.

II.3. APPLICATION

We demonstrate the use of our method on 12-lead 10 second ECG recordings. The study population comprised individuals from the Copenhagen General Population Study, who were randomly selected from the general population in Copenhagen, Denmark (Fuchs et al., 2023; Kühl et al., 2019). All ECGs were obtained with participants at rest and in a supine position, prior to undergoing a cardiac computed tomography scan. Written informed consent was secured from all participants, and the study received approval from the local ethics committee (H-KF-01-144/01).

A common characteristic of ECGs is that the number of heartbeats within a 10 second interval varies among participants. With some exceptions, standard NN methods can only handle input data with fixed length. We adapted the JAM_AB algorithm described in Section II.2.2 to account for data with varying lengths. Knowing the number of heartbeats for each participant, the observed data can be easily scaled, so that we can use the methods for quasi-periodic data as defined in Pham et al. (2023). Figure II.5 shows an example

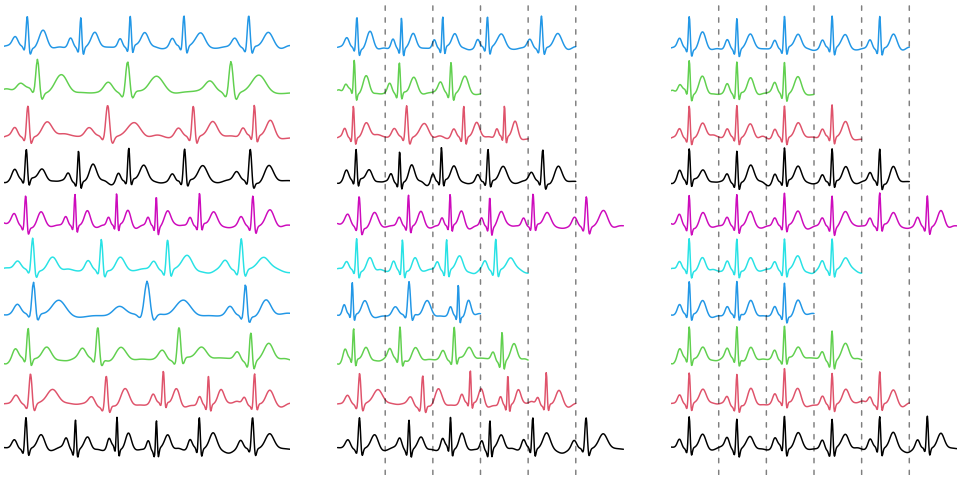


FIGURE II.5. *Example of univariate data with signals of varying length, where each row represents the recorded ECG of one participant. The observed quasi-periodic data is shown in column 1, their scaled versions in column 2, and the underlying periodic data with amplitude variability is represented in column 3. The gray vertical dashed lines represent isometric periods.*

of such data with varying number of heartbeats, namely the observed data (column 1), scaled data (column 2), and the aligned data (column 3). In this example, the maximum number of heartbeats on the interval $\mathcal{T}$, which can without loss of generality be considered $[0, 1]$, is $K = 6$. Let H_i denote the number of heartbeats for each observation. The signals are then linearly scaled so that they lie in intervals $[0, H_i/K]$, yielding the scaled data. This scaled data then enters the DeepJAM module to produce the aligned data. In the application, we examine two scenarios. First, we limit the analysis to a fixed number of heartbeats per participant, constrained by the minimum number of heartbeats, which is six in our case. Second, we allow for a varying number of heartbeats per subject, resulting in a functional input of 12 heartbeats in our datasets, with the input data represented as shown in column 2 of Figure II.5.

As an outcome, we use the calcification score (CAC) which is calculated using a specialized CT scan that measures the amount of calcium in the coronary arteries. This score helps predict the risk of cardiovascular diseases, such as heart attacks and strokes. We used the dichotomized version of the CAC, specifically

$$y = \begin{cases} 1 & CAC > 0, \\ 0 & CAC = 0. \end{cases}$$

We compare our method to four other architectures. The first architecture is the convolutional neural network (CNN, LeCun et al., 1989), because it has been one of the standards for time series analysis, and especially ECG analysis (Roopa & Harish, 2017). In this case, we did not align the data before using the functions as an input. The second architecture builds on CNNs, and uses the DeepJAM warping module first to align the quasi-periodic data. The third considered architecture is a functional neural network with an adaptive basis without the alignment module. All these models consider both the ECG measurements

as functional predictors and additional covariates such as age and sex. Finally, we also compare our method to a fully connected neural network without the ECG predictors.

For a fair comparison of the various model architectures, we used Bayesian optimization (Snoek et al., 2015) for hyperparameter tuning. For the convolutional layers, the tuned hyperparameters were the number of layers, filters, and kernel size. For fully connected parts of the CNN and the functional neural network, the tuned parameters were the number of layers and units.

Due to computational constraints, all the convolutional layers share the number of filters and size of the kernels, all fully connected layers share the number of units, and all basis nodes in the adaptive basis share the number of layers and the number of units in each layer. In the DeepJAM warping module, all the layers share the number of filters and the kernel size. In addition to these parameters, we tuned the initial learning rate used in the Adam optimizer (Kingma & Ba, 2015).

Table II.1 shows the performance of the various architectures. The functional neural network with an adaptive basis and alignment module with inputs of varying number of heartbeats performs the best regarding the binary cross-entropy, the mean squared error, and AUC. Figure II.6 illustrates the ICP of this model as well as the aligned data. The

architecture	alignment	heartbeats	binary cross-entropy	MSE	AUC
AB	yes	varying	0.552	0.187	0.791
CNN	yes	varying	0.555	0.188	0.791
AB	no	fixed	0.555	0.188	0.789
AB	yes	fixed	0.555	0.188	0.787
CNN	no	varying	0.555	0.188	0.790
AB	no	varying	0.555	0.188	0.790
FNN			0.558	0.189	0.786
CNN	no	fixed	0.560	0.189	0.786
CNN	yes	fixed	0.561	0.190	0.784

TABLE II.1. *Performance of the neural network models. The architecture represents the type of network, namely the functional neural network with an adaptive basis layer (AB), convolutional neural network (CNN), and fully connected neural network (FNN). The models are sorted based on the binary cross-entropy (lowest to highest), which was used as the loss function. Additionally, we present the mean squared error (MSE), and the area under the ROC curve (AUC). The heartbeats column informs about whether only a fixed number of heartbeats was considered, or whether all the heartbeats per participant were considered, leading to a varying number of heartbeats.*

largest difference between the two classes are located at the beginning and end of the T wave of the ECG.

II.4. DISCUSSION

This study demonstrates the potential of functional neural networks with adaptive basis layers for analyzing multivariate quasi-periodic functional data, specifically focusing on 12-lead electrocardiogram (ECG) recordings. Our proposed method integrates a functional alignment module, which enhances the accuracy of subsequent analyses by reducing vari-

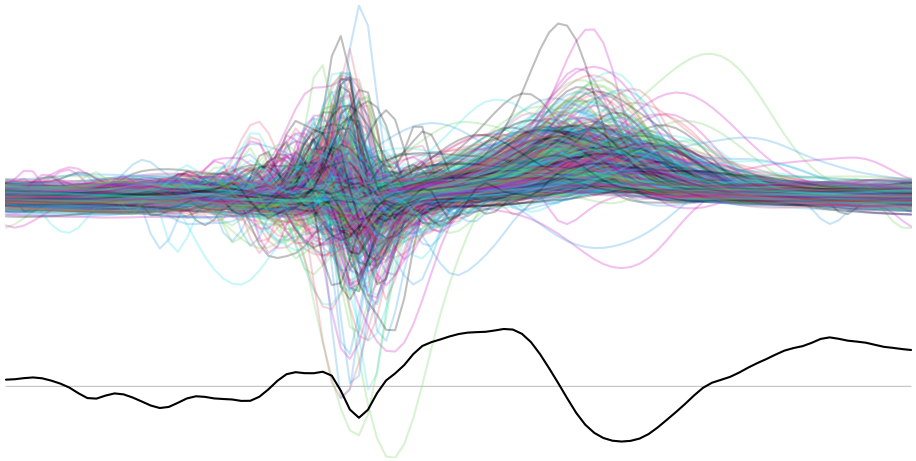


FIGURE II.6. *Aligned heartbeats and ICP difference. The top shows the aligned heartbeats (lead V5). The bottom shows the difference between the average ICP (black) for the two classes,* $y = 1$ *and* $y = 0$*. The difference is amplified for the sake of illustration and the horizontal gray line is added for reference and represents the difference of zero.*

ability caused by misalignment. This approach not only improves the interpretability of the results but also ensures that the observed patterns are due to actual phenomena rather than artifacts introduced by misalignment.

The results from our experiments indicate that the functional neural network with an adaptive basis and alignment module outperforms traditional convolutional neural networks in terms of cross-entropy, mean squared error, and AUC.

One of the key contributions of this work is the introduction of instantaneous contribution to the total probability for improved interpretability (ICP) of the model's predictions. The ICP allows us to identify critical areas of the input signal that influence the model's output, thereby providing valuable insight into the underlying mechanisms of the data-generating process.

Despite the promising results, there are several areas for future research, First, the computational complexity of the proposed model, particularly the alignment module, can be a limiting factor for large-scale applications. Optimizing the computational efficiency of the model will be essential for its practical deployment. Second, further validation on diverse datasets from different domains will be necessary to establish its generalizability.

In conclusion, this study presents an approach for the analysis of multivariate quasi-periodic functional data using deep learning techniques. The integration of functional alignment and adaptive basis layers in neural networks offers a powerful framework for extracting meaningful patterns from complex datasets.

BIBLIOGRAPHY

ANBARASI, J., KUMARI, R., GANESH, M., & AGRAWAL, R. (2024). Translational connectomics: Overview of machine learning in macroscale connectomics for clinical insights. *BMC Neurology*, *24*(1), 364. https://doi.org/10.1186/s12883-024-03864-0

ARCHANA, R., & JEEVARAJ, P. S. E. (2024). Deep learning models for digital image processing: A review. *Artificial Intelligence Review*, *57*(1), 11. https://doi.org/10.1007/s10462-023-10631-z

ATTIA, Z. I., FRIEDMAN, P. A., NOSEWORTHY, P. A., LOPEZ-JIMENEZ, F., LADEWIG, D. J., SATAM, G., PELLIKKA, P. A., MUNGER, T. M., ASIRVATHAM, S. J., SCOTT, C. G., CARTER, R. E., & KAPA, S. (2019). Age and sex estimation using artificial intelligence from standard 12-lead ECGs. *Circulation: Arrhythmia and Electrophysiology*, *12*(9). https://doi.org/10.1161/CIRCEP.119.007284

ATTIA, Z. I., NOSEWORTHY, P. A., LOPEZ-JIMENEZ, F., ASIRVATHAM, S. J., DESHMUKH, A. J., GERSH, B. J., CARTER, R. E., YAO, X., RABINSTEIN, A. A., ERICKSON, B. J., KAPA, S., & FRIEDMAN, P. A. (2019). An artificial intelligence-enabled ECG algorithm for the identification of patients with atrial fibrillation during sinus rhythm: A retrospective analysis of outcome prediction. *The Lancet*, *394*(10201), 861–867. https://doi.org/10.1016/S0140-6736(19)31721-0

GREGG, R. E., ZHOU, S. H., LINDAUER, J. M., HELFENBEIN, E. D., & GIULIANO, K. K. (2008). What is inside the electrocardiograph? *Journal of Electrocardiology*, *41*(1), 8–14. https://doi.org/10.1016/j.jelectrocard.2007.08.059

KINGMA, D. P., & BA, J. (2015). Adam: A method for stochastic optimization. *Proceedings of the 3rd International Conference on Learning Representations*. https://doi.org/10.48550/ARXIV.1412.6980

LECUN, Y., BENGIO, Y., & HINTON, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

LECUN, Y., BOSER, B., DENKER, J., HENDERSON, D., HOWARD, R., HUBBARD, W., & JACKEL, L. (1989). Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, *2*.

LI, M., JIANG, Y., ZHANG, Y., & ZHU, H. (2023). Medical image analysis using deep learning algorithms. *Frontiers in Public Health*, *11*, 1273253. https://doi.org/10.3389/fpubh.2023.1273253

LIN, M., CHEN, Q., & YAN, S. (2013). Network in network. https://doi.org/10.48550/ARXIV.1312.4400

PHAM, V. T., NIELSEN, J. B., KOFOED, K. F., KÜHL, J. T., & JENSEN, A. K. (2023). Joint alignment of multivariate quasi-periodic functional data using deep learning. https://doi.org/10.48550/ARXIV.2312.09422

RAMSAY, J. O., & SILVERMAN, B. W. (2005). *Functional data analysis* (2nd). Springer. https://doi.org/10.1007/b98888

ROOPA, C., & HARISH, B. (2017). A survey on various machine learning approaches for ECG analysis. *International Journal of Computer Applications*, *163*(9), 25–33. https://doi.org/10.5120/ijca2017913737

SANNINO, G., & DE PIETRO, G. (2018). A deep learning approach for ECG-based heartbeat classification for arrhythmia detection. *Future Generation Computer Systems*, *86*, 446–455. https://doi.org/10.1016/j.future.2018.03.057

SNOEK, J., RIPPEL, O., SWERSKY, K., KIROS, R., SATISH, N., SUNDARAM, N., PATWARY, M., PRABHAT, M., & ADAMS, R. (2015). Scalable bayesian optimization using deep neural networks, 2171–2180.

THIND, B., MULTANI, K., & CAO, J. (2023). Deep learning with functional inputs. *Journal of Computational and Graphical Statistics*, *32*(1), 171–180. https://doi.org/10.1080/10618600.2022.2097914

UCHIYAMA, R., OKADA, Y., KAKIZAKI, R., & TOMIOKA, S. (2022). End-to-end convolutional neural network model to detect and localize myocardial infarction using 12-lead ECG images without preprocessing. *Bioengineering*, *9*(9), 430. https://doi.org/10.3390/bioengineering9090430

YAO, J., MUELLER, J., & WANG, J.-L. (2021). Deep learning for functional data analysis with adaptive basis layers. *139*, 11898–11908.

Manuscript III

DEEP LEARNING OF EVENT RISK IN CONTINUOUS TIME WITH COMPETING RISKS

VI THANH PHAM
Section of Biostatistics, University of Copenhagen, Denmark

ANDERS MUNCH
Section of Biostatistics, University of Copenhagen, Denmark

THOMAS ALEXANDER GERDS
Section of Biostatistics, University of Copenhagen, Denmark

ABSTRACT

This paper proposes a novel approach using neural networks to predict time-to-event outcomes in continuous time with competing risks. The method leverages two alternative approximations of the hazard functions, one based on piecewise constant hazard functions, and the other based on the Weibull distribution. These models can handle various types of input data, including functional, and image data, and can simultaneously predict the risk of the event of interest, competing risks, and the probability of censoring.

The proposed continuous-time neural network approach offers several advantages over traditional methods, such as not requiring pre-specified interactions or non-linear terms and being capable of estimating complex functional relationships. However, they also come with challenges like overfitting and computational expense, which can be mitigated through techniques such as weight decay and increasing sample size.

The application of these models is demonstrated using data from a study on Prostate-Specific Antigen (PSA) kinetics in patients with localized prostate cancer. The results show that the neural network models perform well in terms of time-dependent AUC and time-dependent Brier score, particularly for long-term predictions.

Keywords: Deep Learning; Time-to-event outcome; Competing risks.

III.1. INTRODUCTION

Time-to-event analysis focuses on understanding the time until a specific event occurs. In medical applications, this event could be anything related to health, such as disease recurrence, remission or death. Survival analysis is characterized by censored data, where only partial information about the event time is observed.

The fundamental concepts in survival analysis include the survival function, hazard function, and cumulative hazard function. The survival function, $S(t)$, represents the probability of the event occurring at or after time t. The hazard function, $h(t)$, describes the instantaneous rate at which events occur, given that the individual has survived until time t. The cumulative hazard function, $H(t)$, accumulates the hazard over time.

Competing risks and multi-state models extend traditional survival analysis to scenarios where individuals may experience multiple types of events or transition through several states over time (Andersen & Keiding, 2002; Andersen et al., 2002).

We propose using neural networks for the prediction of time-to-event outcomes in continuous time with competing risks. Some advantages of using neural networks is the possibility of using complex inputs, such as functional data and image data. Our implementation uses the negative log-likelihood of the observed censored data as the loss function, hence this extension can be easily incorporated to existing neural network architectures. In addition, our neural network implementation can be used for simultaneous prediction of the risk of the event of interest, the competing risks, as well as the probability of censoring.

An advantage of using neural networks compared to more traditional models, such as Cox regression, is that we do not need to pre-specify interactions between variables nor non-linear terms of continuous variables, because with a complex enough architecture, the neural network should be able to estimate any functional relationship (Hastie et al., 2009, chap. 11). However, increasing the complexity of neural networks comes at a cost, such as overfitting and computational expense. Overfitting neural networks can be mitigated by introducing constraints on the model parameters and/or using weight decay, which is comparable to LASSO or ridge regression.

III.1.1. *Related Work*

Recently, there has been a surge of machine learning and deep learning methods for survival data. In the deep learning world, some architectures try to imitate Cox regression by using the Cox partial likelihood as the loss function (Ching et al., 2018), some use the event time as one of the input covariates (E. Biganzoli et al., 1998; E. M. Biganzoli et al., 2006), and some use discrete time scales with multiple outputs to predict the risk of event in the pre-specified time intervals (Wright et al., 2021). Wiegrebe et al. (2023) present an extensive review of existing deep learning methods for survival analysis. Our aim is to analyze time-to-event data in continuous time and in the presence of competing risks, and in addition, we want to incorporate high-dimensional data.

Additionally, due to the complexity of interpreting hazard rates, our goal is to predict the cause-specific risk. To achieve this, we have worked out two alternative approximations of the cause-specific and censoring-specific hazard function. The first approximation uses Weibull parametrizations, as demonstrated Bennis et al. (2020, 2021), Nagpal et al.

(2021), and Paolucci et al. (2023). Notably, only Nagpal et al. (2021) address competing risks. The second approximation uses the piecewise constant hazard functions. For both approximations of the hazard functions, we then derive the cause-specific cumulative incidence functions based on the hazard functions.

III.1.2. *Proposed approach*

The proposed method introduces a class of neural networks, that can simultaneously predict cause-specific risks, as well as the censoring distribution. We introduce "parametric" neural network models, which assume the Weibull distribution of the underlying generating processes, and output the parameters (shape and scale) of the distribution, and "non-parametric" models, which parameterize the (cause-specific) hazard functions directly.

III.2. METHOD

Let $(\widetilde{T}, \Delta, \widetilde{D}, \boldsymbol{X})$ be right-censored data, where $\widetilde{T} = \min\{T, C\}$ is the observed time, the minimum of the event time T and the censoring time C, Δ is the censoring indicator, $\Delta = \mathbb{1}\,(T \leq C)$, $\widetilde{D} = \Delta D \in \{0, 1, \ldots, J\}$ is the censored event type, where $D \in \{1, \ldots, J\}$ represents one of the J events, and 0 represents censoring, and finally, $\boldsymbol{X}$ is the set of scalar and functional covariates. Further, let $S(t \mid \boldsymbol{x}) = \mathbb{P}(T > t \mid \boldsymbol{X} = \boldsymbol{x})$ be the conditional survival function of the uncensored event time, and let $\widetilde{S}(t \mid \boldsymbol{x}) = \mathbb{P}(\widetilde{T} > t \mid \boldsymbol{X} = \boldsymbol{x})$ be the conditional survival function of the observed time. Let $F_j(t \mid \boldsymbol{x}) = \mathbb{P}(T \leq t, D = j \mid \boldsymbol{X} = \boldsymbol{x})$, $j = 1, \ldots, J$ be the conditional cause-specific distribution functions in the uncensored setting, and $\widetilde{F}_j(t \mid \boldsymbol{x}) = \mathbb{P}(\widetilde{T} \leq t, \widetilde{D} = j \mid \boldsymbol{X} = \boldsymbol{x})$, $j = 0, \ldots, J$ be the conditional cause- and censoring-specific distribution functions in the censored setting. Further, let $G(t \mid \boldsymbol{x}) = \mathbb{P}(C \leq t \mid \boldsymbol{X} = \boldsymbol{x})$ be the conditional distribution function of the censoring time, and let $S_0(t \mid \boldsymbol{x}) = \mathbb{P}(C > t \mid \boldsymbol{X} = \boldsymbol{x})$ be the conditional survival function of the censoring time. Furthermore, we assume that the event time T and the censoring time C are conditionally independent given $\boldsymbol{X}$, that is $T \perp C \mid \boldsymbol{X}$.

Under this assumption, we have the following relationship between the cause-specific hazard functions in the uncensored and censored settings. For an event of cause j the hazard functions h_j and $\tilde{h}_j$ are equal:

$$h_j(t \mid \boldsymbol{x}) = \frac{dF_j(t \mid \boldsymbol{x})/dt}{S(t- \mid \boldsymbol{x})} = \frac{d\widetilde{F}_j(t \mid \boldsymbol{x})/dt}{\widetilde{S}(t- \mid \boldsymbol{x})} = \widetilde{h}_j(t \mid \boldsymbol{x}). \tag{III.1}$$

Similarly, the hazard rate for censoring is

$$h_0(t \mid \boldsymbol{x}) = \frac{dG(t \mid \boldsymbol{x})/dt}{S_0(t- \mid \boldsymbol{x})} = \frac{d\widetilde{F}_0(t \mid \boldsymbol{x})/dt}{\widetilde{S}(t- \mid \boldsymbol{x})} = \widetilde{h}_0(t \mid \boldsymbol{x}). \tag{III.2}$$

In this paper, we explore two approaches for the neural network architecture to estimate the hazard rates. One approach, we call it the "parametric" approach, approximates the unknown cause-specific hazard functions and the censoring hazard function with Weibull parametrization (Weibull, 1951) and uses the fully connected neural network (Abu-Mostafa et al., 2012) to estimate the parameters of this Weibull distribution. The

second approach, we call it the "non-parametric" approach, uses the convolutional neural network (LeCun et al., 2015) to approximate the cause- and censoring-specific hazard functions by piecewise constant hazard functions. Both approaches use the negative partial log-likelihood of the observed data to simultaneously estimate the hazard functions of the competing events and the censoring.

Loss function. Given observed data $\left(\widetilde{T}, \Delta, \widetilde{D}, \boldsymbol{X}\right)$, the likelihood is defined as

$$\begin{aligned}
\mathcal{L}\left(\widetilde{t}, \widetilde{d}, \boldsymbol{x}; \boldsymbol{h}\right) &= \widetilde{S}\left(\widetilde{t}- \mid \boldsymbol{x}\right) \prod_{j=0}^{J} \left(h_j\left(\widetilde{t} \mid \boldsymbol{x}\right)^{1\{\widetilde{d}=j\}}\right) \mu(\boldsymbol{x}) \\
&= \exp^{-\int_0^{\widetilde{t}-} \sum_{j=0}^{J} h_j(s|\boldsymbol{x})\, ds} \prod_{j=0}^{J} \left(h_j\left(\widetilde{t} \mid \boldsymbol{x}\right)^{1\{\widetilde{d}=j\}}\right) \mu(\boldsymbol{x}) \\
&= \prod_{j=0}^{J} \left(\exp^{-\int_0^{\widetilde{t}-} h_j(s|\boldsymbol{x})\, ds}\right) \prod_{j=0}^{J} \left(h_j\left(\widetilde{t} \mid \boldsymbol{x}\right)^{1\{\widetilde{d}=j\}}\right) \mu(\boldsymbol{x}) \\
&= \prod_{j=0}^{J} \left(\exp^{-\int_0^{\widetilde{t}-} h_j(s|\boldsymbol{x})\, ds} h_j\left(\widetilde{t} \mid \boldsymbol{x}\right)^{1\{\widetilde{d}=j\}}\right) \mu(\boldsymbol{x})
\end{aligned}$$

where $\boldsymbol{h} = (h_0, h_1, \ldots, h_J)$ and $\mu(\boldsymbol{x}) = \mathbb{P}(\boldsymbol{X} \in dx)$. This yields the following log-likelihood

$$\ell\left(\widetilde{t}, \widetilde{d}, \boldsymbol{x}; \boldsymbol{h}\right) = \sum_{j=0}^{J} \left(-\int_0^{\widetilde{t}-} h_j(s \mid \boldsymbol{x})\, ds + 1_{\left\{\widetilde{d}=j\right\}} \log\left(h_j\left(\widetilde{t} \mid \boldsymbol{x}\right)\right)\right) + \log\left(\mu(\boldsymbol{x})\right),$$

and the estimator of the hazard functions is obtained by solving

$$\boldsymbol{h}^* = \operatorname{argmin}_{\boldsymbol{h}} \sum_{i=1}^{N} \sum_{j=0}^{J} \left(\int_0^{\widetilde{T}_i-} h_j(s \mid \boldsymbol{x}_i)\, ds - 1_{\left\{\widetilde{d}_i=j\right\}} \log\left(h_j\left(\widetilde{T}_i \mid \boldsymbol{x}_i\right)\right)\right). \tag{III.3}$$

Cause-specific risk. Using the negative partial log-likelihood in Equation (III.3), we obtain the cause- and censoring-specific hazard functions, and thus the cause- and censoring-specific distribution functions of observed time, $\widetilde{F}_j(t \mid \boldsymbol{x})$, $j = 0, \ldots, J$. To calculate the Brier score (BS), we need the cause-specific distribution functions of event time, $F_j(t \mid \boldsymbol{x})$, $j = 1, \ldots, J$, and the survival function of censoring time, $S_0(t \mid \boldsymbol{x})$. We obtain the survival function of the censoring time as $S_0(t \mid \boldsymbol{x}) = \exp(-H_0(t \mid \boldsymbol{x}))$, the event-free survival function and the cause-specific cumulative distribution function as

$$S(t \mid \boldsymbol{x}) = \exp\left(-\sum_{j=1}^{J} H_j(t \mid \boldsymbol{x})\right) \quad \text{and} \quad F_j(t \mid \boldsymbol{x}) = \int_0^t S(s \mid x) h_j(s \mid x)\, ds, j = 1, \ldots, J,$$

where H_j, $j = 0, \ldots, J$, are the cumulative hazard functions.

III.2.1. *"Parametric" neural network*

The first approach, the "parametric" neural network, assumes the Weibull distribution for the censoring and all events and outputs the shape and scale of these distributions. The shape $k > 0$ and scale $\lambda > 0$ parameters define the hazard function of a Weibull distribution

$$h(t; k, \lambda) = \frac{k}{\lambda}\left(\frac{t}{\lambda}\right)^{k-1}, \tag{III.4}$$

and the cumulative hazard function of a Weibull distribution

$$H(t; k, \lambda) = \int_0^t h(t; k, \lambda) = \int_0^t \frac{k}{\lambda}\left(\frac{s}{\lambda}\right)^{k-1} ds = \left[\left(\frac{s}{\lambda}\right)^k\right]_0^t = \left(\frac{t}{\lambda}\right)^k.$$

The parameters k_i and λ_i can depend on the subject i through the covariates, and we use a neural network to describe this relationship. Let η represent the neural network functional, such that

$$\begin{pmatrix} k_{i0} & k_{i1} & \cdots & k_{iJ} \\ \lambda_{i0} & \lambda_{i1} & \cdots & \lambda_{iJ} \end{pmatrix} = \eta(\boldsymbol{x}_i; \boldsymbol{\theta}),$$

where $\boldsymbol{\theta}$ is the set of all parameters/weights of the neural network model, and k_{ij} and λ_{ij} are the shape and scale parameters of the distributions, respectively, for the i-th observation and the j-th cause, $j = 0, \ldots, J$. Thus, the cause- and censoring-specific hazard functions are functions of time t, covariates $\boldsymbol{x}_i$, and parameters $\boldsymbol{\theta}$ as follows

$$h_j(t \mid \boldsymbol{x}_i) = \frac{k_{ij}}{\lambda_{ij}}\left(\frac{t}{\lambda_{ij}}\right)^{k_{ij}-1} = \frac{\eta(\boldsymbol{x}_i; \boldsymbol{\theta})_{1j}}{\eta(\boldsymbol{x}_i; \boldsymbol{\theta})_{2j}}\left(\frac{t}{\eta(\boldsymbol{x}_i; \boldsymbol{\theta})_{2j}}\right)^{\eta(\boldsymbol{x}_i;\boldsymbol{\theta})_{1j}-1}.$$

The corresponding cumulative hazard functions have the form

$$H_j(t \mid \boldsymbol{x}_i) = \left(\frac{t}{\lambda_{ij}}\right)^{k_{ij}-1} = \left(\frac{t}{\eta(\boldsymbol{x}_i; \boldsymbol{\theta})_{2j}}\right)^{\eta(\boldsymbol{x}_i;\boldsymbol{\theta})_{1j}}.$$

Figure III.1 represents the general idea of the proposed "parametric" neural network approach.

$$\underset{\text{input}}{\boldsymbol{x}_i} \longrightarrow \text{FCNN} \longrightarrow \underset{\text{output}}{\begin{pmatrix} y_{i0} & y_{i1} & \cdots & y_{iJ} \\ y_{i0} & y_{i1} & \cdots & y_{iJ} \end{pmatrix}} \xrightarrow{\text{activation}} \underset{\text{response}}{\begin{pmatrix} k_{i0} & k_{i1} & \cdots & k_{iJ} \\ \lambda_{i0} & \lambda_{i1} & \cdots & \lambda_{iJ} \end{pmatrix}}$$

FIGURE III.1. *Illustration of the proposed neural network for the "parametric" approach. The input $\boldsymbol{x}_i$ is passed through a fully connected neural network (FCNN) and outputs the subject-specific shape, k_{ij}, and scale, λ_{ij} parameters of the Weibull distributions for the censoring, event of interest and competing event, respectively. The output is the neural network's output before any activation function is applied, and the response is the output after an activation function is applied. The activation function is usually a non-linear function of the inputs, and it can be different for the various k and λ.*

III.2.2. *"Non-parametric" neural network*

The second approach, the "non-parametric" neural network, parametrizes the hazard functions based on a discrete time scale. This model is termed "non-parametric" not due to the absence of parameters, but in contrast to "parametric" neural networks, which generate parameters for Weibull distributions.

Given times $\mathcal{T} = \{T_0 = 0, T_1, T_2, ..., T_M, T_{M+1} = \infty\}$, $T_i < T_{i+1}$, the "non-parametric" neural network approach models the piecewise constant cause- and censoring-specific hazard functions as

$$\begin{pmatrix} h_{i00} & h_{i01} & \cdots & h_{i0J} \\ h_{i10} & h_{i11} & \cdots & h_{i1J} \\ \vdots & \vdots & \ddots & \vdots \\ h_{iM0} & h_{iM1} & \cdots & h_{iMJ} \end{pmatrix} = \eta(\boldsymbol{x}_i; \boldsymbol{\theta}), \tag{III.5}$$

where η represents the neural network functional, $\boldsymbol{\theta}$ is the set of all parameters/weights of the neural network model, and h_{imj} hazards for the i-th observation, the j-th cause, $j = 0, \ldots, J$ at time point T_m, $m = 1, \ldots, M$. Thus, the cause- and censoring-specific hazards are a function of time t, covariates $\boldsymbol{x}_i$, and parameters $\boldsymbol{\theta}$ parametrized as

$$h_j(t \mid \boldsymbol{x}_i) = \eta(\boldsymbol{x}_i; \boldsymbol{\theta})_{mj}, \text{ for } m \text{ s.t. } T_m \leq t < T_{m+1},$$

such that the corresponding cumulative hazard functions have the form

$$H_j(t \mid \boldsymbol{x}_i) = \sum_{m:\, T_{m+1} \leq t} (T_{m+1} - T_m)\, h_{imj} + (t - T_n)\, h_{inj},$$

where h_{imj} is the piecewise constant hazard between T_m and T_{m+1}, and n is the index for which $T_n \leq t < T_{n+1}$. The neural network output in Equation III.5 is achieved by using a 1D convolutional neural network with a kernel of size one. Figure III.2 represents the general idea of the proposed "non-parametric" neural network approach.

$$\underset{\text{input}}{\boldsymbol{x}_i} \longrightarrow \text{CNN} \longrightarrow \underset{\text{output}}{\begin{pmatrix} y_{i00} & y_{i01} & \cdots & y_{i0J} \\ y_{i10} & y_{i11} & \cdots & y_{i1J} \\ \vdots & \vdots & \ddots & \vdots \\ y_{iM0} & y_{iM1} & \cdots & y_{iMJ} \end{pmatrix}} \xrightarrow{\text{activation}} \underset{\text{response}}{\begin{pmatrix} h_{i00} & h_{i01} & \cdots & h_{i0J} \\ h_{i10} & h_{i11} & \cdots & h_{i1J} \\ \vdots & \vdots & \ddots & \vdots \\ h_{iM0} & h_{iM1} & \cdots & h_{iMJ} \end{pmatrix}}$$

FIGURE III.2. *Illustration of the proposed neural network for the "non-parametric" approach. The input $\boldsymbol{x}_i$ is passed through a 1D convolutional neural network (CNN) with a kernel size of one, and outputs the subject-specific piecewise constant cause- and censoring-specific hazards at pre-defined times $t \in \mathcal{T}$. The output is the neural network's output before any activation function is applied, and the response is the output after an activation function is applied.*

III.3. ILLUSTRATION OF THE METHOD

We use the data studied by Thomsen et al. (2016) to demonstrate our method. The study was aimed to investigate the prognostic value of Prostate-Specific Antigen (PSA) kinetics in patients with localized prostate cancer, who were managed by active surveillance. The primary objective was to determine how PSA kinetics, specifically PSA velocity, were associated with prostate-cancer-related mortality in these patients. The study demonstrated that the prognostic value of PSA kinetics in predicting prostate cancer mortality varies depending on the initial PSA levels. PSA kinetics were more predictive for patients with intermediate PSA levels, but not for those with low or high PSA levels. The PSA kinetics are summaries of the underlying data. Here we use the raw PSA measurements as time series input to the neural network and in this way demonstrate the use of our method on functional data. The time series of the PSA measurements consists of 24 PSA measurements carried out over a period of 2 years, corresponding to approximately one measurement per month. For the sole purpose of illustration, we have replaced missing PSA measurements by linear interpolation.

Our architecture utilized a convolutional NN (CNN) to process the PSA measurements. The output from the CNN module was subsequently concatenated with the other covariates to serve as input for a standard fully connected feed-forward NN (FNN, LeCun et al., 2015). To optimize the hyperparameters and evaluate the performance of the NNs, we employed nested cross-validation (nCV, Bates et al., 2024). This process involved 10 folds in the outer loop and 10 folds in the inner loop. The inner loop was dedicated to hyperparameter tuning, while the outer loop was used for model evaluation.

The hyperparameters subject to tuning included the number of layers (1, 2, 3), the number of filters (1, 2, 3) in each layer, and the kernel size (3, 5, 7) of the CNN. Additionally, for the FNN, we tuned the number of layers (1, 2, 3), the number of nodes (1, 2, 3) in each layer. We also tuned the learning rate (0.1, 0.01, 0.001). The values in parentheses represent the options considered for each hyperparameter. Given that each inner loop iteration could yield different optimal parameters, the final model may differ across folds. Consequently, the reported performance reflects the average performance of the prediction modeling algorithm rather than a single model's performance.

To evaluate the performance of our NN method, we used the outer loop to calculate the time-point specific AUC, as illustrated in Figure III.3, and the time-point specific Brier score, as illustrated in Figure III.4 which shows the time-point specific index of prediction accuracy (IPA, Kattan & Gerds, 2018). Note that IPA $= 1 - \frac{BS_{\text{model}}}{BS_{\text{null model}}}$. We compared the results to two cause-specific Cox models (Ozenne et al., 2017) with (1) other covariates but no PSA measurements, and (2) other covariates and PSA velocity. For the cause-specific Cox models, we employed standard 10-fold cross-validation, utilizing the same folds as those in the outer loop for the NNs.

Both NN methods perform well in terms of AUC, Figure III.3. However, compared to the null model (the combination of the cause-specific Cox models without covariates), all models perform worse for prediction horizons until nine years. All models were performing better than the null model for long-term predictions (after 9 years) in terms of the IPA, Figure III.4. In terms of the AUC, the "non-parametric" NN shows superior performance, while the "parametric" NN performs only slightly better than the cause-specific Cox models with PSA measurements. However, in terms of the Brier score, or specifically the IPA,

for prediction horizons up to 9 years our models and the cause-specific Cox models with PSA velocity perform worse than the cause-specific Cox models without PSA measurements. In the later years of follow-up, both NN models outperform the cause-specific Cox models.

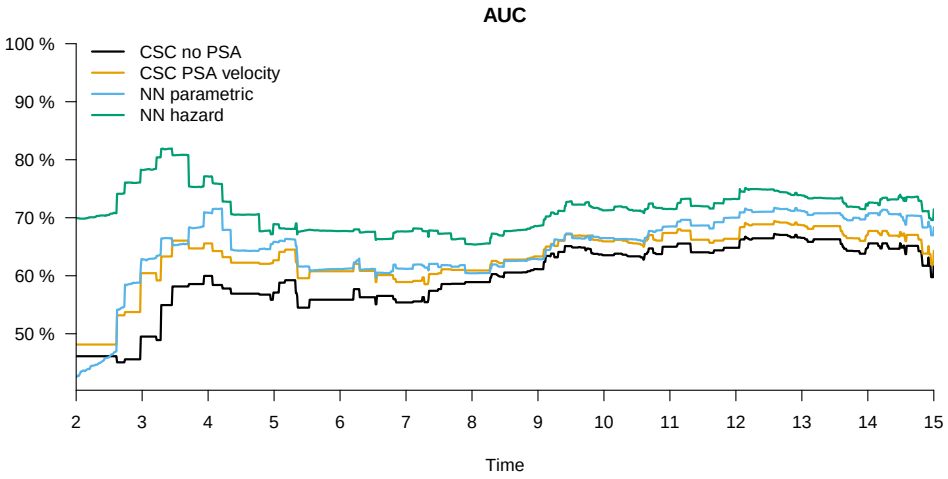


FIGURE III.3. *Comparison of the AUC between the "parametric" NN (NN parametric), "non-parametric" NN (NN hazard), a combination of cause-specific Cox models with only tabular covariates (CSC no PSA), and a combination of cause-specific Cox models with tabular covariates and PSAvel. The reported AUC was obtained using 10-fold nCV for the NN models, and 10-fold CV for the cause-specific Cox models, with the same data split in the folds.*

III.4. DISCUSSION

The results of our study demonstrate the potential of neural networks in predicting time-to-event outcomes in continuous time with competing risks. Our proposed models offer advantages over traditional methods such as the Cox proportional hazards model. These advantages include the ability to handle high-dimensional data, incorporate various types of input data, and estimate complex functional relationships without the need for pre-specified interactions or non-linear terms. Additionally, our method allows for joint estimation of the (competing) events as well as the censoring.

Our neural network models were evaluated using data from a study on Prostate-Specific Antigen (PSA) kinetics in patients with localized prostate cancer. The results indicate that our method performs well in terms of the AUC and Brier score, particularly after the initial years of follow-up.

When compared to cause-specific Cox models, our neural network models showed improved performance in the later years of follow-up. This improvement could be attributed to the neural network's ability to model complex interactions and non-linear relationships in the data. However, it is important to note that in the early years of follow-up, the neural network models performed worse than the null model (the combination of the cause-specific Cox models without covariates). This discrepancy highlights the need for further

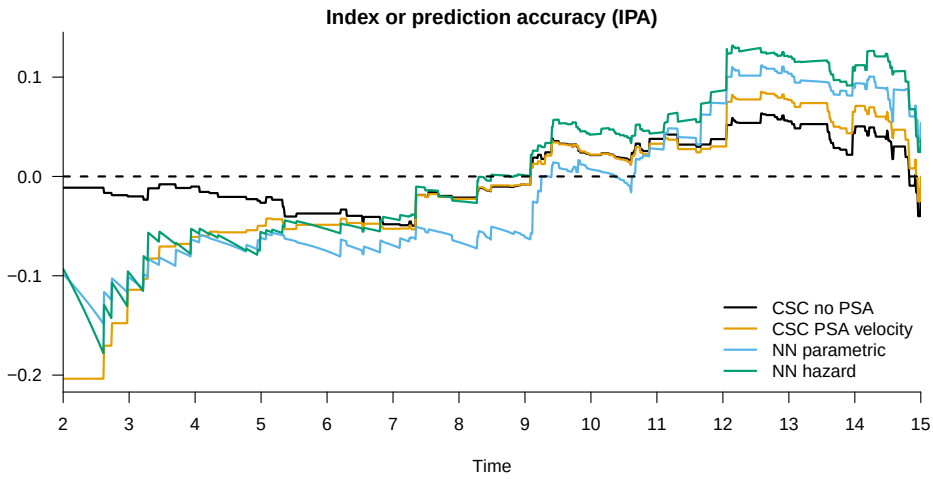


FIGURE III.4. *Comparison of the IPA between the "parametric" NN (NN parametric), "non-parametric" NN (NN hazard), a combination of cause-specific Cox models without PSA (CSC no PSA), and a combination of cause-specific Cox models with covariates and PSA velocity. The reported IPA was obtained using 10-fold nCV for the NN models, and 10-fold CV for the cause-specific Cox models, with the same data split in the folds.*

refinement of the models to improve their performance across all time periods.

Neural networks come with further challenges such as overfitting and computational expense. Overfitting is particularly problematic with small sample sizes, especially when high-dimensional covariates are involved. We employed nested cross-validation to optimize hyperparameters and evaluate model performance, ensuring that the models were not overfitted to the training data.

This study has also several limitations that should be acknowledged. Firstly, we did not conduct simulation studies to demonstrate the behavior and robustness of our models under various conditions. Such simulations could provide further insights into the model's performance in different scenarios. Secondly, we did not compare the performance of our models to existing neural network models for time-to-event outcomes. Benchmarking against established models is crucial for validating the competitiveness of our approach. Additionally, our study is limited by the scope of the data used, which does not fully represent high-dimensional data. Future research should address these limitations by incorporating simulation studies, performing comprehensive comparisons with existing models, and utilizing more diverse datasets to improve the generalizability of the findings.

The findings of this study have several implications for future research. First, the ability of neural networks to handle high-dimensional and complex data makes them suitable for a wide range of applications in survival analysis. Future studies could explore the use of neural networks in other medical and engineering domains, where time-to-event data is prevalent. Second, the integration of different types of data, such as functional and image data, could further enhance the predictive power of neural network models.

In conclusion, our study demonstrates that NNs are a promising tool for predicting time-to-event outcomes in continuous time with competing risks.

BIBLIOGRAPHY

ABU-MOSTAFA, Y., MAGDON-ISMAIL, M., & LIN, H.-T. (2012, March). E-chapter 7: Neural networks. In *Learning from data*. https://amlbook.com/eChapters.html

ANDERSEN, P. K., ABILDSTROM, S. Z., & ROSTHØJ, S. (2002). Competing risks as a multi-state model. *Statistical Methods in Medical Research*, *11*(2), 203–215. https://doi.org/10.1191/0962280202sm281ra

ANDERSEN, P. K., & KEIDING, N. (2002). Multi-state models for event history analysis. *Statistical Methods in Medical Research*, *11*(2), 91–115. https://doi.org/10.1191/0962280202SM276ra

BATES, S., HASTIE, T., & TIBSHIRANI, R. (2024). Cross-validation: What does it estimate and how well does it do it? *Journal of the American Statistical Association*, *119*(546), 1434–1445. https://doi.org/10.1080/01621459.2023.2197686

BENNIS, A., MOUYSSET, S., & SERRURIER, M. (2020). Estimation of conditional mixture weibull distribution with right censored data using neural network for time-to-event analysis. In H. W. LAUW, R. C.-W. WONG, A. NTOULAS, E.-P. LIM, S.-K. NG, & S. J. PAN (Eds.), *Advances in knowledge discovery and data mining* (pp. 687–698, Vol. 12084). Springer International Publishing. https://doi.org/10.1007/978-3-030-47426-3_53

BENNIS, A., MOUYSSET, S., & SERRURIER, M. (2021). DPWTE: A deep learning approach to survival analysis using a parsimonious mixture of weibull distributions. In I. FARKAŠ, P. MASULLI, S. OTTE, & S. WERMTER (Eds.), *Artificial neural networks and machine learning – ICANN 2021* (pp. 185–196, Vol. 12892). Springer International Publishing. https://doi.org/10.1007/978-3-030-86340-1_15

BIGANZOLI, E., BORACCHI, P., MARIANI, L., & MARUBINI, E. (1998). Feed forward neural networks for the analysis of censored survival data: A partial logistic regression approach. *Statistics in Medicine*, *17*(10), 1169–1186. https://doi.org/10.1002/(SICI)1097-0258(19980530)17:10⟨1169::AID-SIM796⟩3.0.CO;2-D

BIGANZOLI, E. M., BORACCHI, P., AMBROGI, F., & MARUBINI, E. (2006). Artificial neural network for the joint modelling of discrete cause-specific hazards. *Artificial Intelligence in Medicine*, *37*(2), 119–130. https://doi.org/10.1016/j.artmed.2006.01.004

CHING, T., ZHU, X., & GARMIRE, L. X. (2018). Cox-nnet: An artificial neural network method for prognosis prediction of high-throughput omics data (F. MARKOWETZ, Ed.). *PLOS Computational Biology*, *14*(4), 1–18. https://doi.org/10.1371/journal.pcbi.1006076

HASTIE, T., TIBSHIRANI, R., & FRIEDMAN, J. (2009). *The elements of statistical learning: Data mining, inference, and prediction*. Springer Science & Business Media.

KATTAN, M. W., & GERDS, T. A. (2018). The index of prediction accuracy: An intuitive measure useful for evaluating risk prediction models. *Diagnostic and Prognostic Research*, *2*(1), 7. https://doi.org/10.1186/s41512-018-0029-2

LECUN, Y., BENGIO, Y., & HINTON, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. https://doi.org/10.1038/nature14539

NAGPAL, C., LI, X., & DUBRAWSKI, A. (2021). *Deep Survival Machines* : Fully parametric survival regression and representation learning for censored data with competing risks. *IEEE Journal of Biomedical and Health Informatics*, *25*(8), 3163–3175. https://doi.org/10.1109/JBHI.2021.3052441

OZENNE, B., SØRENSEN, A. L., SCHEIKE, T., TORP-PEDERSEN, C., & GERDS, T. A. (2017). Riskregression: Predicting the risk of an event using cox regression models. *The R Journal*, *9*(2), 440–460.

PAOLUCCI, I., LIN, Y.-M., ALBUQUERQUE MARQUES SILVA, J., BROCK, K. K., & ODISIO, B. C. (2023). Bayesian parametric models for survival prediction in medical applications. *BMC Medical Research Methodology*, *23*(1), 250. https://doi.org/10.1186/s12874-023-02059-4

THOMSEN, F., BRASSO, K., BERG, K., GERDS, T., JOHANSSON, J.-E., ANGELSEN, A., TAMMELA, T., & IVERSEN, P. (2016). Association between PSA kinetics and cancer-specific mortality in patients with localised prostate cancer: Analysis of the placebo arm of the SPCG-6 study. *Annals of Oncology*, *27*(3), 460–466. https://doi.org/10.1093/annonc/mdv607

WEIBULL, W. (1951). A statistical distribution function of wide applicability. *Journal of applied mechanics*.

WIEGREBE, S., KOPPER, P., SONABEND, R., BISCHL, B., & BENDER, A. (2023). Deep learning for survival analysis: A review. https://doi.org/https://doi.org/10.48550/arXiv.2305.14961

WRIGHT, M. N., KUSUMASTUTI, S., MORTENSEN, L. H., WESTENDORP, R. G. J., & GERDS, T. A. (2021). Personalised need of care in an ageing society: The making of a prediction tool based on register data. *Journal of the Royal Statistical Society: Series A (Statistics in Society)*, *184*(4), 1199–1219. https://doi.org/10.1111/rssa.12644