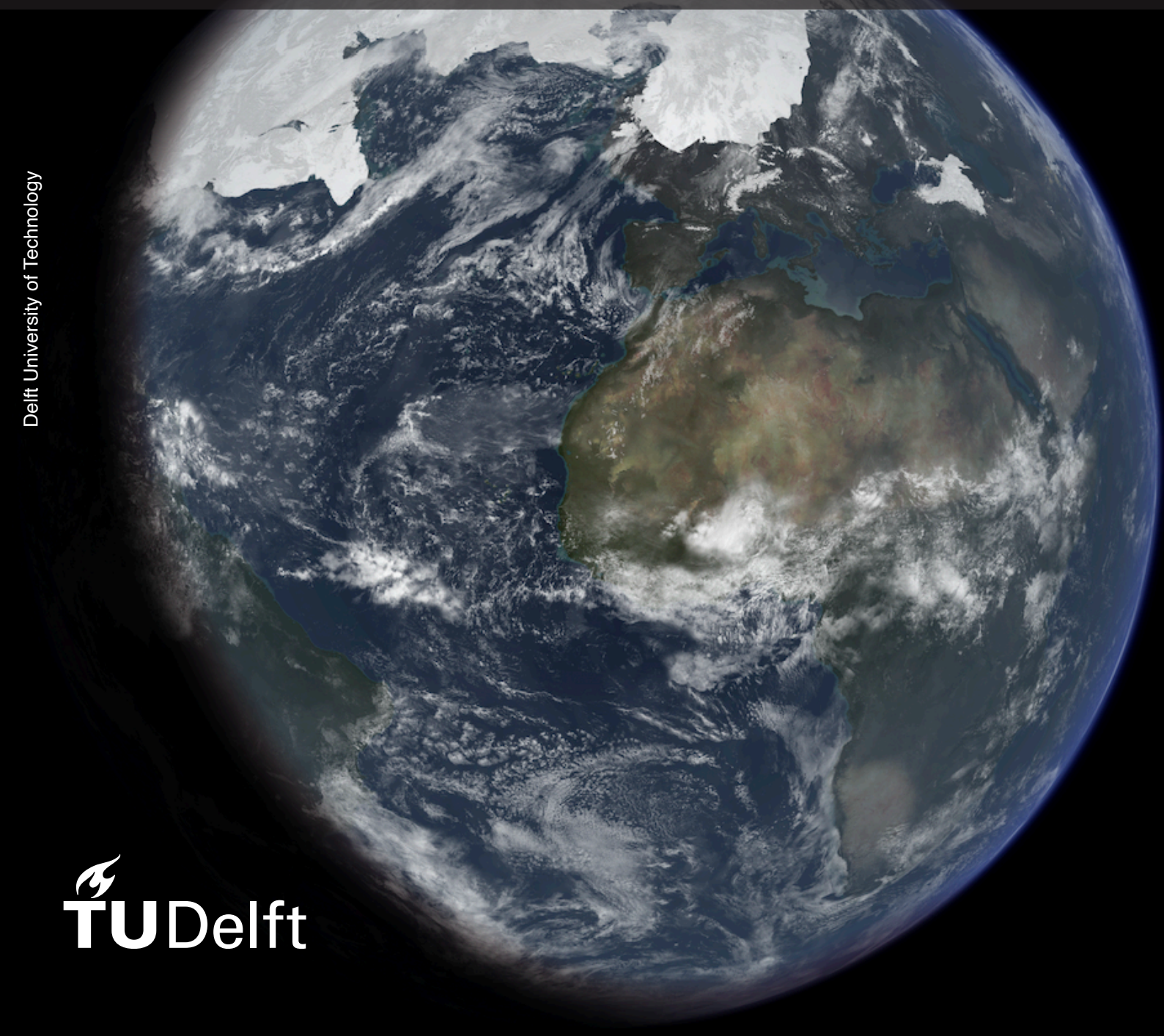
Glacial Isostatic Adjustment Forward Model with a Convolutional Neural Network

Master Thesis

Quinten Bouma



Delft University of Technology

TU Delft

Glacial Isostatic Adjustment Forward Model with a Convolutional Neural Network

Master Thesis

by

Quinten Bouma

in partial fullfilment of the requirements for the degree of

Master of Science
in Aerospace Engineering

at Delft University of Technology

Student Number:	4386582
Master Track:	Space Flight
Master Profile:	Space Exploration
Supervisor:	Dr.ir. W. van der Wal
Place:	Faculty of Aerospace Engineering, Delft
Project Duration:	October, 2022 - June, 2023

Cover Image: Artist impression of the Earth at the last glacial maximum Crowley, 1995

TU Delft

Preface

Completing this thesis signifies the culmination of an exhilarating period in my academic journey, marking the end of my Master of Science degree in Space Exploration. The adventure of navigating through the complex terrains of Glacial Isostatic Adjustment and Machine Learning has been as rewarding as it has been challenging. Though I am eager to present the results of my efforts, encompassed in thousands lines of code and self-explaining schematics, a tinge of regret accompanies the realization that this chapter is drawing to a close.

In the exploration of the multifaceted capabilities of machine learning in the context of GIA, I have gained confidence that these fields will continue to intersect and yield exciting insights. The anticipation of future research in this sphere fills me with excitement. While the intricate interplay between these areas might seem challenging to those outside these disciplines, the broader implications are universally significant and understandable. However, the fusion of these fields requires a foundational understanding of both, as this thesis does not dive into every fundamental aspect.

The journey wasn't without challenges, but they were mitigated by the strong support system that stood with me throughout my thesis. Foremost, I extend my profound gratitude to my supervisors, Wouter van der Wal from TU Delft and Marc Rovira Navarro from University of Arizona, who has proved to be an invaluable mentor from multiple corners across the globe during my thesis period. Their deep knowledge in Earth Sciences, including GIA, complemented by their, surprising, solid understanding of machine learning, was instrumental in bridging these two domains. I have always felt more at ease after an explanation from either of them.

Additionally, I owe a debt of gratitude to Yucheng Lin from Durham University, whose concurrent PhD work on a closely related problem provided a wealth of shared knowledge and ideas.

Last but not least, I am deeply thankful to my family and friends for their unwavering support, motivation, and much-needed distractions. Even when the complex issues I grappled with were beyond their comprehension, their willingness to lend an ear and provide comfort was immensely encouraging.

Quinten Bouma
Amsterdam, June 2023

Abstract

Glacial Isostatic Adjustment (GIA) is the Earth's viscoelastic response to changes in surface ice mass distribution, such as those seen in the last glacial cycle. Melting ice redistributes mass on Earth's surface, causing crustal deformation and altering the geoid, thus affecting sea levels. Addressing GIA modeling and understanding the changes in the Earth's systems involves two significant challenges. The first is the uncertainty in GIA model outputs due to the poorly-constrained ice load history, which refers to the spatio-temporal variations of ice mass during the last glacial cycle. Historic ice loading, reconstructed from geological and geophysical sources, feeds into GIA models but is often incomplete or uncertain. The Sea-Level Equation (SLE) has traditionally been used to infer Relative Sea-Level (RSL) changes from ice load history and Earth rheology. This process is also known as the forward model. However, the SLE is computationally expensive, complicating efficient exploration of diverse ice load histories and Earth parameters. The second challenge lies in tackling inverse problems. This involves inferring ice load history and Earth's interior properties from observations of crustal deformation and sea-level changes, essentially working backwards from the effects to uncover the causes. This problem is intrinsically underdetermined, leading to a multitude of possible solutions that fit the observed data. Therefore, a surrogate model or emulator capable of rapidly estimating RSL history offers a promising solution to both challenges. Such a tool could accelerate evaluation of numerous ice load scenarios, facilitating comprehensive assessment of GIA modeling uncertainties. This, in turn, could improve predictions of sea level changes, informing coastal development and climate change mitigation strategies, while enhancing our understanding of Earth's past climatic events and interior properties. Machine Learning (ML) has proven a valuable tool in solving complex problems and making predictions in various fields, including geophysics and climate science. Its ability to handle complex non-linear relationships, automate feature extraction, and manage large datasets makes ML particularly suited for GIA modeling. By incorporating ML techniques, emulators can achieve more realistic accuracies in GIA modeling outputs. The use of a high resolution forward emulator offers a more efficient and cost-effective research approach, making large ensemble studies accessible even to those with limited computational resources. By enhancing the realistic accuracy of model outputs and facilitating a thorough exploration of possible scenarios, ML can effectively address both challenges, providing a powerful tool for advancing our understanding of GIA and its impacts. Current efforts are underway to integrate ML techniques into the field of GIA, as demonstrated by the ongoing PhD research on a deep learning based forward model GIA emulator by Lin et al., 2023. This research aims to design an algorithm capable of serving as a surrogate for the SLE, leveraging Convolutional Neural Networks (CNNs) based on their known efficiency with image-to-image regression tasks. Given the inherent three-dimensionality of geophysical signals from the GIA model, the data is mapped onto a spherical representation to counter limitations of planar projections. Spherical representations better capture the nature of the GIA model, aiding in improved learning and information retention. The network architectures were adapted from existing works of Zhao et al., 2019, Defferrard et al., 2020 and tested empirically for efficiency. The network was designed using a U-Net architecture, introduced by Ronneberger et al., 2015, to incorporate strong hierarchical feature extraction necessary for an adequate detection of the complex patterns of the GIA model. In the thesis two experiments are conducted to evaluate the performance of the newly designed GIA emulators, specifically comparing their efficiency, accuracy, and versatility against existing GIA modeling methods. The first experiment assesses one emulator's ability to convert ice load history into RSL, given a constant Earth model. The second experiment evaluates the capability of an adapted variant of the initial emulator to generate a different output: GIA uplift rates derived from ice load history, thereby simplifying the multi-channel output to a single channel. The results of these experiments provide insights into the capabilities and effectiveness of the new GIA emulators in replicating the physical model. The emulator from the first experiment achieves a Mean Absolute Error (MAE) 1.13 m between RSL prediction and target output, with satisfactory consistency in terms of spatial prediction performance and trend capture across timesteps. Despite high precision, the predictions showed lower accuracy relative to the RSL data, due to factors such as resolution differences, simplifications in the Sea-Level Equation (SLE) mathematical method, and potential

inaccuracies in the Earth model. The prediction of GIA uplift rates from ice loading performed in the second experiment achieved a MAE of 0.082 mm/yr. In some regions, especially those with limited observational data, the emulator's prediction error remained significantly lower than the typical model-data discrepancies observed in GIA modeling. This average accuracy measure makes the emulator performance acceptable as a surrogate for the physical model. In addition, the emulators were able to make predictions far more quickly than traditional models, with computation time averaging between 0.5 - 1.5% of the time needed for the pseudo-spectral TUDSLE. This increase in efficiency opens the door for more extensive exploration of various scenarios and parameters, characteristic of the inverse problem and GIA sensitivity analyses. While the research successfully developed a GIA emulator, its broader contribution is subject to several limiting factors. The emulator's accuracy has been reasonably achieved, but there is room for further development in the global grid resolution of its predictions necessary for any real practical application in GIA research. Currently, the emulator operates at a grid resolution of 3.75 x 3.75 degrees, while a more detailed 1 x 1 degree grid is commonly used in the field. Enhancing the resolution would render the emulator's predictions more precise and nuanced, thereby increasing its utility and relevance. Another considerable limitation arises from the fact that both emulators are built using a single Earth model. This approach heavily depends on the assumptions inherent to that model, which does not hold universally or under varying conditions. To increase their applicability across various scenarios, future research should focus on incorporating a diverse range of Earth models to account for varying conditions and assumptions.

Contents

Nomenclature

Abbreviations

AE	Autoencoder
CMB	Cosmic Microwave Background radiation
CNN	Convolutional Neural Network
CPU	Central Processing Unit
DiNe	Direct Neighbor
EVD	Eigenvalue Decomposition
GCNN	Graph Convolutional Neural Network
GIA	Glacial Isostatic Adjustment
GPS	Global Positioning System
GPU	Graphics Processing Unit
GLAC	Greenland Ice Model
GRACE	Gravity Recovery and Climate Experiment
GSP	Graph Signal Processing
LGM	Last Glacial Maximum
LSTM	Long Short-Term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MSE	Mean Squared Error
NMM	Normal Mode Method
PCA	Principle Component Analysis
PGR	Post-Glacial Rebound
PSNR	Peak Signal-to-Noise Ratio
R2	R-Squared Coefficient
ReLU	Rectified Linear Unit
RNN	Recurrent Neural Network
RSL	Relative Sea Level
SHT	Spherical Harmonic Transform
SLE	Sea Level Equation
SSIM	Structural Similarity Index Measure
SVD	Singular Value Decomposition
UQ	Uncertainty Quantification
VLM	Vertical Land Motion
WLSQ	Weighted Least Squares Optimization
WPCA	Weighted Principle Component Analysis

List of Symbols

Greek Symbols

θ	Latitude	[rad]
λ	Longitude	[rad]
ω	Weight	[-]
$\mathcal{N}(\sigma,\ \mu)$	Normal distribution	[-]
σ	Standard deviation	[-]
μ	Mean	[-]
γ	Reference surface gravity	[m/s^2]
ρ_i	Density of ice	[kg/m^3]
ρ_o	Density of ocean	[kg/m^3]
ρ	Density	[kg/m^3]
$\mathcal{L}$	Surface load variation	[-]
$\mathcal{O}$	Ocean function	[-]
$\mathcal{G}$	Geoid variation	[m]

Latin Symbols

$\mathbf{I}$	Ice load history	[m]
$\mathbf{I_s}$	Shifted ice load history	[m]
t	Time	[ky]
$\mathbf{R}(t)$	Synthetic generated ice load history	[m]
$\mathbf{X}$	Data matrix	[m]
$\mathbf{X}_c$	Centered data matrix	[m]
$\mathbf{X}_z$	Standardized data matrix	[-]
$\hat{\mathbf{X}}_z$	Reconstructed data matrix	[-]
$\mathbf{C}$	Weighted covariance matrix	[-]
$\mathbf{W}$	Weight matrix	[rad]
$\mathbf{Q}_k$	Principle coefficients	[-]
$\mathbf{V}_k$	Principle eigenvectors	[-]
$\mathbf{Z}_k$	Transform	[-]
k	Number of principle components	[-]
S	Scaling factors	[-]
$\mathbf{S}_{region}$	Regional spatial PCA map	[-]
$\mathbf{T}_{region}$	Regional temporal PCA map	[-]
$\mathbf{M}(t)$	Mean ice load history	[-]
$S(\omega, t)$	Sea level change	[m]
$U(\omega, t)$	Vertical displacement of the sea floor	[m]
$C_{SL}(t)$	Spatially uniform constant	[-]
G_ϕ	Green's function for total variation of gravity potential	[-]
G_u	Green's function for vertical displacement	[-]
G_S	Sea level Green's function	[-]
L	Laplacian matrix	[-]
D	Degree matrix	[-]
A	Adjacency matrix	[-]
K	Chebyshev polynomial order	[-]

List of Figures

List of Tables

Chapter 1

Introduction

This chapter serves as an introduction to the research topic of Glacial Isostatic Adjustment (GIA). First, the general problem and its relevance are explored in Section 1.1. The section starts with a detailed explanation of the concept of GIA and its modeling approaches. Then the complexity is discussed along with the associated uncertainties and limitations. Addressing the need for an efficient solution to these challenges, the potential applicability of machine learning in GIA modeling is proposed and elaborated upon in Section 1.2. The overarching aim of this study is to develop a deep learning model capable of emulating the complex and computationally intensive GIA forward model. To achieve this goal, a main research question and subquestions have been formulated to guide the research, presented in Section 1.3. The chapter is concluded with an outline of the report.

1.1 Problem statement and relevance

Glacial isostatic adjustment (GIA) refers to the secular coupled response of the (viscoelastic) solid Earth to changes in ice mass distribution on the surface, such as those that occurred during the last glacial cycle (Whitehouse, 2018), shown in Figure 1.1.

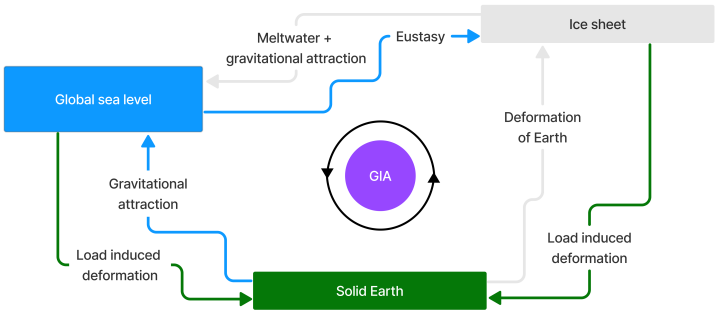
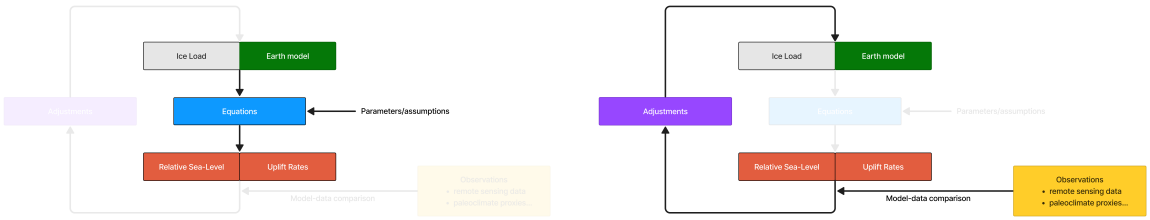


Figure 1.1: Conceptual representation of the GIA problem showing the dynamic interactions between sea-level, ice sheets and the solid Earth. Eustasy refers to the global rise or fall in sea level caused by changes in the volume of water in the world's oceans

The melting of ice sheets and glaciers leads to a redistribution of mass on the Earth's surface, causing the Earth's crust to rebound and deform, as shown schematically in Figure 1.3. These changes affect sea level, create local variations in crustal deformation, and alter the geoid, a term referring to the shape that the ocean surface would take under the influence of Earth's gravitation and rotation alone. Understanding GIA processes is important for a variety of fields, including but not limited to geodesy, oceanography, climate science and the study of Earth's interior properties. One of the primary challenges in the modeling of GIA is the uncertainty in the ice load history [1]. This term refers to the spatial and temporal variations in the amount of ice mass that was present on the Earth's surface during the last glacial cycle, which is reconstructed from geological and geophysical data. The ice loading history is approximated using a combination of data sources, such as glacial landforms, ice core records, sea-level records and observations, and numerical ice flow modeling. The data sources help in estimating the extent, thickness, and timing of past ice sheets, and when integrated, they enable a comprehensive reconstruction of ice load history that can be used as input for GIA modeling. However, the available data is often incomplete or uncertain, making it challenging to accurately model GIA processes.

GIA processes are best described by dividing into two distinct models: the forward and inverse, (Figure 1.2).



(a) Forward model **(b)** Inverse model

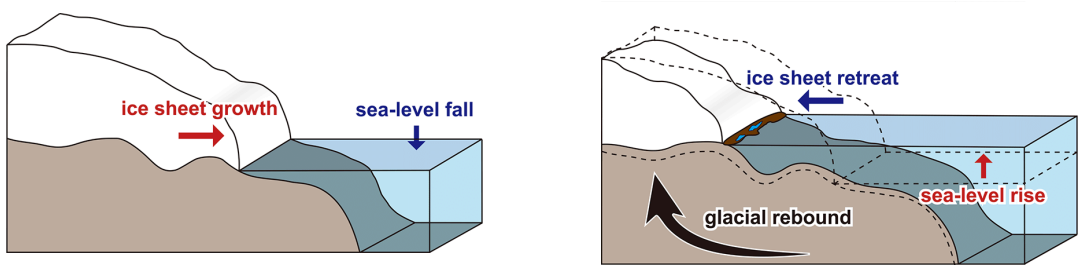
Figure 1.2: Generic depiction of the forward (a) and inverse (b) modeling processes in GIA. An important distinction between the models is that the forward model (a) takes as input the ice load and Earth model, and generates output such as RSL, while the inverse model (b) begins with the output and uses it to infer the input parameters. The diagrams highlight the distinct models and the connections between them

The forward model uses known inputs to predict outputs without including any connections to observations in the computation. It is a simulation describing the physical process, traditionally through a set of equations. In the context of GIA, the inputs comprise the ice loading history and the Earth model (which describes the Earth's structure and rheological properties). The forward model uses these inputs to calculate the expected outputs, such as changes in sea level, land uplift, and changes in the Earth's gravity field. The inverse model works backwards from observed outputs to infer the inputs that would have produced those outputs. It refines the input parameters by calculating the misfit between the model's output and observational data. This observational data includes remote sensing data, such as satellite altimetry and Global Positioning System (GPS) measurements, as well as paleoclimate proxies like ice cores, sediment cores, and sea-level indicators, among others. The process of inferring the Earth properties and ice load history requires many iterations of the forward model. Each iteration involves adjusting the input parameters, running the forward model to generate new outputs, and then comparing these outputs to the observed data. The goal is to find the set of input parameters that minimizes the misfit between the model's output and the observational data. Traditionally, the forward model in GIA studies has been solved using the Sea-Level Equation (SLE), represented by the 'equations' in Figure 1.2a. This mathematical model translates the ice load history and Earth rheology parameters into corresponding changes in Relative Sea-Level (RSL)[1]. Relative Sea Level (RSL) is defined as the difference between the sea surface and the land surface at a particular location, which is influenced by factors such as the geoid variations and land deformations. However, the SLE-based method is computationally intensive and time-consuming. This poses significant challenges to the efficient exploration of a large ensemble of ice load histories and combinations of Earth parameters, which is essential for the application of the inverse model and GIA sensitivity analysis.

GIA modeling is inherently complex and challenging due to a multitude of intertwined factors. In GIA modeling, there are two fundamental unknowns that limit precise and accurate GIA modeling: (1) the Earth's interior parameters, which describe the rheological properties of the lithosphere and mantle and (2) the ice loading history. The first unknown, the solid Earth's response to surface loading, is a nonlinear process that depends on the distribution, timing, and magnitude of the loading. The deformation relies on the density, structure and mechanical properties (e.g. shear modulus, viscosity, Poisson ratio) of the mantle, core and lithosphere. However, our understanding of Earth's interior structure is limited, causing estimates of mantle viscosity and core properties to vary significantly, which in turn leads to considerable uncertainty in GIA predictions. Reconstructions of ice load history are the second fundamental unknown. They are categorized based on the construction approach: the indirect method, also known as the classic approach, and the direct method, also known as the physical approach. The indirect method, which gives rise to classic ice models, reconstructs the history of ice loading using a variety of geological and geophysical data. This data encompasses sea-level indicators, sedimentary records, glacial landforms, gravity measurements, and geodetic observations. In this approach, the forward and inverse models are fused together, incorporating various proxies in an iterative process.

[1] For increased understanding example visualizations of ice load history and RSL are provided in Figure B.1 and Figure 2.1

However, this method is not without its limitations. Uncertainties arise due to factors like the distribution of data, the assumptions made during data interpretation, the techniques used for dating, and the absence of constraints related to ice mechanics or knowledge of past climate conditions. Furthermore, this method often relies on simplified Earth models, which can introduce additional uncertainties into the predictions. The direct method, produces physical ice models using advanced computational techniques to model ice sheet behavior based on physical principles and observed data. This approach simulates ice sheet dynamics, taking into account factors such as climate, topography, and bedrock properties, which can vary both spatially and temporally. However, like the classic ice models, the physical ice model also has inherent limitations and uncertainties concerning input parameters and model complexity. Consequently, alterations in loading history or model parameters can significantly influence GIA predictions.



(a) Glacial cycle

(b) Interglacial

Figure 1.3: Schematic showing the relation between sea-level ice loading and the solid Earth during the (a) glacial cycle and (b) interglacial from Suganuma et al., 2020

The ability to determine GIA, ice load history and Earth interior properties is further constrained by the availability and quality of data, particularly in remote regions such as Antarctica and Greenland. Errors and biases in observational data can propagate into GIA models. The data limitations results in multiple plausible GIA response predictions. This nonuniqueness makes it difficult to discern which solution best represents the actual GIA processes. As a consequence, there is no strong consensus on the ice load history and Earth rheology (Spada, 2017, Whitehouse, 2018). This lack of consensus emphasizes the sensitivity of GIA predictions to the choice of model parameters, resolution, and boundary conditions. Moreover, GIA models generally involve intricate numerical simulations that necessitate substantial computational resources and assumptions. In many cases, it is beneficial to have a more efficient computation that does not need to include complex physics to be comprehensive.

The development of a surrogate model, capable of providing rapid RSL history estimates based on paleo-ice load inputs, offers a promising solution to overcome the computational limitations of the SLE method. Such a surrogate model or emulator would enable researchers to explore new hypotheses and conduct more efficient analyses across a wide range of Earth science disciplines. The emulator's accelerated assessment of various ice load scenarios and Earth parameters broadens the understanding of GIA modeling uncertainties, enhancing knowledge of past and future sea level changes and their societal and economic impacts. For example, these improved predictions can guide coastal planning and climate mitigation efforts. Simultaneously, the inverse model's repeated use of the forward model plays a crucial role in constructing an ice model. The surrogate model's computational efficiency can significantly improve ice model refinement, enhancing our understanding of past climate change and future ice sheet and sea level responses to global warming.

Given these challenges and potential solutions, focusing on the timeframe from the Last Glacial Maximum (LGM), which occurred around 20,000 years ago, to the present provides an opportunity to study a period for which there is relatively more preserved evidence. The erosive power of ice sheets is more prevalent during glaciation compared to deglaciation, as observed in Figure 1.3. As ice sheets expand and advance (Figure 1.3a), they exert significant force on the underlying bedrock and sediment, leading to the alteration or removal of geological features and evidence that would provide information about that movement. In contrast, during deglaciation, the retreat of ice sheets allows for the preservation of various geological features and landforms, which provide evidence for understanding past ice loading.

Moreover, the transition from the LGM to the present is ongoing and affects many current measurements making it a clear choice.

Machine learning can play an important role in addressing the complexities and uncertainties associated with GIA modeling. A surrogate model, built using a comprehensive set of GIA simulations, can provide rapid estimates of GIA outputs based on given inputs, effectively replacing the computationally intensive physics model calculations. In the following section, the potential applications of machine learning to solve the GIA forward model are discussed in more detail.

1.2 Machine Learning

Machine learning (ML) has emerged as a powerful tool for solving complex problems and making predictions in various fields, including geophysics and climate science (Karpatne et al., 2019). It has proven effective in addressing challenges where traditional methods face issues with uncertainties, computational and physical complexity, and time-consuming calculations. ML offers several advantages in GIA modeling, including flexibility in handling complex, non-linear relationships, adaptability to new information, automatic feature extraction, and the ability to work with large datasets. These capabilities make ML a well-suited basis for tackling the complexities of the forward model, particularly when it comes to managing large prediction variations arising from uncertainties in the ice load history and Earth models. ML simplifies the problem by learning the relationship between known parameters and the Earth's computed response, serving as a practical replacement for the SLE, thereby streamlining the computational process. This learning task is generally feasible due to the deterministic nature of the forward model, where a given input leads to a unique output. Moreover, with enough encompassing training data, ML models can learn this mapping with a high level of accuracy, effectively bypassing the need for computationally expensive SLE. However, the application of ML to the GIA inverse problem presents more complex challenges, though it has been tackled for similar inverse problems (Araya-Polo et al., 2018, Jin et al., 2017). Unlike the forward model, the inverse problem seeks to infer unknown input parameters from observed earth responses. This issue is considerably more challenging due to nonuniqueness, stability issues, and data availability and sparsity. The ML model would need to learn not just a single mapping, but a range of possible mappings, increasing its complexity. In addition, data limitations are more prevalent in the inverse model due to their ill-posed nature. Lastly, the sparsity and uneven distribution of the constraining data poses a significant challenge as it can disrupt the focus of the learning process. Despite the challenges, it is important to highlight that implementing ML in the forward model can provide considerable benefits to the inverse model, largely due to the enhanced computational efficiency (time and resources) it brings to the forward calculations.

In this thesis, the focus is on applying ML to the GIA forward model. ML provides certain benefits over traditional GIA methods by reducing computational resources, which makes conducting large ensemble studies more feasible. The development of an efficient emulator is not only beneficial to the reliability of GIA-related research findings but also encourages more cost-effective research. A faster computation from ice load and Earth model to GIA output can potentially enhance inverse modeling techniques, for instance, in applications like ice sheet reconstruction. Sensitivity analysis can benefit from a more efficient forward model; for example, it can provide insights into the impact of different Earth rheologies or ice histories on inferred post-glacial rebound, as derived from geodetic measurements. Despite the more common use of ML in fields such as geophysics, climate research, and glaciology, its adoption in GIA is still in its infancy. To the author's knowledge, the only other research effort in this area is a study currently in progress that is exploring a deep learning-based GIA forward model emulator using spectral graph convolution (Lin et al., 2023).

1.3 Research questions

The purpose of this study is to investigate if a GIA emulator can be created using a learning-based autonomous algorithm. This section details the main research question and sub questions that were formulated for this work. The main research question is:

> **How can a deep learning network be designed to accurately emulate the complex and computationally intensive GIA forward model?**

Here the forward model refers to solving the SLE using varying ice load and constant Earth model. To address the main research question, several sub-questions were devised that will help guide the study. These sub-questions are ordered and will be answered as the research progresses. The answers to these sub-questions will collectively contribute to answering the main research question. There are three subquestions that define the phases of the research.
First, a multitude of datasets is required to train and evaluate the learning-based GIA emulator, which are not available. Therefore they have to be artificially created (input) and computed (output). The input is generated based on established ice models and the output is calculated using a existing mathematical model.

1. What are the optimal methods for generating synthetic datasets to train the GIA emulator effectively, ensuring high learning capabilities?

When the datasets have been created, the design of the emulator starts by selecting an appropriate network architecture. The architecture is a fundamental aspect of the design, as different architectures (e.g., Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), fully-connected networks, etc.) are designed for specific problems and data structures. Therefore, the most suited architecture for the GIA emulator has to be identified, considering the data structure and characteristics of the GIA modeling problem.

2. Which architecture is most suited as a basis for the network?

Finally, the overall capability and usefulness of the created emulator is assessed. The traditional GIA model demonstrates significant versatility. The versatility of the mathematical model or Sea Level Equation (SLE) originates from three distinct categories of abilities: (1) changing model fidelity by modifying assumptions, (2) allowing varying input and (3) producing different output estimates. In the first category, model fidelity can be adjusted by selectively enabling or disabling the inclusion of specific GIA processes, such as changes to the geoid or rotational feedback. The second, varying input, entails the possibility of inserting variable ice loading, different Earth models, or both. Finally, the SLE has the ability to yield different outputs like RSL, radial displacement and uplift rates, among others. These dimensions of versatility are essential characteristics that should be reflected in the evaluation of the proposed emulator. Within the scope of the thesis, this is accomplished by systematically varying the ice loading input and estimating RSL and uplift rates as output, while the Earth model remains constant.

3. How does the GIA emulator's performance compare to traditional GIA models in terms of precision, efficiency, and versatility?

To address this sub-question, two experiments will be conducted, examining the emulator's capability for two different outputs: RSL changes and uplift rates. These investigations aim to better understand the potential benefits of the GIA emulator in facilitating more efficient research and enhancing scientific findings in GIA-related studies.

The remainder of this thesis is structured as follows. Firstly, the synthetic dataset generation required for the training of the emulator will be discussed in Chapter 2. Secondly, the design of the algorithm including the network configuration and training optimization is explained in Chapter 3. Then, the two experiments and their methodology are explained in Chapter 4 followed by the results of the respective experiments in Section 4.3. Afterwards, the dataset generation, algorithm design and experiments are discussed together with associated limitations and statements regarding the works's contribution are detailed in Chapter 5. Finally, an outline of potential future work is provided in Chapter 6.

Chapter 2

Synthetic Dataset Generation

This chapter is aimed to answer the first subquestion. Training the deep neural network requires a large collection of synthetic datasets of ice load input and output estimates. As the deep learning model emulates the GIA forward model, synthetic datasets are created by implementing modifications on existing ice models and then applying traditional numerical methods to compute corresponding RSL and uplift rates. The learning of physical relation is strongly related to the quality and diversification of the training data, achieved by imposing randomization methods that extrapolate on the existing ice models. This chapter first discusses the ice models used to generate the input in Section 2.1. Then the two methods for generating the artificial ice load histories are discussed in Section 2.2 and Section 2.3. Finally, the chapter ends with a section on the numerical physics-based method or SLE used to compute the outputs from the artificially created ice load histories.

2.1 Ice Models

This section briefly introduces the selection of existing ice models that will form the basis for training dataset of the neural network. Ice models are ice load histories generated using different methods. They are categorized as: classic ice models, physical ice models and hybrid ice models. Classic ice models have an ice load history created using empirical data, whereas physical ice models are created based on theory by simulating ice flow dynamics. For the emulator the ICE-5G (W. R. Peltier, 2004), ICE-6G (W. Peltier et al., 2015), ICE-7G (Roy, 2017), GLAC-1D (Tarasov, Hughes, et al., 2014) and Gowan (Gowan, 2019) ice model have been used.

2.1.1 Classic ice models : empirically constraint

Classic ice models, such as ICE-5G, ICE-6G, and ICE-7G, are developed with the inverse model using geophysical constraints and are based on empirical data from various sources, such as geomorphological evidence, RSL observations, and global positioning system (GPS) measurements. The process of constructing these classic ice models involves iteratively adjusting the ice load history until the modeled GIA predictions match the observed geological and geodetic constraints. By comparing the predicted GIA signals with the available data, the ice load history is refined and updated to achieve a better fit. However, classic ice models are limited by their reliance on relatively simple assumptions and their sensitivity to Earth model choice. The Earth model defines the viscoelastic structure of the Earth, commonly divided into three layers: the lithosphere and the upper and lower mantle. Further, the classic models lack ice-mechanical constraints and the availability of quality data.

2.1.2 Physical ice models: ice mechanically constraint

Physical ice models, also known as thermodynamical ice models, simulate the physics of ice flow to predict the evolution of ice sheets over time. These models generate an ice loading history or future projections in the form of spatiotemporal variations in the ice thickness field, ensuring glaciological self-consistency (Tarasov and W. R. Peltier, 2002). These models use advanced numerical techniques to simulate the dynamic processes of ice flow and its interaction with the underlying bedrock and the surrounding climate. They solve a number of sequential partial differential equations governing the processes of mass balance and ice flow dynamics. The Gowan ice model used in the thesis is such a numerical ice flow model, although it does not match proxy-based sea level reconstructions (Gowan, 2019). The primary advantage of physical ice models is their relatively lower dependence on Earth models compared to empirically constrained ice models. While these models do often require some information on the Earth's topography and occasionally the properties of the Earth's mantle and lithosphere, their primary focus is on the physics of ice flow. However, physical ice models are computation-

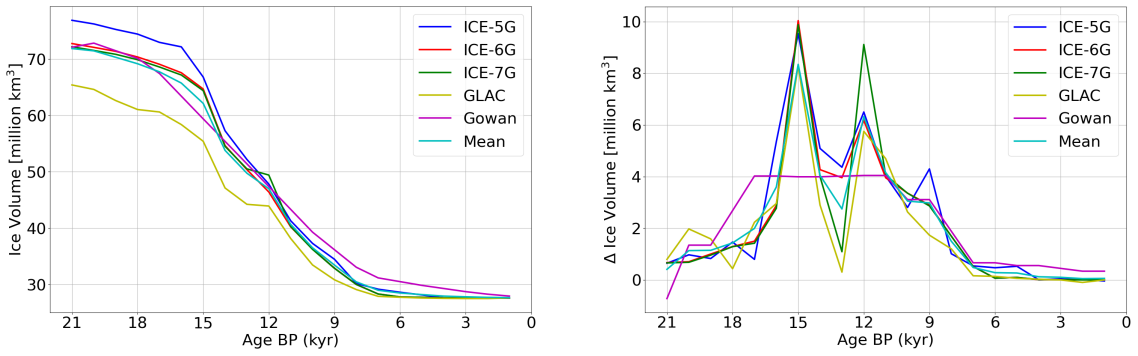
ally demanding due to the complexity of the differential equations involved and require a high level of detail in input data. Furthermore, the ice thickness field is highly sensitive to climatological data, which is often poorly constrained. Despite these challenges, the glaciological self-consistent property makes physical ice models scientifically valuable.

2.1.3 Hybrid ice models

Hybrid ice models, such as GLAC-1D, represent a fusion of classical and physical ice modeling approaches. By incorporating elements from both methodologies, hybrid models aim to balance the simplicity of classical models with the accuracy and detail of physical models. Hybrid models are capable of integrating observational data and constraints from various sources, while simultaneously accounting for the underlying physics of ice sheet dynamics. For the GLAC ice model this is achieved as a result of simulating large ensembles and comparing them with empirical data, while taking into account the physics of ice flow and the interaction between the ice sheets and the underlying bedrock. This combined approach offers improved accuracy and flexibility compared to using either classical or physical ice models independently. It is considered to be the most effective approach for ice sheet modeling, providing a comprehensive understanding of ice dynamics while maintaining a manageable level of complexity.

2.1.4 Ice model selection

The integration of classic ice models (ICE-5G, ICE-6G, and ICE-7G), a physical ice model (Gowan), and a hybrid ice model (GLAC-1D) brings together diverse modeling techniques and assumptions. This diversity helps capture various aspects of ice loading reconstruction and provides a more robust foundation for generating synthetic ice load histories. These discussed differences between the ice models are more apparent when inspecting the cumulative and relative ice volumes over time (Figure 2.1).



(a) Cumulative ice volume change from LGM to present

(b) Relative change in ice volume from LGM to present per timestep

Figure 2.1: Relative change and cumulative ice volume of the selected ice models over time from 21,000 before present (BP) to present. Present ice volume is defined at 27.6 million km^3 from Lemke et al., 2007, where ice sheet volume is comprised of 24.7 million km^3 for Antarctica and 2.9 million km^3 for Greenland

The higher ice volume in the ICE-5G model, shown in Figure 2.1a, is due to simplifying assumptions and data quality, with no access to Gravity Recovery And Climate Experiment (GRACE) measurements during its construction (Schmidt et al., 2014). In contrast, the GLAC-1D model shows the lowest ice volume due to the inclusion of ice flow dynamics, higher spatial resolution, and climate sensitivity. The Gowan ice model, generated using the ICESHEET method, assumes perfectly plastic, equilibrium conditions to produce a plausible ice sheet configuration (Gowan et al., 2016). Gowan's smoother ice volume curve, noticeable in Figure 2.1b, is a result of continuous thermodynamical modeling coupled with a later applied timestep interpolation. The ice models are required to have 1,000 year timesteps necessary for compatibility and to fit the requirement of the mathematical model. However, the Gowan database contains ice loads at epochs of 2,500 years so the data has to be linearly interpolated, which adds to a smoother ice volume curve. The discussed models are widely used with well-documented datasets. However, apart from the GLAC ice model, most lack any associated error bars. This further substantiates the need for a large ensemble method. The combination of the selected ice models ensures reliability in the integration of the existing ice models into the generation process.

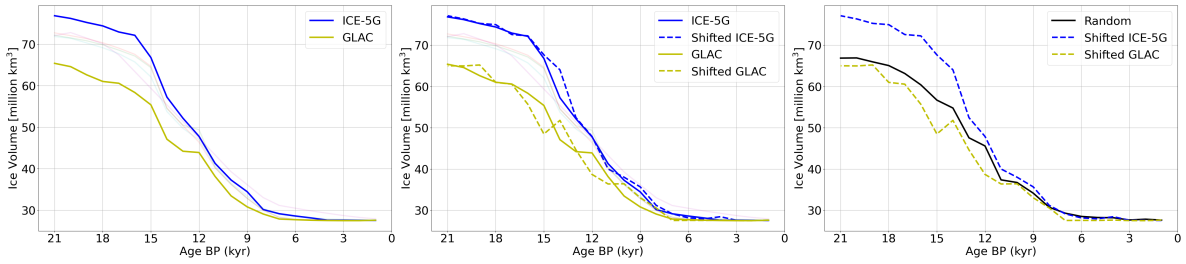
2.1.5 Dataset requirements

Synthetic ice load histories are developed through a mix of randomization and linear combination, which should allow for the creation of up to 10,000 samples within a practical timeframe. Ice model characteristics, such as thickness, extent, and outliers, need to fall within reasonable bounds of the limits set by established models and show roughly equal variations outside these limits in terms of total ice volume per timestep. More specifically, large deviations in total ice volume should not be due to localized extreme values in small areas, ensuring a balanced distribution. This balance in distribution can be reinforced by having a denser representation around the mean values derived from the existing ice models. These synthetic histories retain physical realism to better assist the machine learning algorithm in understanding the physical relationships. Furthermore, ice load histories should be generated using some method that helps the ML algorithm in training, which can be achieved by emphasizing the largest trends in contrast to the smaller variations or 'noise'. Doing so can make the task of feature detection easier for the learning algorithm.

For generating these synthetic datasets, two different methods are used: one simple method that imposes changes on the entire ice model based on a normal distribution and a second, more complex method that examines the direction of the strongest variance in specific regions and uses that as a basis for imposing changes similar to the first method. The upcoming sections will discuss the details of both these generation methods.

2.2 Method One: Select, Shift and Combine

The first method is divided into three steps: selection of ice models, applying a time shift on the selected ice models using interpolation and finally a linear combination of the shifted ice models. Due to simplicity of the method the range of the randomized ice load histories is easy to control.



(a) Step 1: Select **(b)** Step 2: Shift **(c)** Step 3: Combine

Figure 2.2: Example visualization of the three steps that constitute the artificial ice load generation of Method One

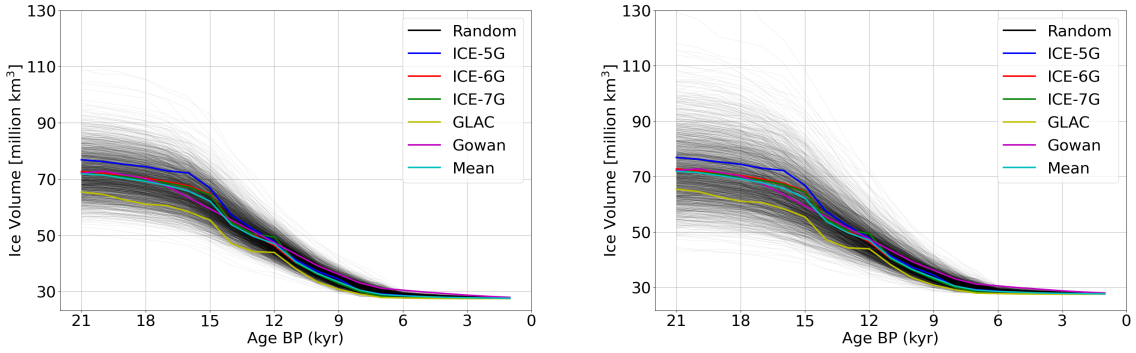
As shown in Figure 2.2a two or three (quantity varies with each iteration) out of five ice models are randomly selected for each run (in the example, ICE-5G and GLAC). This number of ice model selection is performed to avoid potential bias towards individual ice models in the linear combination, which would result in an uneven distribution of ice volumes. Subsequently, the chosen ice models undergo a shift via a one-dimensional interpolation function, applied to the ice loading data matrix $\mathbf{I}$ and its corresponding time series. From normal distribution $\mathcal{N}_{shift}(\mu_s,\ \sigma_s)$ a sample of time shift values Δt is generated to shift the selected ice models, where μ is the mean and σ the standard deviation. Values for μ_s and σ_s are provided in Figure 2.3. The shifts are applied to the selected original ice model loading history $\mathbf{I}$ to create the shifted ice load $\mathbf{I}_s$:

$$\mathbf{I}_s(t) = \mathbf{I}(t + \Delta t) \tag{2.1}$$

where $\mathbf{I}$ and $\mathbf{I}_s$ are relative ice loading (only visualizations are cumulative). Occasionally, in Equation 2.1, the applied Δt reaches beyond the original time series, then the first and last epochs undergo extrapolation instead of interpolation. In Figure 2.2b it is observed that the size of the shifted ice volume varies per epoch instead of being "copied" and moved. This outcome is anticipated since the shift Δt remains constant across all epochs, whereas the increase/decrease in ice load, as observed in Figure 2.1b, does not. Following the shifting of selected ice models, a linear combination is applied:

$$\mathbf{R}(t) = \frac{\omega_1 \, \mathbf{I}_{ice5g}(t) + \omega_2 \, \mathbf{I}_{glac}(t)}{\sum_{n=1}^{n=2} \omega_n} \tag{2.2}$$

Here, **R** represents the randomly generated synthetic ice load history, and ω_n denotes the weights. For every run, two (or three) linear combination weights ω_n are generated using another normal distribution: $\mathcal{N}_\omega(\sigma_\omega,\ \mu_\omega)$. Method One is computed twice with different shift and weight normal distributions $\mathcal{N}_{shift}$ and $\mathcal{N}_\omega$ as shown in Figure 2.3. The resulting ice load history sets are designated as Medium variation and Large variation, based on the extent of randomization.
A slight upward trend from the existing ice models is evident in Figure 2.3b, and the higher average ice volume can be attributed to the second step of the generation method. The $\mathcal{N}_\omega$ normal distribution generates random shift values evenly distributed before and after each epoch. However, ice load changes up to 12,000 years before present are generally increasing (calculation is performed on relative ice load), causing higher change for positive Δt due to the relative difference between later and earlier epochs. Given the high σ_s in Figure 2.3b, this offset is counterbalanced by a lower time shift mean μ_s. The linear combination further contributes to this issue, but the impact's magnitude is reduced and cannot be rectified by altering the random weights.



(a) Medium variation ice load set: 2000 runs with $\mathcal{N}_{shift}(\mu_s = 0,\ \sigma_s = 0.8)$ and $\mathcal{N}_\omega(\mu_\omega = 1,\ \sigma_\omega = 0.2)$

(b) Large variation ice load set: 2000 runs with $\mathcal{N}_{shift}(\mu_s = -0.2,\ \sigma_s = 1.5)$ and $\mathcal{N}_\omega(\mu_\omega = 1,\ \sigma_\omega = 0.3)$

Figure 2.3: Results of the two Method One runs with different normal distributions $\mathcal{N}_{shift}$ and $\mathcal{N}_\omega$. The mean μ_s and standard deviation σ_s connected to the shift represent timesteps Δt of 1,000 years, while the μ_ω and σ_ω of the combination step are dimensionless scalars

The select, shift, combine approach introduces scalar fluctuations throughout the entire global grid, causing ice load variations to be contingent on the differences among the five existing ice models. The resultant variations therefore resemble a blend of the five original models, as observed in the plots of the generated datasets and the original models. When the shift is increased, conformity is mitigated, as evidenced by the contrasting variations between the Medium set (Figure 2.3a) and the Large set (Figure 2.3b). The neural network's potential performance is subject to the balance between diversity in the training data and learning capability; increased diversity, though beneficial for model performance, can complicate the learning process. Therefore, the magnitude of the variation is empirically determined during training optimization, illustrating the need for multiple datasets of different scales. The neural network, an autonomous algorithm, performs by identifying patterns in training data, specifically short-wavelength spatial and long-wavelength temporal components inherent in ice load variations and RSL changes. By using a distinct method to generate synthetic ice load histories emphasizing these features, the network's prediction quality can be improved.

2.3 Method Two: Weighted Principle Component Analysis

This section explains the second dataset generation method using Principal Component Analysis (PCA). Principal component analysis (PCA) is a technique for increasing interpretability, while minimizing information loss (Jolliffe et al., 2016). The idea for using WPCA and the specific use of the weights was suggested by Lin et al., 2023 in a verbal communication. PCA is implemented to interpret the most prominent features in (or across) the ice models. In the context of ice models, it explains the variance of the ice thickness in the spatial and temporal domain. The PCA process can be briefly summarized as

follows: first, the covariance matrix of the data is computed, which measures the relationships between different features in the dataset. Next, the covariance matrix undergoes eigenvalue decomposition, which results in a set of eigenvectors and eigenvalues. Finally, these eigenvectors, known as principal components (PC), are used to project the data onto a lower-dimensional space while preserving the variance.

Method Two is divided into three sections: preprocessing (Section 2.3.1), computing the PCs (Section 2.3.2) and generating randomized ice load histories (Section 2.3.3). The steps described in Section 2.3.2 are calculated using the WPCA package repository from Van der Plas, 2016, which implements an adjusted version of the PCA from Pedregosa et al., 2011.
A principal component (PC) is a vector formed through the linear combination of the original features in a dataset. In the context of the GIA data, the "features" refer to contours on the ice load history and RSL maps, for example, the "jumps" in RSL on the coastlines. The PCs can be thought of as new "axes" in the feature space that are aligned with the directions of greatest variability. PCs are organized in a hierarchical manner, with the first PC capturing the largest variance's direction, followed by the second PC capturing the second largest variance's direction, and so on. Weighted PCA (WPCA) is simply a variant of conventional PCA that incorporates weights in the computation of the PCs. The weights are implemented because of the difference in grid area caused by the distortion of map projections near the polar regions. For example, in Figure 2.6 it is observed that Greenland appears larger than Australia in the equirectangular projection even though the area in km^2 is actually only $\sim$ 30% of Australia.

2.3.1 Preprocessing

Before a WPCA method can be applied, the data, represented as matrix $\mathbf{X}$, must undergo preprocessing via a procedure known as standardization. Standardization is essential in PCA, because it projects the original data onto directions which maximize the variance, making the operation unit and scale sensitive. This process, which entails two distinct steps, centering and scaling, is performed in a specific order (Jolliffe et al., 2016). 'Standardization', in general, encompasses both centering and scaling, and it should be noted that centering is a prerequisite for computing the covariance matrix.

1. **Data Preprocessing**

   - **Centering:** This step subtracts the column (sample) weighted mean, $\boldsymbol{\mu}_{sample} = \frac{\sum w_{i,j}\mathbf{x}_{i,j}}{\sum w_{i,j}}$, from the data, where $w_{i,j}$ are the weights and $\mathbf{x}_{i,j}$ are the data matrix entries, effectively zeroing the weighted mean. The centered data matrix $\mathbf{X}_c$ is represented as $\mathbf{X}_c = \mathbf{X} - \boldsymbol{\mu}_{sample}$.
   - **Scaling:** This step scales the centered data to have unit variance. Here, $\boldsymbol{\sigma}_{sample}$ represents the standard deviation of the data. The standardized data is computed as $\mathbf{X}_z = \mathbf{X}_c / \boldsymbol{\sigma}_{sample}$.

The sequence of centering and scaling during standardization is critical due to the differences in the range of ice load values across various epochs. The mean of these values is non-zero, creating an offset. If the data were to be scaled before centering, i.e., divided by the standard deviation before subtracting the mean, it could lead to undesired results, especially if the mean value of the data is significantly different from zero. Therefore, the mean offset, or bias, is removed first by centering the data.

2.3.2 Computing Principle Components

Principal components (PCs), denoted as $\mathbf{V}_k$, are calculated by computing the covariance matrix $\mathbf{C}$ of the ice model and subsequently performing a direct eigenvalue decomposition (EVD) on $\mathbf{C}$. These PCs are ranked in descending order based on the variance they capture. After the number of PCs, k, is selected, they are projected onto the original data to generate "PC maps", represented as $\hat{\mathbf{X}}$. These PC maps form the foundation for the generation of synthetic ice load histories. The algorithm that further describes the process is presented in Delchambre, 2014.

2. **Covariance Matrix Calculation**

   - Objective: Compute weighted covariance matrix $\mathbf{C}$.
   - Input: Data matrix $\mathbf{X}_z$ and weight matrix $\mathbf{W}(\phi, \lambda) = \cos(\phi_{lat})$.

- Calculate $\mathbf{C} = \frac{1}{\sum w_i}\mathbf{X}_z^\top \mathbf{W}\mathbf{X}_z$.

3. **(Direct) Eigenvalue Decomposition (EVD)**
   - Objective: Decompose covariance matrix to identify main axes of variance.
   - Perform EVD on $\mathbf{C}$ to get eigenvalues $\mathbf{\Lambda}$ and eigenvectors $\mathbf{V}$.
   - EVD: $\mathbf{C} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^T$.
4. **Weighted Least Squares (WLSQ) Optimization**
   - Objective: Iteratively determine set of principal coefficients $\mathbf{Q}$ to minimize weighted residuals for each observation using the weighted PCs.
   - Minimize $\mathbf{W}_i(\mathbf{X}_i - \mathbf{X}_i\mathbf{V}_k\mathbf{V}_k^T)$, subject to $\mathbf{V}_k\mathbf{V}_k^T = \mathbf{I}$ in order to obtain $\mathbf{Q}_k$.
5. **Reconstruction**
   - Objective: Reconstruct data using the principle coefficients and reapply original scale and centering using the mean and standard deviation from the preprocessing step.
   - Create transform $\mathbf{Z}_k$, by projecting $\mathbf{X}_k$ onto the space spanned by the first k PCs using the principle coefficients: $\mathbf{Z}_k = \mathbf{X}_z(\mathbf{Q}_k\mathbf{V}k)$.
   - Reconstruct the data: $\hat{\mathbf{X}} = (\mathbf{Z}_k\mathbf{V}_k^T) \cdot \boldsymbol{\sigma}_{sample} + \boldsymbol{\mu}_{sample}$.

The weight matrix $\mathbf{W}$ is based on the cosine of the latitude on the equirectangular map. High latitudes, near the equator, have low values. Low latitudes, near the Earth's poles, have high values. Notice the weight matrix is used three times: first in preprocessing when centering the data, second in determining the covariance matrix, and later in the iterative WLSQ optimization. The WPCA method does not use the more general singular value decomposition (SVD) directly on $\mathbf{X}$, instead it computes the covariance matrix $\mathbf{C}$ and later applies the EVD. This choice was based on computational expense as explained in Delchambre, 2014. The principle vectors $\mathbf{V}_k$ can be regarded as the directions of maximal variance, while the principle coefficients $\mathbf{Q}_k$ indicate how much to 'move' in each of those directions to approximate each data point in the original dataset. The size of $\mathbf{Q}_k$ depends on the size of the original data and the number of PCs k. The WLSQ optimization adjusts these coefficients iteratively to minimize the weighted residuals between the original and reconstructed data. The standardization conducted in preprocessing is essential for PCA, which seeks to explain the data by projecting the original data onto directions that maximize the variance. Without standardization it appears as if first PC explains $\sim$ 95% of the variance. However, when standardization is applied, the first component explains around 70% of the variance, and the 2nd and 3rd components become more significant.

2.3.3 Generating ice load histories

Synthetic ice load histories are produced through a two-stage process. In the first stage, the Principal Components (PCs) $\mathbf{V}_k$ and the transforms $\mathbf{Z}_k$ that map the chosen PC onto the original data are precomputed using the existing five ice models. The second stage involves the random and linear combination of PC maps, following a normal distribution, similar to the process outlined in the select, shift, and combine method (Section 2.2). The restructured PC map elements from this first stage are then added to the mean ice history at each time step. The artificial data generation process of the second stage is divided into five steps, as shown in Figure 2.6.

Stage 1: Precomputing PC maps

The computation of the PC maps, as described in Section 2.3.2, starts with the full data matrix $\mathbf{X}$ with dimensions (ice models, ϕ, λ, timesteps), where ϕ is 180 degrees and λ is 360 degrees, totaling 64,800 ice loading values on each global grid. However, the WPCA method requires data matrices of size (samples, features). Therefore, WPCA can be applied either to the spatial pattern across the ice models for each time step using $\mathbf{X}_S$ = (ice models, ϕ, λ), or to the temporal pattern across the timesteps for every ice model using $\mathbf{X}_T$ = (timesteps, ϕ, λ). The comparison between spatial and temporal pattern WPCA is provided in Figure B.4 and Table 2.1.
To capture the strongest variance in both spatial and temporal dimensions, a combination of both spatial and temporal correlation in the data is extracted and used in the second stage. However, due to

Spatial Pattern
X_S = (ice models, ice load values)
Results: 21 sets of PCs (one per timestep)
Variance is determined per time step over all ice models
Applied to relative ice load

Temporal Pattern
X_T = (timesteps, ice load values)
Results: 5 sets of PCs (one per ice model)
Variance is determined over all timesteps for every ice model
Applied to cumulative ice load

Table 2.1: Comparison of WPCA methods applied to existing ice model data: assessing spatial and temporal patterns. A visualization of the spatial and temporal data matrices X_S and X_T is provided in Figure B.4. The temporal WPCA is applied to the cumulative ice load because it retains a more evident temporal relation of the deglaciation trend, more detail on the temporal pattern is provided in Figure B.3

the large number of matrix multiplications required, the process is slow for large datasets. Additionally, when applying WPCA to the global grid, the first PCs capture unexplainable features due to the dataset's large value variations where there was ice and zero values elsewhere. This distribution interferes with the principal components, as it causes the eigenvectors to attempt explaining variance in the three separated regions simultaneously. By concentrating on specific regions, the analysis captures more localized trends and patterns that would otherwise be obscured when applying WPCA to the full global map. Therefore, to improve both performance and computational efficiency, WPCA is applied to three geographically distinct areas, as visualized by the red boxes in the bottom center of Figure 2.6. The subsequent step is determining the values of k, which represents the number of PCs that are to be applied. When including a higher number of PCs, the difference between the reconstructed and original data diminishes. The primary advantage of the WPCA method lies in accentuating the data's features and largest pattern, (e.g. deglaciation trend in the temporal dimension, see Figure B.3). Thereby enabling the neural network to identify them more easily. The focus is therefore on selecting only the most prominent features through fewer PCs. The substantial explained variance of the first PC in both the spatial and temporal pattern as illustrated in Figure 2.4 supports the decision to implement $k = 1$ in the random generation process. Adding more components either adds noise or insignificant

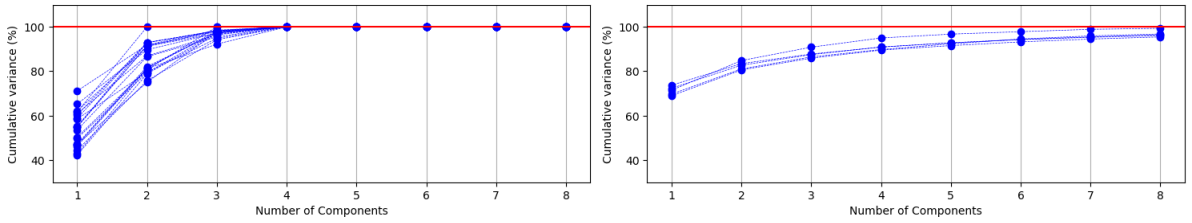


Figure 2.4: Cumulative explained variance of North America for the first eight components. Left figure shows the explained variance for the spatial map of all 21 timesteps. The right figure shows the temporal explained variance for the five ice models

signals, disturbing the signal from the first component. The choice of number of components adheres to the requirements set in Section 2.1.5. For the temporal and spatial pattern, the transform $\mathbf{Z}_k$, representing the projection of the original data onto the component space, and principal components (PCs) $\mathbf{V}_k$ are precomputed for each region and both patterns, example PC maps are provided in Figure 2.5. Therefore, $\mathbf{X}_S$ and $\mathbf{X}_T$ only exist in the WPCA during the first stage.

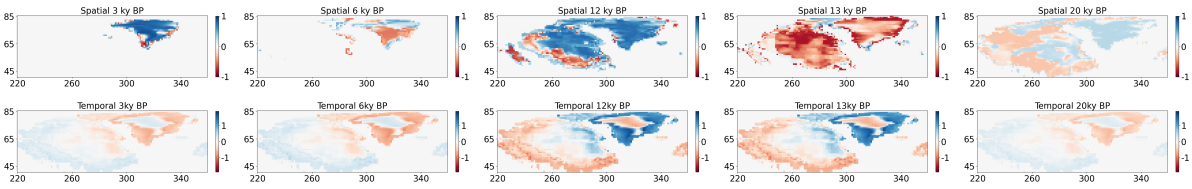


Figure 2.5: PCA maps of North America for ICE-7G of the first component for the spatial and temporal pattern from selected timesteps. The values indicate the importance, where blue is high importance and red low. The spatial maps exhibit stronger variation across the spatial dimension due to the differences between ice models. The temporal maps are consistent across timesteps but vary in magnitude, reflecting the degree of deglaciation. Color scales are constant per pattern. Axes show the co-latitude and longitude on the y- and x-axis, respectively

Stage 2: Data generation

The second stage starts with the results yielded by the first stage. The first stage results in two distinct sets of PC map elements, $\mathbf{Z}_k$ and $\mathbf{V}_k$, for all three regions: one set for spatial patterns and another for temporal patterns. These sets include reconstruction pairs (PC + transform) for each ice model at every timestep, derived through random selection of two ice models (as outlined in step 1 of Figure 2.6). An important distinction is the difference between the original data denoted by $\mathbf{X}$ and the reconstructed data from the WPCA element pairs denoted by $\hat{\mathbf{X}}$. The WPCA datasets are specified using the notation $\hat{\mathbf{X}}_{S,region}(i,t)$ and $\hat{\mathbf{X}}_{T,region}(i,t)$ for spatial and temporal WPCA reconstructions, respectively. In these expressions, $i = 1, 2, ..., 5$ denotes the ice model and $t = 1, 2, ..., 21$ denotes the timestep.

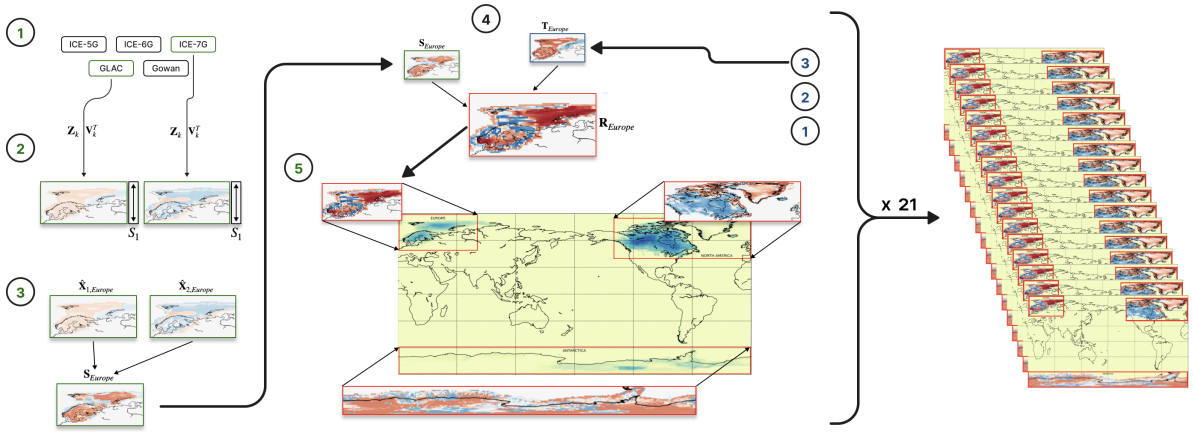


Figure 2.6: WPCA computation schematic showing the process of generating every time step starting from selection of the precomputed reconstruction elements (transform and PC) in (1). Followed by the scaling in (2) and subsequent combination of selection in (3). Steps (1), (2) and (3) are computed for the spatial and temporal pattern, indicated by green and blue respectively, which are combined in (4). The final geographic section altered PC maps for North America (and Greenland), Europe (Fennoscandia) and Antarctica are added to the mean ice load history in (5). The values depicted in the global map show the cumulative ice load of the ICE-6G model at LGM

The ice load histories $\mathbf{R}$ are then generated by first imposing variations on the precomputed PCs $\mathbf{V}_k$ based on scaling factor S_1 (step 2 in Figure 2.6). Then the scaled PC is multiplied with the transform $\mathbf{Z}_k$ and their associated sample mean is added back (from reconstruction step in Section 2.3.2):

$$\hat{\mathbf{X}} = \mathbf{Z}_k(S_1\mathbf{V}_k^T) + \boldsymbol{\mu}_{sample} \quad \text{with} \quad \mathcal{N}_{S_1}(\mu = 1,\ \sigma = 5) \tag{2.3}$$

The magnitude of the standard deviation for scale factor S_1 is determined by the significance of the principal component vector. The vector outlined in $\mathbf{V}_k$ carries specific information associated to the WPCA method. Consequently, it is desirable to have a large variation in S_1, because it is directly multiplied with the entire principal vector. In the following step the reconstructions $\hat{\mathbf{X}}$ of both selected ice models (in Equation 2.5 denoted by 1 and 2) are combined using linear combinations with weights sampled from $\mathcal{N}_\omega(\mu = 1, \sigma = 0.2)$ (step 3 in Figure 2.6). The normal distribution for the linear combinations weights is also derived empirically based on the requirements set in Section 2.1.5, similarly to Method One. Steps 1 through 3 are computed for both the spatial and temporal pattern temporal and spatial PC maps, $\mathbf{S}$ and $\mathbf{T}$ respectively:

Spatial Pattern

$$\mathbf{S}_{region}(t) = \frac{\omega_1\ \hat{\mathbf{X}}^1_{S,region}(i,t) + \omega_2\ \hat{\mathbf{X}}^2_{S,region}(i,t)}{\omega_3} \tag{2.4}$$

Temporal Pattern

$$\mathbf{T}_{region}(t) = \frac{\omega_4\ \hat{\mathbf{X}}^1_{T,region}(i,t) + \omega_5\ \hat{\mathbf{X}}^2_{T,region}(i,t)}{\omega_6} \tag{2.5}$$

The two PC maps of every region are added to the mean ice model history $\mathbf{M}(t)$ using scaling factor S_2 as demonstrated in steps 4 and 5 of Figure 2.6 and described by:

$$\mathbf{R}(t) = \mathbf{M}(t) + S_{2,i=1,2,3}\ \mathbf{S}_{region} + S_{2,i=4,5,6}\ \mathbf{T}_{region} \quad \text{with} \quad \mathcal{N}_{S_1}(\mu = 1,\ \sigma = 1) \tag{2.6}$$

There are six scaling values, two for each region. The standard deviation is again derived empirically from examining the sensitivity of the results with the respect the ice volume requirements. The mean ice history is the average of all ice models per epoch (Figure 2.1). The process depicted in Figure 2.6 is repeated for every time step using the corresponding spatial and temporal first component PC maps for that time step. When steps are completed a time shift is applied similar to the shift in Section 2.2. The time shift is sampled from $\mathcal{N}_{shift}(\mu = 0,\ \sigma = 1)$, again based on visualizing the ice volume sensitivity. The process as depicted in Figure 2.6 is performed on the mean of the cumulative ice history, in contrast to Method One because the temporal correlation had to be calculated on the cumulative ice loading (Figure 2.5) and due to standardization and centering of the data (Section 2.3.2). When the artificial ice load histories have been computed they can be inserted in a mathematical model that calculates RSL changes from ice load history.

The first sub-research question explores the optimal methodologies for generating synthetic datasets to effectively train the GIA emulator. This has been addressed through the application of two complementary methodologies: the "Select, Shift, Combine" methodology (Method One) and the WPCA method (Method Two). The detailed and unique features of the ice models, such as ice sheet extent and ice thickness changes are translated using the "Select, Shift, Combine" methodology, while the underlying spatial and temporal patterns are incorporated through the WPCA method. The Method One methodology, introduces diversity into the datasets applying scalar alterations to the entire ice load history, which inherently includes spatial and temporal components. The "Shift" operation introduces temporal changes, adjusting the chronological sequence of the ice model's features, while the "Combine" process, through linear combination, modifies the spatial distribution or the 'map' of these features. It's important to note that this method does not mitigate the conformity from the ice models; rather, it maintains a certain 'bias' towards existing ice models. As a consequence, overlapping features from the combination might be amplified, creating undesired noise. However, the general outline of the specific ice models is preserved, ensuring the physical integrity of the synthetic data, which was determined to be valuable to the learning potential of the ML-algorithm. In other words, no ice load histories are generated by alterations imposed solely on a single ice model; instead, a combination is always applied. Method Two uses WPCA to incorporate the strongest underlying spatial and temporal patterns within the ice models. This technique translates the nuanced details of the ice models, such as ice sheet extent and ice thickness changes, into a more concentrated and interpretable form, thus enhancing the emulator's potential capacity for feature detection. Together, these methods provide a comprehensive strategy for generating synthetic datasets that are diverse and representative of the inherent patterns within the ice models, therefore both will be used in the remainder of the thesis.

2.4 Numerical Data Generation Method

The numerical method used to compute the dataset outputs, models the redistribution of surface mass, which in turn affects the shape and gravity field of the solid Earth. This process has an instantaneous impact on the distribution of ocean water. The complex interactions between ice loads, the solid Earth, and the global sea level is approached by the Sea-Level Equation (SLE), originally introduced by W. E. Farrell et al., 1976. The SLE specifically models how the sea level at rest responds self-consistently to changes in ice loading and subsequent deformation of the solid Earth. This integral equation is used to analyze RSL changes that occur due to the accumulation and ablation of ice sheets, redistribution of surface mass, and the Earth's response to surface loads. The SLE comprises two primary components: a spatial component accounting for sea level variations and a temporally variable, yet spatially uniform component tracking the time variation of the reference geoid, thus enforcing mass conservation. The principle of mass conservation ensures the total mass of the system, comprising ice sheets and oceans, remains constant and simultaneously accounts for the eustatic sea level change that arises from the relationship between ice sheets and oceans due to meltwater distribution. The general form of the SLE providing the spatiotemporal evolution of the sea level change (RSL output) S is:

$$S(\omega, t) = \mathcal{G}(\omega, t) - U(\omega, t) + C_{SL}(t)^{1} \tag{2.7}$$

where ω represents (θ, λ) latitude and longitude. The first and second term on the right hand side represent geoid variation $\mathcal{G}$, created by dividing the total variation of gravity potential with the reference

[1] Notation not exactly taken from literature, but slightly altered for clarification

surface gravity and U the vertical displacement of the sea floor, respectively. The third term C_{SL} is invoked to satisfy the conservation of mass, because the sea surface does not necessarily remain on the same equipotential surface as the volume of ocean changes (Whitehouse, 2018). This simplified form of the SLE (Equation 2.7) considers both ice and water loads on the Earth's surface. The integration of the ice load history and Earth model within the SLE is not apparent from Equation 2.7, therefore a brief explanation is provided. The ice load history I is added to compute the surface load variation $\mathcal{L}$:

$$\mathcal{L}(\omega, t) = \rho_{\mathsf{i}} I + \rho_{\mathsf{w}} S\mathcal{O} \tag{2.8}$$

where ρ_{i} and ρ_{w} are the densities for ice and water and $\mathcal{O}$ denotes the ocean function, which is equal to one for locations on the ocean and zero on land. The Earth model is added to the SLE by inserting Green's functions G, a mathematical tool used to describe the response to an impulse load, using the Love numbers for a viscoelastic Earth (W. Farrell, 1972). The complete response of the Earth is approximated by three distinct Green's functions: the first, G_ϕ, signifies changes in the Earth's gravitational field; the second, G_u, captures the vertical displacement or deformation of the Earth's surface; and the third, G_S, represents sea level changes in response to a unitary surface load. It should be noted that these Green's functions do not necessarily capture the 'complete' response of the Earth. The Earth's response to changes in ice load is inherently complex and involves many processes that may not be fully captured, such as non-linear viscoelastic deformation, effects of Earth's rotation, gravitational self-consistency, etc. The SLE constituents $\mathcal{G}$, U and C_{SL} are created through spatial and temporal convolutions of the surface load $\mathcal{L}$ with the three respective Green's functions.
From the appearance of Equation 2.8 (presence of S on the right hand side as well) it is derived that the SLE is an integral equation and an iterative numerical approach is required to solve it. Furthermore, the SLE's algorithmic design allows for different representations of the same ice model, leading to significant differences and artificial contributions to sea level estimates, both globally and regionally (Barletta et al., 2013). In the specific version of SLE applied to this study, referred to as the TUDSLE model (Martinec et al., 2018), a number of features have been modified or deactivated Table 2.2 in order to enhance computational efficiency and align with the characteristics of the input data, thus providing a tailored solution for the research problem.

SLE feature	Description
Rotational Feedback	Switched off to simplify calculations and reduce computational complexity without significantly compromising the accuracy of the results. Omission of this feature simplifies the geophysical problem by not accounting for changes in Earth's rotation induced by ice loading/unloading.
Geocenter	Also switched off to simplify calculations. The geocenter correction accounts for changes in the center of mass of the Earth due to redistribution of surface mass. By omitting this, Earth's center of mass remains constant.
Self-Gravity	This feature is turned off, meaning that changes to the geoid (the $\mathcal{G}$ term in Equation 2.7) are ignored. By doing this, the SLE becomes linear and can be solved more straightforward, without the need for iterative methods. The drawback is that this assumes the gravitational field does not change due to mass redistribution.
Maximum Degree LM	Set to 120 to balance computational efficiency and spatial resolution. This parameter determines the highest spherical harmonic degree used in the calculations, which effectively controls the level of spatial detail in the model.
Input Data Resolution	Set to 1x1 degree resolution of relative ice load with steps of 1,000 years, corresponding to a spherical harmonic degree of 180. This resolution provides a detailed representation of ice load history and allows for accurate calculations of sea level variations.

Table 2.2: Settings of the features of the TUDSLE that describe the resolution and detail of the numerical method

The TUDSLE model, is a Fortran 90 code from Schotman, 2007 based on the pseudo-spectral SLE method from Mitrovica et al., 1991, which accounts for time-dependent continental margins and water redistribution near the ice sheet edges (Martinec et al., 2018). This code has been adjusted for different ice loading histories and better convergence when solving for self-consistent sea level and pre-glacial topography. It requires input files with surface load Love numbers for the Earth rheology, ice load history in relative ice thickness, and a topography at specified grid resolutions and timesteps. In summary, the RSL at continental locations, particularly in areas with former or current ice sheets, reflects the combined effects of vertical land movement and geoid variations due to GIA processes.

Chapter 3

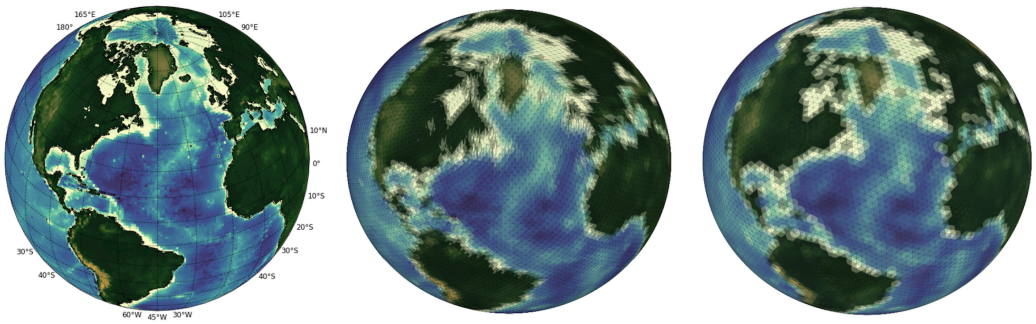
Algorithm Design

The objective of this research is to establish how to design an algorithm that can effectively serve as a surrogate of the GIA model. In this section the research question on specifics of the network will be outlined. The GIA model's input-to-output computation, which transforms ice load into RSL changes and uplift rates, shares structural similarities to a video-to-video regression problem. In this regression scenario, a video comprises a sequence of image frames, with each individual frame representing geophysical signals on a two dimensional global map and the temporal stack represents the epochs between LGM and present. Convolutional Neural Networks (CNNs), a class of deep learning models, are highly effective in image-to-image regression and computer vision tasks in general because of their specialized architecture. The architecture enables the identification and processing of local and high-level patterns and features within images. This ability makes CNNs well-suited for handling the complex geophysical data structures of the GIA forward model.
This chapter, the second part of the methodology, outlines the design process for a CNN that learns to predict RSL changes based on ice load history. Section Section 3.1 discusses the mapping of two-dimensional (2D) synthetic datasets onto a three-dimensional (3D) spherical representation. The second section Section 3.2 will detail the CNN architecture and the different convolution methods that are used to detect the features and reduce dimensionality in the data. Three distinct convolutional filter designs are introduced in this section, with the choice of convolution expected to significantly impact performance and computational expense. In the following section Section 3.3 the physical structure of the algorithm called the architecture is discussed. The architecture entails the configuration of the convolutions and the connections between them. Section Section 3.4 details the process of training and optimizing the network and any design additions implemented to enhance performance. Finally, the section provides an analysis on the data normalization before the data is entered into the network. Although normalization is part of preprocessing and chronologically occurs before any training can be conducted, the sections are switched because the basics of training are required to better understand the implications of the different normalization methods.

3.1 Spherical Representation

In this study, both original and synthetic data are presented using an equiangular 2D latitude-longitude grid of 1 x 1 degree, resulting in a size of (180, 360), containing 64,800 signal values. The synthetic data is computed by solving the SLE in spherical harmonic domain, which is later converted back to the same spatial domain as the provided data. To enhance computational efficiency of training the learning algorithm, this 2D grid is downsampled with bicubic interpolation to a new latitude-longitude grid size of (48, 96). This downsampling reduces the number of signal values to 4,608 while retaining the aspect ratio and ensuring the grid remains easily divisible (a prerequisite described in Section 3.2). Bicubic interpolation is a 2D image resizing technique. It uses cubic splines to interpolate pixel values, by considering the average of a 4x4 grid around each pixel. Although the equiangular grid is commonly applied for its simplicity and ease of use, it is important to address the limitations of representing global signals in a 2D space. One such limitation is significant inaccuracies resulting from polar distortions, an attribute of the equiangular grid, also mentioned in Section 2.3. Convolutional Neural Networks (CNNs) have become the method of choice for learning problems involving planar images. Recently, CNNs have also become more common for spherical data analysis. Examples include omnidirectional vision for drones, autonomous cars, molecular regression problems, global weather and climate modeling, (Cohen et al., 2018) or research into Cosmic Microwave Background radiation (CMB) (Krachmalnicoff et al., 2019). According to Cohen et al., 2018, any application of convolutional networks to a planar projection of a spherical signal is destined to fail due to the space-varying distortions introduced by such a projection. These distortions result in ineffective translational weight sharing and because of

challenges with padding, as well as an excessive focus on the poles due to the imbalance in data quantity between high and low latitudes. The geophysical signals from the GIA model are on the Earth's surface, which is inherently three-dimensional. This discrepancy can result in poor learning and a loss of relationship information. Consequently, spherical representations are explored to capture the spherical nature of the GIA model. There are several methods for representing planar data on a spherical mesh, including, but not limited to, the introduced equiangular latitude-longitude grid, HEALPix (Górski et al., 2005), and icosahedral meshes (Figure 3.1).



(a) Equiangular Grid (Orthographic Projection)

(b) HEALPix Mesh (HEALPix Górski et al., 2005)

(c) Icosahedron Mesh (Trimesh, Dawson-Haggerty et al., n.d.)

Figure 3.1: Spherical representations of the 1 x 1 degree Earth topography (for visualization) for three different spherical grids and associated projection method. The HEALPix mesh is created with an $nside = 32$ (12,288 pixels), while the icosahedron mesh is generated with five subdivisions (10,242 vertices). The HEALPix projection appears more distorted because the data of (b) and (c) is visualized by calculating the mean of the pixels/vertices connected to each face for coloring

Orthographic projection maps the equiangular latitude-longitude grid to Cartesian coordinates. However, Cartesian coordinates are sub-optimal for computational purposes, because it is computationally expensive to have three independent variables to represent a point in space. The HEALPix mesh is a specific partitioning of a sphere aimed at more equal-area pixels to represent a more accurate representation of spherical data. The mesh is coarser on the equator and finer near the poles (2D HEALPix grid provided in Figure B.5). The mesh resolution (number of pixels) is defined by the $nside$ parameter; the number of divisions along the base pixels. The icosahedron, similar to the HEALPix, is a regular polyhedron with equilateral triangular faces. The standard size of an icosahedron consists of twelve vertices and 30 faces. Each vertex (node) in the mesh is connected to one or more adjacent vertices by edges, which are the line segments that connect the vertices. The edges form the boundaries of the triangles in the mesh, and the triangles themselves are the faces. This icosahedron is the starting point for subdivision. The subdivision is then applied using a recursive subdivision technique to create finer icosahedral meshes. Recursive subdivision is performed by splitting each face into four smaller triangles. The number of vertices at each subdivision level L of the icosahedron mesh can therefore be computed as $10 * 4^{(L-1)} + 2$. Using this formula, the number of vertices is built up from the original twelve vertices as 42, 162, 642, 2652, 10242, etc.

If the 2D data does not have the same number of pixels or nodes as the desired spherical mesh, interpolation is necessary during the mapping process, not only to reconcile the differing quantities of points but also to account for variations in area, even when the points match in number. Multiple mapping methods are possible, the regular grid can be represented in other projections and interpolation can be implemented. To compare the best and most common methods a standardized RSL of a single time step from the downsampled grid is mapped to a spherical mesh and back. The mapping performance is measured by calculating the mean absolute error (MAE) (Equation 3.1) between the RSL map before and after mapping Table 3.1,

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{3.1}$$

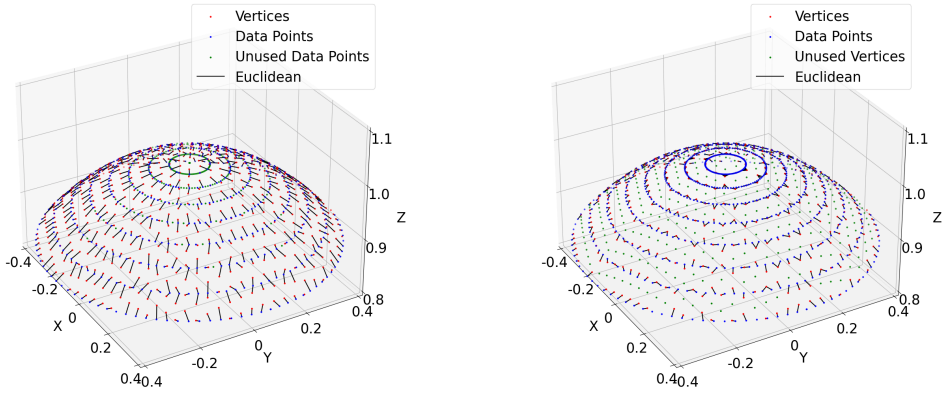
where n is the total number of data points, y_i is the actual value, and $\hat{y}_i$ is the predicted value. The absolute difference between the actual and predicted values are summed up and then divided by the

total number of points to get the MAE. If the mapping is performed with nearest neighbor distances the $cdist$ from SciPy (Virtanen et al., 2020) is implemented. The number of coordinates in the mesh always significantly exceeds the number of coordinates in the equiangular grid.

Grid Type	Interpolation	Description	Mesh Resolution	MAE
Regular Cartesian	Nearest Neighbor	Pairwise distances between regular grid and spherical mesh are found by finding nearest-neighbor indices in Cartesian coordinates	Icosahedron with 10,242 vertices	1.916
Regular Spherical	RectBivariateSpline	Interpolates between regular spherical grid and spherical mesh using piecewise bivariate spline function for a smooth two-dimensional interpolation in rectangular domains.	Icosahedron with 10,242 vertices	15.643
Spherical Harmonics	Nearest Neighbor	Transform the regular grid in spherical coordinates to spherical harmonic coefficients and evaluate spherical harmonics surface at mesh coordinates	Icosahedron with 10,242 vertices	15.513
Regular Cartesian	Nearest Neighbor	Pairwise distances between regular grid and the spherical mesh in Cartesian coordinates are calculated by finding their nearest-neighbor indices	HEALPix with 12,288 pixels	1.841
Regular Spherical	Ordinary Kriging	Ordinary Kriging is a geostatistical interpolation technique that predicts values at specified locations using a weighted linear combination of nearest neighbors	Icosahedron with 10,242 vertices	21.358
Regular Cartesian	Nearest Neighbor	Pairwise distances between regular grid and spherical mesh are found by finding nearest-neighbor indices in Cartesian coordinates	Icosahedron with 40,962 vertices	0.491

Table 3.1: Mapping methods used to transform the geophysical signal on the regular grid to a spherical mesh. The MAE is calculated for RSL on one downsampled spatial grid at LGM, because values peak at LGM and consequently the magnitude of the mapping error. Visualizations of the mapping errors are provided in Figure B.2

Based on the results shown in Table 3.1, the decision was made to implement pairwise distance methods exclusively with the fifth subdivision icosahedral mesh containing 10,242 vertices. Although the HEALPix mesh is slightly better at mapping, the lower error does not compensate for the increased computational complexity that stems from its irregular structure (Figure B.5). Furthermore, for nearest neighbor interpolation, the mapping error can be reduced through higher icosahedron subdivision, because it increases the proportion between the given data grid and spherical mesh. This error reduction occurs around the poles, however, it simultaneously introduces more duplicates near the equator. This redundancy arises because these duplicates, sharing the same data points, are still used in ML-algorithm training. In addition, almost quadrupling the number of vertices in the spherical mesh significantly increases the computational cost. This duplicates issue associated to nearest neighbor interpolation is best explained using the illustration provided in Figure 3.2.



(a) Mapping from 2D to 3D

(b) Mapping from 3D back to 2D

Figure 3.2: Visualization of the nearest neighbor mapping issues around the northern pole, transitioning from 2D to 3D and vice versa. The mapping is indicated by plotting the Euclidean distances. The figures highlight the density disparity between the spherical mesh and the equiangular grid through unused data points (a) and unused vertices (b)

When mapping from 2D (spherical coordinates) to 3D (Figure 3.2a) every vertex is assigned an equiangular grid index. Because there are 4,608 data points and 10,242 vertices some data points at the higher latitudes go unused, while around the equator data points are mapped up to six different vertices. Conversely, during the reverse mapping (Figure 3.2b) there are unused vertices, resulting in redundancy surrounding the equator and loss of information near the poles. The information loss extends to the lowest 'latitude ring' shown in Figure 3.2. After reverse mapping, the Euclidean distance error disappears, because the nearest distances between the equiangular grid and spherical mesh are intrinsically defined and invariant within the grid structures. It's worth noting that because the SLE is solved in the spherical harmonic domain the output can be mapped to any another spatial domain. If a

spherical mesh is selected this can circumvent the need for mapping and interpolation altogether. This approach would demand some programming effort from altering the SLE, which potentially slows down the calculation.

3.2 Convolutional Neural Network

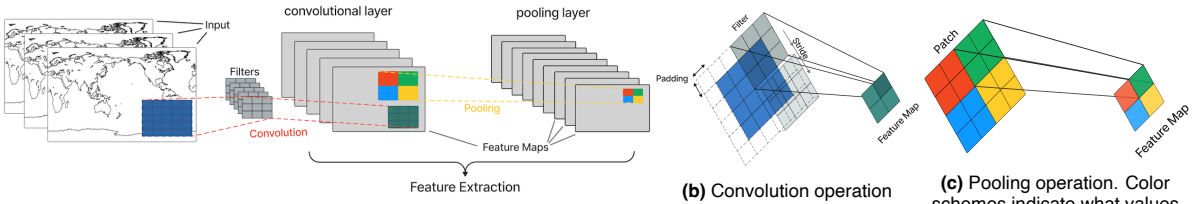
A Convolutional Neural Network (CNN) is a class of deep learning models specifically designed for tasks related to computer vision analysis, including image-to-image regression, object recognition, and image segmentation. The core component of a CNN is the convolutional layer, which applies filters to extract local patterns and features from the given data. Convolution constitutes a special linear operator that transforms the input that is passed though the network. Filters, also known as kernels, are small matrices that convolve with the data and perform element-wise multiplication followed by a summation, enabling the detection of various features (e.g. edges, shapes, etc.). In the case of ice load history and RSL changes these features are primarily slopes and sharp edges. Slopes, for example, are temporal trends such as the deglaciation or rising sea level and sharp edges are the ice sheet extent and coastlines. Some visualizations of the filters from a trained conventional CNN on the equiangular grid are provided in Figure B.6. This section discusses essentially three types of convolutional methods of which two are explored in designing the GIA emulator. The first convolutional method is the more common two-dimensional convolution, with a filter called "Conv2D," which is discussed in Section 3.2.1 and is applied on the equiangular map. The next two methods involve filters designed for processing data on non-Euclidean domains like icosahedral meshes: geodesic convolution Section 3.2.2 and the spectral convolution Section 3.2.3. A geodesic is the shortest path between two points on a curved surface, as measured along the surface and spectral refers to the domain the convolution operates in. The three methods share similarities and differences across a variety of aspects, therefore an initial overview on selected categories related to the GIA problem data structure is provided in Table 3.2.

Convolution	Domain and Format	Local Neighborhood	Convolution Operation	Complexity	Application
Conventional	Spatial, Regular grid (e.g., 2D images, 1D signals)	Convolution on original 2D/1D data with pixels of fixed -size neighborhoods	Sliding filter over input data, computing inner product	Low	Image processing, object detection and segmentation
Geodesic	Local Geometric, Surface meshes	Convolution on local mesh neighborhoods defined by surface geometry (e.g., tangent plane)	Convolution in intrinsic space of each vertex in surface mesh	Medium	3D shape analysis, biomedical imaging (MRI scans of brain)
Spectral	Spectral, Irregular data structures (e.g., graphs, meshes)	Global neighborhood: graph pixel matrix defined by spectral transform method	Independent approximation of graph structure with a filter in the spectral domain	High	CMB analysis, climate modeling, medical imaging (PET scans)

Table 3.2: Overview of three convolutional methods based on distinctive criteria. Local neighborhoods describe how the geophysical signal's connection are interpreted by the filter. Complexity represents difficulty of implementation and computational expense

3.2.1 Conventional convolution

Conventional convolution is a fundamental operation for image processing tasks (Goodfellow et al., 2016). CNN primarily focusses on feature extraction from input data, with convolution and pooling as their key operations. Convolution operations detect features and pooling intensifies those features while reducing the dimensions. The convolution operation is performed at each location of the input grid, producing a new grid known as the 'feature map', as illustrated in Figure 3.3a. In the training phase of a CNN, the filters' learnable parameters are updated to improve feature detection. The learnable parameters in a CNN are weights and biases. Weights can be thought of as influence factors assigned to the input features to generate an output prediction, while biases are constants that provide an additional degree of freedom to the weights, adjusting the output along with the weighted sum of the inputs. The features identified by these updated filters are then integrated into the feature map. A feature map is created in steps, sliding across the grid to systematically convolve with the input data. The size of the feature map depends how the filter slides over the data, defined by: the filter size, stride and padding, visualized in Figure 3.3b. The distance between the sliding steps is called stride. Padding is the number of pixels added to the edges of the input image to preserve the spatial resolution of the feature maps in order to retain the spatial dimensions.



(a) Feature extraction process of a CNN; convolution and pooling on two-dimensional input data

(b) Convolution operation showing the feature map size and 'sliding window' process

(c) Pooling operation. Color schemes indicate what values are pooled

Figure 3.3: Conventional convolution schematics in two dimensions for feature extraction and sliding window process

The number of filters implemented in a convolutional operation determines the number of output channels, also referred to as feature maps. Consider an input tensor of dimensions (100, 100, 20) for instance, where 20 represents the number of input channels (i.e., different images). Every filter in the convolutional layer interacts with all input channels, generating an output channel for each filter. Each filter consists of a unique set of weights, one for each input channel, that convolve across their corresponding channels and the results are then summed to form an output channel. If 25 filters are deployed in this layer, 25 output channels (or feature maps) will be created. Consequently, the output of this operation, which serves as the input to the subsequent layer, would have dimensions of (100, 100, 25).

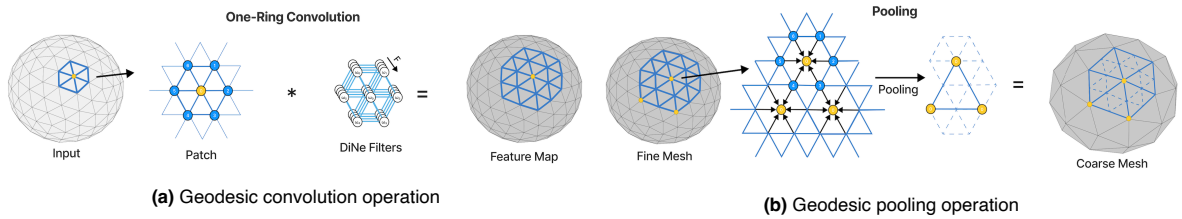
Convolutional layers are typically followed by a pooling operation. Pooling is used to reduce the spatial dimensions of the feature map while preserving its most important information. The spatial dimension is reduced by a deterministic operation of taking the maximum or average value within a patch as shown in Figure 3.3c.The choice of filter size and other parameters influences the features that can be extracted from the input data. A larger filter size can capture more global information, while a smaller filter size focuses on local details. The operations of convolution and pooling using filters exists for all three convolution methods, but are very different in their application.

3.2.2 Geodesic convolution

The term "geodesic convolution" was introduced by Zhao et al., 2019 to describe a novel convolutional method designed to detect features based on geodesic connections between points on a sphere. This geodesic approach has roots in medical imaging, where many structures, due to their spherical topology, can be understood as existing within a manifold space. One such example is the human brain's cortical surface, which is highly irregular due to its convoluted arrangement. This irregularity presents unique challenges for feature detection and pattern recognition. The methodology proposed by Zhao et al., 2019 involves mapping and sampling the complex cortical surface of the human brain to a smoother, more regular structure; an icosahedral mesh. This mesh, which is equidistant between nodes and contains equilateral triangular faces, is then subject to geodesic feature extraction. The geodesic convolution in Zhao et al., 2019 is used to accomplish two key tasks: classification/segmentation for cortical surface parcellation, and regression in cortical attribute map prediction. The process of CNNs refers to the task of identifying distinct regions within an image. In this context, these regions could correspond to different anatomical or functional areas of the brain's cortical surface.

The proven ability of the geodesic convolutional method to perform regression makes the method very interesting for application on the GIA forward model. Though brain cortical surfaces and geophysical signals don't strike apparent similarities, their equiangular projection and spatial patterns do. The convolution filter that was designed, named the Direct Neighbor (DiNe), considers the sphere as an equilateral triangular structure like the icosahedral mesh. The points on this sphere correspond to the vertices of the mesh, and the geodesic connections form the edges of the icosahedron. The DiNe filter is defined by the nearest neighbors to each vertex, with six vertices of equal distance forming a ring around a central vertex in an icosahedron. The direct neighborhood can also be depicted as a square map, analogous to traditional convolution on an image grid. However, this projection strategy can introduce feature distortion and require re-interpolation, consequently increasing computational complexity and reducing accuracy (Zhao et al., 2019). The definition of the DiNe filter aligns with the expansion and contraction (or subdivision) process of an icosahedron, wherein vertices contribute to or aggregate from the information of their direct neighbors at each iteration process. In the application of the DiNe

convolution filter, the subsequent steps involve surface convolution and pooling within the spherical space. The feature extraction that constitutes these two operations is illustrated in Figure 3.4.



(a) Geodesic convolution operation

(b) Geodesic pooling operation

Figure 3.4: Geodesic feature extraction process. The DiNe-filter size is the equivalent to the patch shown in Figure 3.3b. The number of filters F equals the number of feature maps in the icosahedron. Each filter is applied to every vertex of every timestep, representing every connection in the entire mesh using the weights in the filters. In the pooling operation average-pooling is applied to the six neighboring vertices to pool four triangles into one. Illustration based on Zhao et al., 2019

The pooling operation depicted inFigure 3.4b applies ”nearest neighbor interpolation” to downsample the feature maps. This contrasts with conventional 2D convolution methods that use average or max pooling over rectangular patches, because that introduces distortions when applied to a spherical surface. These distortions come from the mismatch between the rectangular pooling filters and the spherical topology, which results in a loss of local information. The ’one-ring’ structure the pooling is based on mitigates this problem. Because of the DiNe-filter’s structure the method can not be applied to the HEALPix mesh as the distances and connections between the grid points are not consistent across the sphere.

3.2.3 Spectral convolution

This section provides a detailed exploration of the principles of spectral convolution, a key component of Graph Convolutional Networks (GCNNs). The application of convolutional networks on spherical data through a graph-based discretization is a subject of Graph Signal Processing (GSP) (Defferrard et al., 2020), often referred to as Graph Convolutional Networks (GCNNs). The principles of spectral convolution encompass domain transformation, the use of spectral filters, filter size consideration, and the execution of pooling operations. A graph, in this context, is a mathematical representation where objects (or value points) are represented as vertices, and their relationships are denoted by edges linking them together (Defferrard et al., 2020). Spectral convolution is advantageous for handling irregular data and complex graph structures (such as the HEALPix and icosahedron mesh), as it considers the entire globe when executing convolution operations. This global approach sets it apart from traditional 2D CNNs, which also operate on the global map but do so in incremental steps. This attribute is particularly useful when dealing with phenomena like the GIA signal, which is characterized by long-wavelength processes, such as glacial rebound and deglaciation.

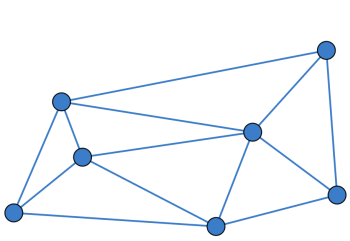
Regular convolution operation involves the sliding window technique using a filter, applying a convolution at each step. This technique, however, becomes significantly more complex when applied to irregular 3D structures like graphs due to the lack of a natural, fixed coordinate system. The proposed spectral method does not try to approach this 3D problem, but instead transforms the graph to the spectral domain (i.e., wavelengths and amplitudes) and uses a GSP substitute for the conventional 2D filter. In this spectral domain, the graph structure is typically represented by a matrix, which captures the relationship between nodes (vertices) in the graph. An example of this matrix representation of a graph can be found in Figure 3.5b, where the vertices (or nodes) are depicted by the red diagonal line akin to an identity matrix, while other colors indicate the connectivity to other vertices. Convolution operations in the spectral domain are performed on the eigenvectors and eigenvalues derived from the eigendecomposition of this matrix. This process enables feature extraction that respects the original 3D topology of the graph. The operation performed on the eigenvectors is analogous to the filter application in conventional CNNs, therefore it is further referred to as the (spectral) filter. The transformation from graph via matrix to eigenvectors (spatial to spectral), is the property that allows for irregular mesh data. Various methods are available for this domain transformation, with the Spherical Harmonic Transform (SHT) and the standard Laplacian matrix being the most appropriate for the thesis problem. Further, a variety of filters can be applied to the eigenvalues and eigenvectors of the graph

Laplacian, such as the Fourier basis approach, Heat kernel, Chebyshev polynomials, and Cayley filters. A detailed description of these spectral filters and their application is provided in Table 3.3.

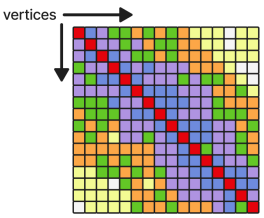
Graph Structure	Filter	Convolution	Description & Learnable Parameters
Laplacian (discrete)	Fourier basis	Multiplication of the Fourier coefficients (eigenvalues) with the filter's spectral corresponding value (amplitude) in the spectral domain, modifying the signal's frequency components	The Fourier basis approach uses the Fourier basis as a set of basis functions in the spectral domain. The learnable weights correspond to the amplitudes of the basis functions
	Heat kernel	Applies an exponential decay function to the Laplacian eigenvalues, attenuating higher frequencies in the graph signal	The Heat Kernel is a low-pass filter that dampens higher frequency components and smoothens the graph signal. The learnable weights are the decay rates applied to the eigenvalues
	Chebyshev polynomials	Filter is represented as a linear combination of Chebyshev polynomials. Each polynomial is scaled by its corresponding coefficient and then summed together	Chebyshev polynomials are used to approximate the spectral response of certain filters. They form an orthogonal basis in the spectral domain. The learnable weights in this case are the coefficients of the polynomials
	Cayley filters	Multiplication of the Cayley transform of the eigenvalues of the Laplacian with the filter coefficients	Cayley filters operate directly on the eigenvalues of the Laplacian matrix and are designed to be stable and localized in the spatial domain. The learnable weights are the coefficients used in the Cayley transform
Spherical Harmonic Transform (SHT) (continuous)	Gaussian smoothing	The spherical harmonic coefficients (eigenvalues) are multiplied by a filter function (e.g., Gaussian), modulating the spectral content based on the filter design reducing frequency noise	The SHT operates solely on functions defined on a sphere, providing spherical harmonic coefficients for filter operations. The learnable weights in this case correspond to the parameters of the Gaussian (e.g., the mean and standard deviation).

Table 3.3: Overview of common methods for transforming data from the spatial domain to the spectral domain and applying filters in the spectral domain for convolutional operations. One key benefit of performing these operations in the spectral domain is that convolution becomes simple multiplication, which is computationally efficient

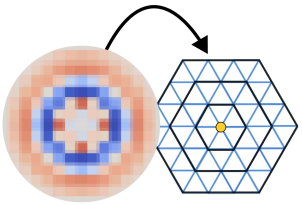
The methodology adopted for representing the graph structure divides the spectral convolution methods into two main categories: the Laplacian matrix and the Spherical Harmonic Transform (SHT). The Laplacian method involves the discrete graph Laplacian, represented as L = D - A. Here, D = *diag*(d_1,...,d_n) stands for the degree matrix, a diagonal matrix that features the degree of each vertex on its diagonal. A = [Wij] is the adjacency matrix, a square matrix where each element represents the weight W_{ij} of the edge between the vertices x_i and x_j (Hexmoor, 2015). The Laplacian matrix captures the graph structure and encodes the relationships between nodes. In contrast, the SHT represents a function on the sphere in terms of a basis of spherical harmonics, recognized as the eigenfunctions of the Laplace operator on the sphere (Driscoll et al., 1994). Unlike the Laplacian method, the SHT does not require an eigendecomposition, marking a fundamental difference in their approach. However, implementing SHT is associated with limitations, specifically the significant computational costs that arises from its complex functions, especially for higher orders. Thus, to simplify the process, the spectral convolution method used in this study is sourced from the DeepSphere package (Defferrard et al., 2020). More specifically, an implementation designed for PyTorch, a Python package that offers tensor computation and is an ideal fit for deep neural networks (Paszke et al., 2017). The DeepSphere package transforms graph structured data on the combinatorial Laplacian matrix, and uses the K-degree Chebyshev polynomials as a filter on the eigenvalues of the Laplacian. The two most important distinct features of the DeepSphere spectral convolution process, the domain transformation and the spectral filter application, are detailed in Figure 3.5.



(a) Graph structure in 2D showing the nodes and connections (vertices and edges)



(b) Laplacian matrix displaying connectivity between vertices



(c) Spatial visualization of third order Chebyshev spectral filter's 'receptive field' for a specific node on the graph (filter image from Levie et al., 2017)

Figure 3.5: Spectral feature extraction process. The input data is transformed to the spectral domain using the Laplacian matrix of the entire graph structure

Using terminology from the regular convolution method this Chebyshev polynomial is regarded as the 'filter'. Since the polynomial approximates the entire graph Laplacian, and thus the entire global map, the spectral filter as shown in Figure 3.5c does technically not exist. Rather, it is a visualization of the filter in the context of the graph structure. The image is a heatmap showing the connectivity from a specific node to its neighboring nodes derived from the entire Chebyshev polynomial evaluated on the graph. The color intensity denotes the influence. Using the visualization from Figure 3.5c and the properties of the Chebyshev polynomial, the filter size can be defined. The spectral filter is applied to the entire Laplacian, but a Chebyshev polynomial of order K, has the effect of considering information from nodes up to K steps or rather rings away. In Figure 3.5c the filter is from a third order (K=3) polynomial as indicated by three black rings surrounding the fictional center node in yellow. A higher order polynomial will consider nodes that are further away. This is analogous to how the filter size in a standard convolutional layer determines the spatial extent of the convolution operation.

Similar to conventional, spectral convolution also uses a pooling operation designed to decrease the spatial dimension of the input data. This process effectively reduces computational complexity and mitigates the risk of the neural network excessively fitting the training data. The DeepSphere package performs a relatively simple form of pooling: it retains only a subset of nodes from the input data. This means that the pooling operation is not really computing any aggregate information from the neighborhood of each node, as in conventional pooling operations or geodesic convolution (max or average pooling). Instead, it reduces the number of nodes in the graph by keeping only a subset of them, in the structured order defined by the specific mesh (e.g. icosahedron graph).

3.2.4 Summary

While conventional 2D CNNs have proven to be efficient for image processing, their application will not be extended further in this study. Apart from the mentioned advantages of spherical representations over planar in Section 3.1, there are other limitations inherent to 2D CNNs. These networks struggle with maintaining the spatial dependencies between distinct locations on the grid, an issue that becomes prominent when the input and output are different. Even more when some high-level spatial features should still be preserved. Consequently, 2D CNNs have an inconsistent performance across regions because they can not capture the physical relationships between different geographical locations. Another limitation of 2D CNNs lies in their inability to handle wraparound continuity. Since the GIA-signal is inherently 3D the left and right edges (as well as the top and bottom edges) are actually adjacent and connected. This is not considered in 2D CNN, resulting in discontinuity at the map boundaries and poorer performance near the edges due to problems with padding. Furthermore, given the complexity and dataset size of the GIA problem many convolutions are likely required and since the problem is regression also many deconvolutions. Deconvolutional layers, typically used for upsampling feature maps, can lead to 'checkerboard' artifacts due to uneven filter overlap (Odena et al., 2016). This uneven overlap results in an alternating pattern that resembles a checkerboard. Geodesic convolution and spectral convolution operate differently from conventional CNNs. The geodesic method applies convolutions in the graph's geodesic space, consistently considering a predefined one-ring neighborhood. Given the spherical nature of the grid, no moving window technique is necessary. With a constant stride and no padding, the geodesic convolution method effectively mitigates issues of filter overlap. In the spectral method, the graph structure is transformed into the spectral domain. It avoids the local neighborhood definition and the stride issue, as it considers the entire structure at once during convolution. Therefore, both these methods avoid the core issue that often leads to 'checkerboard' artifacts in 2D CNNs. However, there is also a shared limitation among conventional, geodesic and spectral convolution methods, which is their inherent lack of temporal context. By default, the three methods treat each frame as an individual map, lacking the capacity to encode temporal dependencies between different frames. This is particularly problematic for tasks involving video-to-video regression, where understanding the sequence and temporal evolution of frames is critical. Although temporal context can be replicated or approached by adding depth to the network architecture, it presents additional complexities and considerations. The concept of attempting temporal context within the network architecture will be further explored in Section 3.3.

The feature extraction of the spectral and geodesic methods is only half the equation, the remaining abilities are derived from the structure of the learning process; the network architecture.

3.3 U-Net Architecture

The U-Net architecture, used by both the DeepSphere package and geodesic convolution, is applied to analyze and learn features and patterns. Originally developed for biomedical image segmentation, this architecture was introduced by Ronneberger et al., 2015. This section is divided into two parts; the first part explains the conceptual framework of the U-Net configuration, while the second part provides a detailed discussion on the CNN and added components that make up the layers of the U-Net.

3.3.1 U-Net Configuration

The U-Net architecture constitutes three integrated components: (1) a contracting path designed to capture context, paired with (2) a symmetrical expanding path that enables precise localization of the information and (3) special connections between them. Localization refers to determining the locations of specific features within an image. Localization is important in image regression and segmentation because the model needs not only to recognize certain features but also to pinpoint their exact position. The context is captured by detecting features through compact representation of the data. The compaction is achieved by decreasing the feature size and increasing the channel width. Features are the datapoints, in this case the geophysical signal on a global grid and the channels are the timesteps in the input. The channel width is equal to the number of timesteps. The intermediate layers also have channel width, but that is determined by the number of filters and no longer equals the timesteps. The expanding path uses the compact information from the contracting path and gradually recovers the spatial dimensions that were lost during contraction. The architecture consist of layers referred to as the 'layer depth', and it is this arrangement of layers and paths that gives the U-Net its name. A detailed graphic version of the U-Net architecture, which forms the basis for the emulator, is illustrated in Figure 3.6. A full page high resolution version of Figure 3.6 is additionally provided in Appendix A.

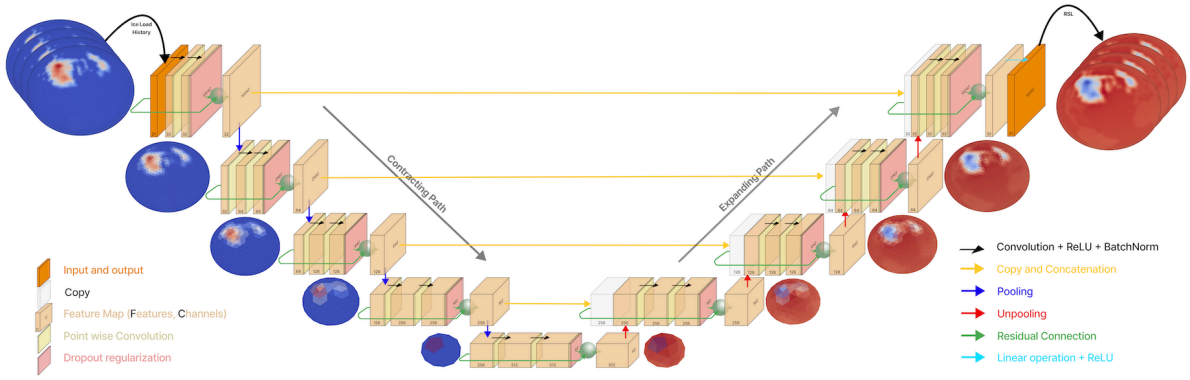


Figure 3.6: U-Net architecture configuration and components. The images illustrate the coarseness of the icosahedral meshes through the layers as a product of the pooling and unpooling, where the RSL output is colored negative for visualization. The input of the network is cumulative ice loading (not changes) so it corresponds to RSL. The point-wise convolution and dropout regularization are displayed as thin rectangles, but have no channel width as they are operations similar to arrows and not feature blocks. Note: the spherical surfaces are all equal size in reality, they are only visualized in different sizes

This expanded version of the original architecture as proposed in Ronneberger et al., 2015, introduces several improvements aimed at better capturing spatial and 'temporal' relationships in the data. The information in the U-Net structure progresses in such a way that the spatial dimension of the feature maps is reduced, while the channel width of these maps expands. This is a typical characteristic of CNNs, where the reduction in spatial dimensions allows the network to focus on the more abstract features rather than localized details, while the increase in channel width captures a wider array of features from the data. Each feature map typically specializes in detecting a specific kind of feature (learned through filter weight updates) so having more filters enables the network to detect more complex and diverse patterns, capturing more high-level details, or in this case the long-wavelength GIA signal. However, these feature maps do not inherently possess a temporal component so the long-wavelength GIA signal is not intrinsically learned. Instead the CNN is generally used to learn spatial hierarchies of features from the data. Still, convolutional layers can also learn filters that respond to patterns across input channels, effectively enabling the network to learn temporal patterns.

The channel width is constrained by the pooling and unpooling operations and the recursive subdivision of the icosahedron mesh. Consequently, the initial channel width is either 24 or 32 (larger than the original of 21 timesteps), ensuring divisibility for the subsequent operations. The deeper layers follow a pattern of doubling the channels in size, resulting in 24, 48, 96, etc. The combination of the recursive icosahedron mesh subdivision steps (e.g., 12, 42, 162, 642, 2562, 10242) with the increase of channel width equals halving of the number of trainable parameters with each layer. The contracting path of the U-Net follows the conventional architecture of a CNN, as shown in Figure 3.3, Figure 3.4, and Figure 3.5. The hierarchy changes in layer depth are performed by pooling in the contracting path and unpooling in the expanding path. The specific methods for pooling (decreasing spatial resolution) and unpooling (increasing spatial resolution) depend on the applied convolutional method. Unpooling in the geodesic method is an upsampling operation that fills the new vertices in the finer mesh using transposed 1D convolution by averaging the features of the neighboring vertices from the coarser mesh, essentially the opposite of process in Figure 3.4b. In the spectral method unpooling is performed by preserving the existing pixel values and padding the additional pixels with a constant value, thereby simply extending the grid without adding information from the coarser mesh.

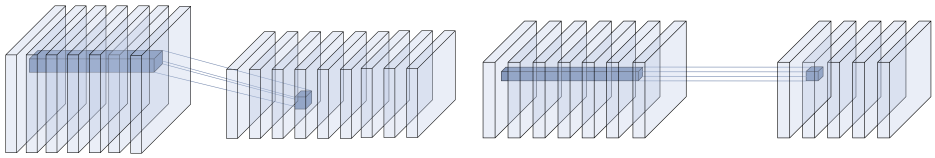
The U-net architecture contains a third integral part that works together with the contracting and expanding path, the skip connections, represented by the yellow arrows in Figure 3.6. Skip connections are designed to address the information loss at each layer in the network. Typically in CNN each layer feeds into the next, meaning that the information flows in one path. This is acceptable for many tasks, but not for learning high resolution details (like in image regression). As information progresses deeper into the network, the input is downsampled (pooling) to extract higher-level features (temporal)[2], but some of the finer details are lost (spatial)[2].Skip connections resolve this loss through a direct path or connection from earlier layers to later layers, effectively ”skipping” the intermediate layers such as the bottleneck. Skip connections are therefore uni-directional from contracting to expanding path. The skip connection is performed prior to pooling, by copying the final feature block in the contracting path and concatenating it with the first feature map in corresponding expanding layer. This transfer is indicated in Figure 3.6 by the yellow arrows combined dashed transparent feature blocks. Thus, while the expanding path is recovering information it gets help from the skip connection. It should be noted that the skip connection is an integral part of the architecture similar to the contracting and expanding path and is always followed.

3.3.2 U-Net CNN Components

The remainder of the architecture can be best understood by exploring the building blocks that constitute its structure. In Figure 3.6, nine building blocks are visible. Each U-Net layer contains two building blocks; one in the contracting path, another in the expanding path; with the exception of the deepest layer, which contains only a single building block. The four blocks in the contracting path combined are known as the encoder, the four blocks in the expanding path the decoder, and the deepest block is called the ’bottleneck’. The encoder-decoder terminology is borrowed from the Autoencoder (AE), another common neural network architecture. The main difference is that the AE is designed to create the bottleneck and the data on either ends is the same dataset. A U-Net block comprises multiple operations and recurrent convolution applications. These processes are thoroughly detailed in Figure 3.6. The input and output in the U-Net are represented by the fully colored orange rectangular solids, while the semi-transparent orange rectangular solids symbolize the feature blocks. Feature blocks (part of the U-Net block) have two dimensions described by two values in the schematic. One value at the bottom indicates the channel width (number of feature maps) and another on the side signifies the spatial dimension or feature size, which corresponds to vertex count of the icosahedron image shown in their respective layer. The original dataset consists of 21 timesteps. Inside the CNNs, these timesteps are replaced by channels, because channels represent the number of feature maps defined by the number of filters. Each convolutional layer typically involves spectral or geodesic convolution, followed by batch normalization and an activation function. Batch normalization and the activation function will be discussed in Section 3.4.2.

[2]While these annotations offer a useful insight into the workings of U-Net, it is not entirely accurate in its portrayal of information loss during the downsampling process.

In the Figure 3.6 U-Net architecture, the total number of convolution operations and the number of convolution operations per U-Net block varies depending on the specific application. In the schematic, each block contains five convolutions; two geodesic or spectral convolutional layers (black arrows), two point-wise convolutions (yellow thin rectangular) and one convolution part of the residual learning (green sphere). Convolution operations in the U-Net are either standard or point-wise as graphically depicted in Figure 3.7.



(a) Standard Convolution **(b)** Pont-wise Convolution

Figure 3.7: Comparison of standard and point-wise convolutions. In standard convolutions, filters are applied across all input channels, allowing impact on both spatial feature size and channel depth. Point-wise convolutions, with a 1x1 filter, allow alteration of the channel depth while preserving spatial feature size

Standard convolution applies a set of filters across all input channels, mixing the channel information together, while point-wise convolution applies a 1x1 convolution, operating only across channels, while treating the spatial dimensions independently. The three convolutional methods described in Section 3.2 are all standard convolutions and represented as black arrows in Figure 3.6. Point-wise convolutions are not followed by batch normalization or activation. The combination of both types of convolutions allows the model to learn spatial and cross-channel features, respectively. In the context of U-Net, the cross-channel techniques allow the network to mimic a form of temporal learning. This is similar to the 'temporal learning' achieved by the channel batch normalization applied after every geodesic and spectral convolution (part of the black arrow). Again it is important to understand that this is not the same as true temporal learning, such as what might be achieved with recurrent neural networks (RNN) or transformers.

Residual learning, introduced by He et al., 2015 and adopted in the ResNet architecture (Targ et al., 2016), is a technique designed to tackle performance degradation in deep CNNs, like those found in U-Net blocks. Rather than learning the full transformation from input to output within a block, residual learning focuses on the 'residual' or the difference between the input and output. This is achieved by adding the unchanged input directly to the output of the block via a residual connection. Such an approach simplifies learning within the block and mitigates issues like the 'vanishing gradient' problem, a technical term used in machine learning to describe the problem where the network weight updates become excessively small, obstructing the learning. The mechanism of learning residuals is similar to the function of skip connections between U-Net layers, providing an optimized learning path in the network's forward progression. In the Figure 3.6, the residual connection is indicated by a green arrow, leading to a point of integration denoted by a green sphere.

The final step in the network is the output layer, which consists of a linear operator and an activation function, represented by the cyan arrow. The purpose of the linear layer, also fully-connected layer, is to transform the learned features into the form used to make predictions. Not mentioned in the discussion is the dropout operation represented by the pink thin rectangular, it will be discussed along with batch normalization and activation in the section on hyperparameters Section 3.4.2.

3.4 Network Training

This section outlines the process of network training and optimization, divided into four subsections. Section 3.4.1 discusses the fundamental learning mechanisms that describe the automatic training process of neural networks. Section 3.4.2 introduces the hyperparameters, which are critical parameters set before the start of training that control and guide the learning process. The third subsection, Section 3.4.3, describes the actual training procedure and the fine-tuning of these hyperparameters. For clarity, the hyperparameters are classified into active and passive categories. Finally, Section 3.4.4 provides a detailed insight into how these hyperparameters are adjusted to optimize the network performance. It's worth noting that the applied methods and techniques are based partly on empirical observation and partly on well-recognized resources in the field (e.g. Goodfellow et al., 2016, Smith, 2017, Santurkar et al., 2018). The choices made may not be directly reproducible and can vary depending on the specific problem. Nevertheless, these techniques are rooted in established literature and guided by a thorough understanding of the problem.

3.4.1 Neural Network Learning Mechanisms

The automatic learning process of a neural network is depicted in Figure 3.8.

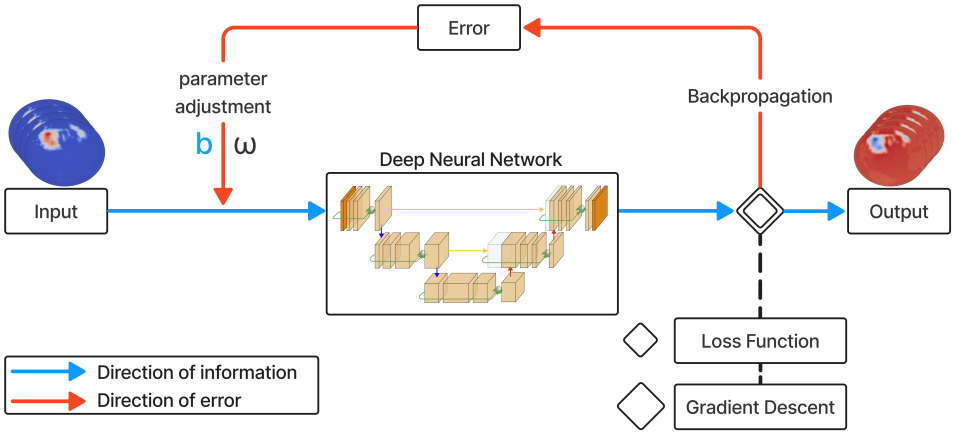


Figure 3.8: Weights ω and biases b are adjusted based on the prediction error. The backpropagation process calculates how much each weight and bias contributes to this error. The gradient descent method then uses these calculations to adjust the weights and biases, aiming to minimize the overall error. Schematic inspired by Preethi et al., 2021

The relation between backpropagation and gradient descent, the two core components of training, are shown in Figure 3.8. Backpropagation, introduced by Rumelhart et al., 1986, is an algorithm used to efficiently calculate the gradients of the loss function with respect to the network's weights (ω) and biases (b). The process involves two stages: a forward pass (blue arrow) where the input is propagated through the network to compute the output, and a backward pass (red arrow) where the prediction error (the difference between the network prediction and the targets) is propagated backward through the network. The backward pass involves applying the chain rule to compute the gradients, and these gradients represent how much a small change in the weights and biases would affect the loss. Note that the backward pass propagates through the entire network, including skip and residual connections. On the other hand, gradient descent is an optimization algorithm used to adjust the weights and biases of the network in a direction that reduces the loss. It uses the gradients computed by the backpropagation algorithm to guide these adjustments.

Before diving into the specifics of hyperparameters and optimization, it is important to understand the concepts of generalization, overfitting, and underfitting. Generalization refers to the model's ability to adapt properly to new, previously unseen data. Overfitting occurs when a model learns the detail and noise in the training data to the extent that it performs poorly on new data. Its a symptom of an excessively complex model. Underfitting occurs when a model is too simple, learning too little from the training data, and produces a poor fit to the data.

3.4.2 Hyperparameters

Hyperparameters are the adjustable settings of the network, distinct from model parameters such as weights and biases, which are automatically updated through forward and backward propagation. Passive hyperparameters are usually set based on the specifics of the architecture and problem, such as data structure and task. Passive hyperparameters typically involve a discrete choice, like the selection of a loss function or activation function. On the other hand, active hyperparameters are the adjustable parts of the network and often fall into a continuous range or come from a varied set of choices. Active hyperparameters include the width and depth of the architecture, the number of epochs, and the settings and types of regularization techniques used.

Active hyperparameters

The **learning rate** is often regarded the most critical hyperparameter in the network. It determines the size of the steps the optimization algorithm takes in the gradient descent process. In essence, it controls how much the model learns from the data during each iteration of training. Second after the learning rate is the **batch size**. Given the magnitude of the generated training data, it is not practical to process the entire training set at once. Therefore, the data is divided into subsets or "batches". This division allows for more frequent model parameter updates and reduces the system's memory load. Now the remaining hyperparameters will be discussed starting with epochs. The number of **epochs**, not to be confused with 'time' epochs, is closely related to both the learning rate and batch size. Each epoch represents one full pass of the training data through the network. More epochs mean more model updates. However, too many epochs lead to overfitting. Conversely, too few epochs can lead to underfitting, where the model fails to learn important patterns in the data. **Batch normalization** is a technique that helps stabilize the learning process while accelerating the training of the neural network. The technique normalizes the output from a previous activation layer through standardization. This normalization process helps address the problem of another technical term 'internal covariate shift', where the distribution of each layer's inputs changes during training, making the learning algorithm forever chase a moving target (Santurkar et al., 2018). Batch normalization is active because it contains a parameter called 'momentum' that determines the weight of the running average relative to the current batch's statistics. A higher momentum value will result in a slower update of the running estimate, while a lower momentum value will result in a faster update. Batch normalization impacts network training in a fundamental way: it makes the landscape of the corresponding optimization problem be significantly more smooth. This ensures, in particular, that the gradients are more predictive and thus allow for use of larger range of learning rates which yields the faster network convergence (Santurkar et al., 2018). **Dropout** is a regularization technique in neural networks. It randomly turns off some weights in a layer during a training pass, effectively reducing overfitting. **Weight decay** is another regularization technique used to prevent overfitting in neural networks by penalizing large weights. It adds a term to the loss function proportional to the square of the weights, which encourages the network to keep the weights small. This leads to a simpler model that generalizes better and discourages overfitting.

Passive hyperparameters

Before training optimization begins, certain elements of the model need to be set, including the loss function, activation functions, and optimizer. The **loss function** acts as a guide in the learning process. To give an analogy, it's like a compass, pointing the network in the right direction towards the target. The chosen loss function is Mean Squared Error (MSE). MSE is defined as:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{3.2}$$

where n is the total number of data points, y_i is the target value, and $\hat{y}_i$ is the predicted value. Squaring the differences amplifies the impact of large errors on the overall error measure, making MSE more sensitive to such errors than MAE (Equation 3.1). MSE is also the most common loss function for tasks involving image-to-image regression, which is what this network is designed for. The data contains some areas with high values, specifically where significant changes like deglaciation and RSL changes occur. On the other hand, the ocean areas present lower values with smoother, plateau-like changes. MSE accentuates these local large variations due to its sensitivity to outliers, making it an appropriate choice. Moreover, MSE has been observed to perform better than MAE in training. The chosen activation function is the Rectified Linear Unit (ReLU) introduced by Nair et al., 2010. **Activation functions**

introduce non-linearity into the network process. They determine the output of a neuron by deciding whether the neuron should be activated or not. Both the geodesic U-Net and the DeepSphere package use ReLU due to its advantages such as reducing the likelihood of 'vanishing gradients' and its computational speed. Vanishing gradients arise when gradients, back-propagated to preceding layers during training, become so small that the weights of these layers no longer contribute. An alteration to ReLU, the LeakyReLU, could also have been a potential choice. However, after examining the network, it was determined that there were no dead neurons. Therefore, the implementation of LeakyReLU was deemed unnecessary. Lastly, the Adam optimizer (derived from 'adaptive moment estimation') from Kingma et al., 2014 is used. It is also implemented in both the geodesic U-Net and the DeepSphere package and it is a popular optimizer overall. Adam self-adjusts its learning rate for different parameters making it highly effective. The structure of the architecture itself, the layer depth and number of convolutions are also considered as passive parameters, even though they are discrete options. This distinction is more clear when assessing how these hyperparameters are chosen as discussed in the third stage of Section 3.4.3.

3.4.3 Training optimization process

The training process involves updating the weights of the filters in the network to minimize the difference between the network's predicted output and the actual output. The back-propagation algorithm propagates the error gradient through the entire network, including the convolutional layers, to update the weights of the filters. For the optimization process a dataset containing 1,000 samples of both Method One (Section 2.2) and Method Two (Section 2.3) data is divided into three subsets: training, validation, and testing, with an 80-10-10 ratio. This is a common practice in machine learning as it provides a robust framework for evaluating the model's performance. The training set is used to fit the model, the validation set is used to tune the hyperparameters and to provide an unbiased evaluation of the model fit, while the test set is used to assess the final prediction performance on data it has not seen before. The three datasets also have associated prediction errors; training loss, validation loss and testing loss. Randomly shuffling the training set ensures that the model is not influenced by the order of the samples, which can lead to 'fake' better generalization. However, shuffling is not necessary for the validation and testing sets as they are used to evaluate the model's performance, not to update the weights.

The optimization of a neural network involves a multitude of tools, one very powerful tool is the hyperparameter search. Essentially all the possible combinations of hyperparameters are provided and an algorithm indicates the performance. However, with an increasing number of hyperparameters, the visualization and interpretation of higher-dimensional results become more challenging. Moreover, the same setting can provide different outcomes, and conducting a hyperparameter search across all parameters and ranges is computationally expensive. Therefore, the optimization process is divided into three phases. The first phase is mainly attributed to problem specifics and the main architecture. The second phase involves adjusting the network complexity, which is defined by layer depth and the composition of the U-Net blocks as defined in Section 3.3. A common quantification of complexity is the number of (trainable) model parameters. The third phase involves optimizing the regularization, which involve methods to prevent overfitting and improve model generalization. The second and third phases can be iterated multiple times because they balance each other out.

Optimization: phase one

The first phase includes the two most significant hyperparameters: learning rate and batch size. The process begins with defining a simplified version of the proposed U-Net architecture, typically with a two-layer structure. Low complexity also ensures cheaper computation, which is desirable for optimization. The learning rate is determined using a systematic approach, of gradually increasing the learning rate as depicted in Figure 3.9.

If all three ranges are observed in the approach the experiment is successful. Using this gradual decay method an initial learning rate of 0.01 was determined. Next is the batch size. According to Brownlee, 2019, batch size is a slider on the learning process. Small values give a learning process that converges quickly at the cost of noise in the training process. Large values give a learning process that converges slowly with accurate estimates of the error gradient (Brownlee, 2019). It is therefore desired to have a larger batch size, because then more information is learned with each model update. Also, a low batch size has high variability or oscillation, and unstable convergence. The downside is that larger batch

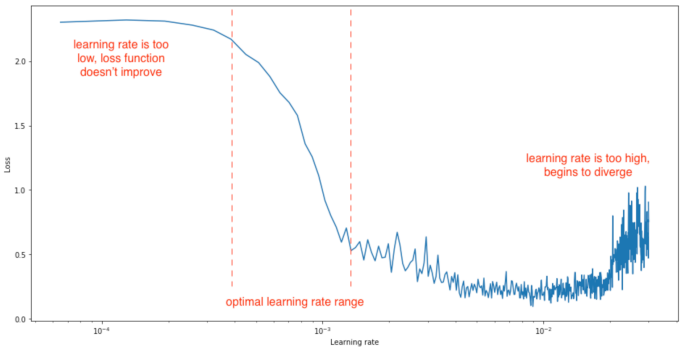


Figure 3.9: Depiction of validation loss against learning rate. The optimal learning rate is determined by progressively decreasing the learning rate and selecting the rate corresponding to the steepest decline in validation loss, as suggested by Jordan, 2023

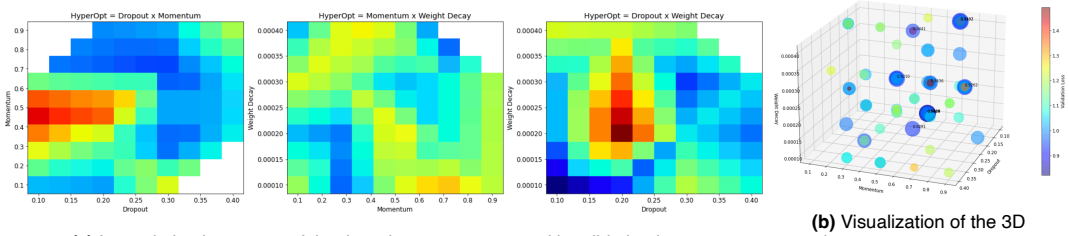
sizes require more memory, and if the batch size is too large the model requires too many epochs to update. In this study, the optimal batch size was determined to be 16. It's important to note that the training was carried out on a computer equipped with a Central Processing Unit (CPU) rather than a Graphics Processing Unit (GPU). CPUs generally have larger memory capacities, thus enabling larger batch sizes

Optimization: phase two

The second phase involves increasing model complexity. Increasing the complexity allows for the detection of higher-level features but can also lead to overfitting due to the almost exponential increase in model parameters associated with increasing the layer depth. The objective is to find a balance. Visualizing the predictions helps in determining the detection of higher-level features. The predictions should also become smoother or less noisy. In addition to layer depth, the composition of the blocks can also be altered. This is how the original U-net architectures have been expanded with batch normalization, point-wise convolutions, and residual layers.

Optimization: phase three

In the third phase, the focus shifts to fine-tuning regularization and momentum. Although a more complex model increases the risk of overfitting, it simultaneously enhances the model's learning capacity. To counteract overfitting while preserving learning performance, regularization techniques are applied. The hyperparameters associated with regularization and momentum are optimized via a search method. This approach is beneficial when dealing with three parameters (here dropout, weight decay and momentum), as it enables the visualization of the hyperparameter space in three dimensions, thereby facilitating the understanding of loss dynamics. The relationships between these hyperparameters are further analyzed through a set of three heatmaps, which can reveal any underlying trends.



(a) Interrelation heatmaps of the three hyperparameters with validation loss.

(b) Visualization of the 3D hyperparameter space.

Figure 3.10: Example representations of a 100-evaluation hyperparameter search for three epochs of a three-layer U-net using geodesic convolution. Color-coding denotes the normalized validation losses: blue signifies low loss and red indicates high loss. In Figure 3.10b, sphere sizes also represent validation loss, with larger spheres corresponding to lower loss, enhancing visibility. The pair of figures work together to help interpret combinational trends in the hyperparameters

Following the heatmap and hyperparameter space evaluation the created network is trained for multiple epochs until convergence is reached. Once converged, the prediction performance measures are

computed and the prediction is visualized using temporally-averaged MAE for the global grid and the MAE per timestep. If the network is overfitting the model architecture is too complex or regularization is set too low, and if the model is not converging to a reasonable prediction the model architecture is too simple. When phase three is completed the number of samples is increased, because it increases the probability of the network learning the GIA forward model, because it has more to learn from. During this process, the batch size and initial learning rate remain unchanged because the Adam optimizer inherently adjusts these parameters (Brownlee, 2019). The only parameter requiring manual adjustment is the learning rate decay.

3.4.4 Performance evaluation

Evaluating the performance of the trained model is an essential part of the machine learning process. Evaluation is performed using the test set that is separate from the network training data. Several metrics are used to measure the model performance, based on the fact that the RSL output and predictions are essentially images. MSE and MAE are calculated over the entire test set. The optimal value is zero for both, meaning there is no difference. Visualization of the prediction error is conducted using temporally-averaged spatial maps of MAE since it is more interpretable than MSE. In addition to these, the spatially-averaged MAE for each time step is evaluated to assess how well the network is learning the temporal relation. The values of these metrics are expected to increase over time, reflecting the increasing values for RSL back in time. Three additional metrics specifically related to image regression are also used:

- **R-Squared Coefficient (R2):** Statistic indicating the proportion of the variance in the dependent variable that is predictable from the independent variable(s). Measure of regression residual. An R2 of 1 indicates that the predictions perfectly fit the data
- **Peak Signal-to-Noise Ratio (PSNR):** Ratio between the maximum possible power of a signal and the power of corrupting noise. A high PSNR value would mean that the predicted image is closer to the actual image.
- **Structural Similarity Index Measure (SSIM):** Method for predicting the quality of digital television and cinematic pictures. SSIM is designed to improve on traditional methods like PSNR and MSE that have proven to be inconsistent with human eye perception. The SSIM index is a full reference metric; the measurement of image quality is based on an initial uncompressed or distortion-free image as reference. An SSIM of 1 equals the highest quality.

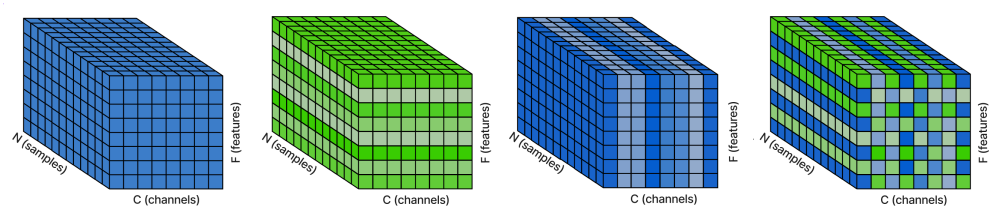
Using these metrics provides a comprehensive assessment of the model's performance across different aspects of the prediction task.

Apart from performance metrics the learning can also be examined by plotting the feature maps of the network; the transparent orange maps show in Figure 3.6. For example, when applying CNN on images of human faces, the low channel feature maps will likely show edges and lines. Deeper in the network the feature maps can show higher level features, such as eyes or ears. Similarly, plotting the feature maps in the U-Net can provide insight into what the network actually is learning. Sadly, as is often the case for custom neural networks, the feature maps of the U-Net network were not visually interpretable.

3.5 Normalization

Data normalization is an important step in ML preprocessing that is often overlooked. Normalization is important in neural networks because it helps to balance the scale of inputs and speeds up the training process. The ice load history input and RSL output in this study are standardized, a common form of normalization, using their respective mean and standard deviation values. Without normalization, a neural network trained on raw, unprocessed data may learn unusual correlations due to the high scale of the original data. Additionally, normalization helps to reduce the risk of vanishing or exploding gradients. However, for GIA data, these risks are already relatively low, due to a less significant scale of ice loading magnitude variation across samples compared to the ice loading magnitude scale between time steps. The synthetic datasets produced in this study have been mapped from an equiangular downsampled grid of 3.75 x 3.75 degree, to an icosahedral mesh of 10,242 vertices. The data can be conceptualized as a 3D matrix with shape (N, F, C), where N represents the number of samples, which

are the synthetic ice load history and RSL, F corresponds to the features presented on 10,242 vertices (i.e., ice loading values), and C stands for the input channels (timesteps), equal to 21. The decision to apply normalization across all axes or a specific subset of axes is significant, as it substantially impacts the training process and prediction accuracy. Normalization is always applied along the sample axis, as the network must learn to distinguish between the various generated datasets. Without this distinction, it would be impossible to build the emulator, as there would be no denormalization values for the desired output. This leaves four possibilities for normalizing, as represented visually by the denormalization application in Figure 3.11.



(a) Single Normalization, axis = NFC **(b)** Feature Normalization, axis = NC **(c)** Channel Normalization, axis = NF **(d)** Sample Normalization, axis =N

Figure 3.11: Visualization of normalization options, colors represent how denormalization is applied, because it is more intuitive. The provided axes indicate how normalization is applied. Each color shade represents a denormalization mean and standard deviation value. In the N direction the colors are always the same

The most common method for normalization is the 'Single' normalization (Equation 3.3); performed along the NFC-axis.

$$\mathbf{X}_{single} = \frac{\mathbf{X} - \mu_{N,F,C}}{\sigma_{N,F,C}} \tag{3.3}$$

In the Single normalization method, all the data is scaled based on the same single mean and standard deviation value, which retains the original relative scales of data values across all three dimensions of the dataset. Denormalization is conducted using only one value, as indicated by one shade of one color in Figure 3.11. Consequently, when a single time step of one sample is plotted, differences are discernible only by the scale. The primary advantage of Single normalization is that every input feature (i.e., every ice loading value across all axes) has the same scale. This homogeneity can lead to a smoother optimization landscape, which makes it easier for the optimization algorithm (gradient descent) to find a good solution. However, this approach sometimes leads to a relatively noisy and non-smooth prediction over the features due to the spatial value scales. For both ice loading and RSL there is a gradual low value scale across oceans and more abrupt changes and high value scale on ice sheet locations. These variances are maintained in Single normalization and, as a result, could create more noise in the predictions because the model is trying to account for a wide range of scale differences. This can manifest as a less smooth, or more erratic prediction over the spatial features. Deep architectures like U-Net tend to average these components, thereby reducing the effect of extreme values and smoothing out the prediction. However, despite this inherent averaging, the presence of extreme or highly variable values can still affect the learning process and the quality of the predictions. In the case of Feature Normalization (Equation 3.4), a denormalization map containing a mean and standard deviation for every pixel on the grid is applied to the prediction, smoothing out the spatial map for learning.

$$\mathbf{X}_{feature} = \frac{\mathbf{X} - \mu_{N,C}}{\sigma_{N,C}} \tag{3.4}$$

This approach subtracts local features, such as coastal trends and constant ice loading, that persist across all datasets and timesteps. Channel normalization ensures all time steps are on the same scale, with temporal differences only reappearing after denormalization.

$$\mathbf{X}_{channel} = \frac{\mathbf{X} - \mu_{N,F}}{\sigma_{N,F}} \tag{3.5}$$

This ensures that each timestep is treated as equally important during training. Lastly, Sample normalization subtracts a feature map for each time step, thereby allowing the network to learn only the most high-level pattern changes, visually resembling a slowly varying continuous manifold.

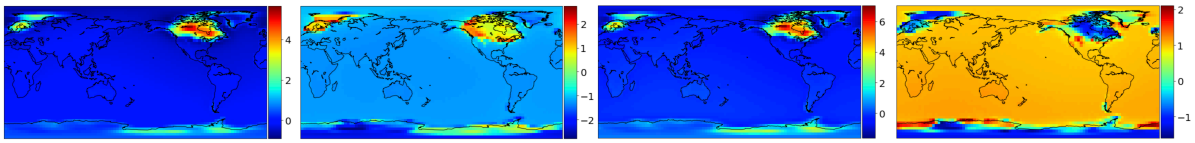
$$\mathbf{X}_{sample} = \frac{\mathbf{X} - \mu_N}{\sigma_N} \tag{3.6}$$

Though that manifold is more constant over time and spatially continuous, it is quite complex. To determine the best normalization method, all four options are compared using the two convolutional methods. This comparison also provides insight into the applicability of the convolutional methods. When Feature, Channel, or Sample normalization is used, a custom MSE function on denormalized loss must be created for the network to interpret the learning progress.

Metrics	Geodesic Convolution				Spectral Convolution			
Normalization	Single	Feature	Channel	Sample	Single	Feature	Channel	Sample
Performance								

Table 3.4: Performance comparison of the geodesic and convolutional method on normalization, where green indicates the best performance and red the worst performance. Datasets and hyperparameters are consistent for both convolutional methods. The models were trained for 15 epochs using a three layer architecture. All five metrics discussed in Section 3.4.4 were evaluated and turned out to be consistent across normalization methods, therefore a relative comparison could be conducted using color scales

The choice of normalization technique has a substantial effect on the efficiency of both the spectral and geodesic convolution methods. It defines what exactly the network learns and how the prediction error is distributed. To help with understanding the consequences of the normalization and how the data is altered with respect to the learning of the network, the normalization methods are visualized in Figure 3.12.



(a) Single normalization **(b)** Feature normalization **(c)** Channel normalization **(d)** Sample normalization

Figure 3.12: Examples of RSL training data for the four normalization methods. The Single and Channel normalization methods are spatially similar but have different temporal scales because in the Channel normalization the temporal scales are added back in the normalization. The sample normalization plot shows the temporal distribution of RSL across the globe, the curved relation is due to the spherical nature of the RSL

Sample normalization proved optimal for the spectral method, while Feature normalization is better suited for the geodesic convolution. This discrepancy can be attributed to the unique characteristics of their respective filters and their application. The geodesic method uses the DiNe-filter, initially designed to better capture irregular spatial features, such as brain parcellation. It operates as a one-ring direct neighborhood filter, adept at identifying sharper variations by treating all neighboring connections as equidistant. In contrast, the spectral method employs a Chebyshev polynomial approximation that operates globally. This 'filter' considers a neighborhood of multiple connection points on a spherical grid for each neighbor, improving the detection of patterns in Sample normalized data. This improvement is due to the more uniform scale distribution of data across features and time relative to the other normalizations, as illustrated in Figure 3.12. Specifically, the rise in sea level from LGM to the present, particularly across the oceans, represents a global pattern. However, Sample normalization averages this global rising pattern over time, creating a more complex sea-level manifold with longer range connectivity. In contrast, this complexity potentially explains why the geodesic method performed better with Feature normalization. Considering that RSL data and ice load history both contain local irregularities and global patterns (see Figure B.1), a comprehensive evaluation of both the geodesic and spectral methods is critical. In conclusion, the GIA emulator will be trained on synthetic data represented on an icosahedron mesh using a CNN with a U-Net architecture, using both the geodesic and spectral filters.

Chapter 4

Experiments and Results

In this section the two experiments designed to answer the final research sub-question on how to compare the created GIA emulator with the existing GIA model or numerical SLE method is presented. The first experiment denotes the main ability of the GIA forward model: creating RSL from a given ice load history using a constant Earth model and is discussed in Section 4.1. The second experiment, discussed in Section 4.2 presents an output alteration, where the RSL output is replaced by uplift rates, changing the multi-channel output to one-channel. The experiments are designed to evaluate the neural network on accuracy, efficiency, and versatility. Finally, the results of the experiments are presented and discussed in Section 4.3.

4.1 Experiment 1: Ice Loading to RSL

The first experiment is the GIA forward model with constant Earth model parameters. This experiment uses two methods to emulate RSL from ice load history: the spectral network and the geodesic network. This section discusses and compares both methods. The geodesic network implementation involves an architecture that resembles the structure detailed in Figure 3.6[1]. This structure comprises a five-layer U-net with a uniform channel depth within each block after the first convolution. Feature normalization is applied in the preprocessing for training the geodesic model. The spectral filter implementation follows a three-layer U-Net architecture as graphically presented in Figure 4.1. Sample normalization is applied in the preprocessing when training the spectral network. The most significant differences between these architectures constitute three aspects: layer depth, channel width, and the number of convolutions in the blocks.

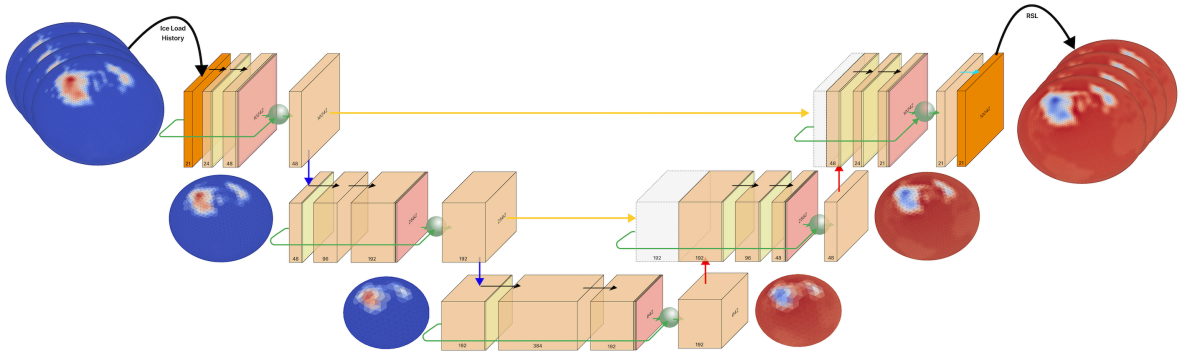


Figure 4.1: Architecture used in experiment one for the spectral network based on the DeepSphere spherical CNN U-Net. Each U-Net block contains five convolution operations

These differences are a combination of the methodological choices associated with their respective filters and design. The geodesic model is designed with more depth, achieving the same channel width in the deepest layer as the spectral network. The combination of channel and depth is a result of the connectivity between local neighborhoods, where the geodesic filter is applied locally and the spectral filter globally. The local neighborhood of the geodesic filter is a property, while the local neighborhood of the spectral filter varies based on the order of the Chebyshev polynomial, which in this experiment is set to a filter size of three. This configuration implies a connectivity range three times that of the one ring DiNe-filter. This distinctive application allows the difference in channel width and depth between the two

[1]The figure displays five convolutions per U-Net block; however, the geodesic architecture used in experiment one contains two additional convolutions per block - one standard convolution and one point-wise convolution

networks. The spectral filter's design enables more variations in channel width with fewer convolutions due to its global application. As it operates on the whole graph, it affords fewer weights for the same range of channel widths to learn spatial features. The global operation simultaneously maintains a lower convolution count due to its reach and influence over the entire data set. In contrast, the geodesic model includes a total of 63 convolution operations (nine blocks of seven convolutions), while the spectral network comprises a total of 25 convolutions (five blocks of five convolutions). Another key distinction between the two methods is the computational expense. The spectral network's training is considerably slower due to the higher computational cost required for the application of the Chebyshev filter. As noted, the spectral filter is applied to the entire graph, making the computations for eigendecomposition and matrix inversion of the pre-calculated Laplacian matrices particularly complex. The Laplacian can be computed beforehand as it solely describes the graph structure. Conversely, the geodesic network requires smaller computations as the one-ring DiNe filter convolutions are comparatively simple. This efficiency is facilitated by pre-determining the indices to the closest neighboring vertices for each vertex, possible due to the known icosahedron mesh structure, similar to the graph Laplacians. Training time for one epoch on average is $\sim$ 4 minutes for the geodesic networks and $\sim$ 20 minutes for the spectral network. The measurements lack precision as it depends on the number of available cores and memory. Multiple users access the used computer, thereby varying the processing power between trainings and evaluations.

4.2 Experiment 2: Ice Loading to Uplift Rates

The second experiment aims to predict global GIA uplift rates based on a given ice load history. To achieve this a new neural network is created from scratch and specifically designed, trained, and optimized to predict uplift rates. A crucial element of GIA modeling is the ongoing glacial rebound, which is observable in the present day. The uplift rate refers to the elevation of the earth's surface due to the substantial weight of ice sheets from the last glacial cycle. This experiment tests the spherical CNN U-Net's capability to predict this key aspect of GIA, thereby demonstrating its versatility. Both the geodesic and spectral networks are implemented and compared, as in the first experiment. This experiment uses Feature normalization, mainly due to the lack of output channels and the necessity for a consistent normalization approach between the input and output. Given that this problem involves the transformation from a multi-channel ice load history to a single-channel uplift rate, it becomes necessary to adjust the U-net architecture. Given the U-net's symmetry requirement between its contracting and expanding paths, an extension of the architecture is required to accommodate the channel change. Two potential methods for extending the model are identified: a direct approach and a gradual approach. The direct approach is comparatively simpler and more efficient. It involves modifying the final convolution in the U-Net to convolve directly to a single channel. This method would require fewer parameters, thus making it less computationally demanding and faster to train. Furthermore, it could potentially reduce the risk of overfitting, since the model would have fewer parameters to fit to the data. The gradual approach introduces intermediate convolutions that progressively reduce channel width. Consequently, the model has the opportunity to learn more abstract features at each reduction stage, potentially increasing accuracy. The choice to implement a gradual approach has been empirically derived based on a lower validation loss after optimization. A hyperparameter optimization was conducted on the intermediate channel depths using two and three convolutions to ascertain the most effective extended path. Point-wise convolutions are also added after each intermediate convolution in the extended path to help the channel reduction progress more smoothly.

Figure 4.2 illustrates the extensions added to the networks used in experiment one. The enhanced performance of the gradual approach is attributed to the skip connections, which facilitate the transfer of temporal learning from the multi-channel layers to a single-channel output. The complexity of the networks also have to be reduced, because the learning of redundant weights can disturb the training, caused by the output channel reduction. The geodesic model has been decreased in depth, because there is less hierarchical features to be detected compared to the RSL data. Since the spectral network already has lower depth a similar reduction is achieved by reducing the filter size.

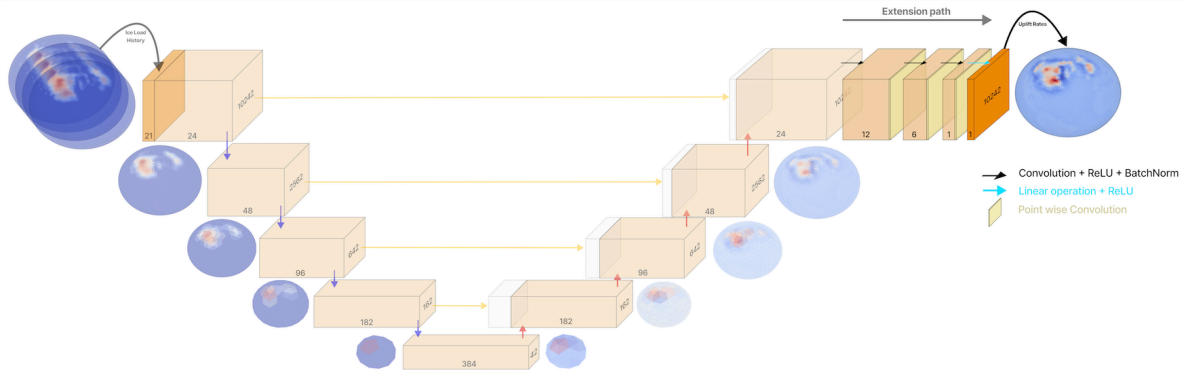


Figure 4.2: Simplified schematic of the U-net architecture with the geodesic DiNe-filter for experiment two. The contracting and expanding blocks have been simplified as they are the same the blocks used for each convolutional method in experiment one

4.3 Results

In this section the results and outcome of the experiments are discussed. The section provides an evaluation of the experiments using measurement metrics and visualizations using the order in which the experiments were presented. For both experiments first the details of the training are provided followed by a table comparing the two implemented convolution methods. Both networks in both experiments are trained on dataset of 2,000 samples comprised of a equal divide between Method One and Method Two synthetic generated datasets. This results in a total of 344,131,200 parameters in the training set. The original grid of the (downsampled) global map is (48, 96), but has been mapped to an icosahedral mesh with 10,242 vertices. The validation and testing set are each comprised of 250 samples.

4.3.1 Experiment 1

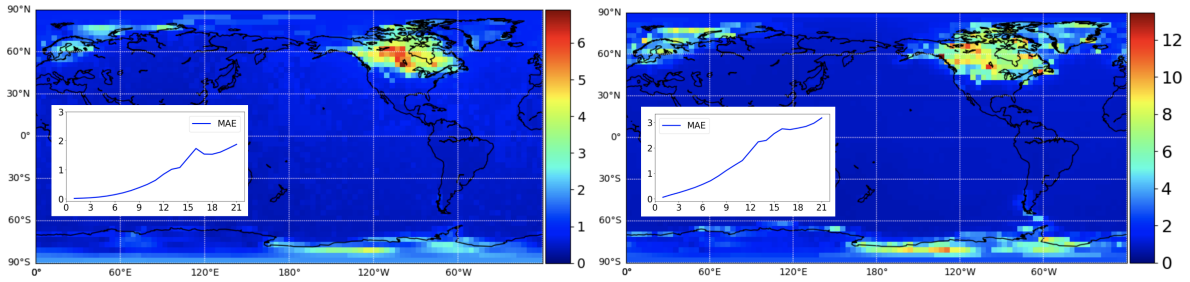
In Table 4.1 the performance of the geodesic and spectral network on experiment one based on the metrics explained in Section 3.4.4 are presented. Both networks are trained for 60 $\pm$10 epochs and use cosine annealing learning rate decay with an initial learning rate of 0.01 and minimum learning rate of 1e-6. The choice for cosine annealing as a learning rate scheduler is explained graphically in Figure B.7. The dropout regularization is equal to 0.2, which is not significant but enough to improve generalization without causing underfitting. A batch normalization momentum of 0.275 was implemented, yielding a relatively high running average resulting in a significantly faster convergence as explained in Section 3.4.2. The weight decay in the Adam optimizer was set to 0.0001 following the visualization of a hyperparameter optimization on both the geodesic and spectral networks.

Metrics	Architecture	Model Complexity	RMSE	MAE	R2	PSNR	SSIM	Avg. Prediction Time
Geodesic	Five-layer	7,625,253 parameters	2.84	1.13	0.9970	50.9	0.9906	0.0468 seconds
Spectral	Three-layer	1,093,692 parameters	3.19	1.49	0.9954	52.2	0.9961	0.151 seconds

Table 4.1: Performance comparison and complexity description of the trained geodesic and spectral networks, where RMSE and MAE are provided in [m]

The emulation speed is most effectively compared to the numerical SLE method, which averages 12.3 seconds to predict RSL for a single ice load history. It is important to note that the neural network training is performed on the Archimedes computer, while numerical SLE computations are carried out on the Hipparchos computer. The Hipparchos has a 64-core CPU with AMD architecture, while the Archimedes has a 72-core CPU with Intel architecture. Despite having similar clock speeds, the Archimedes possesses a much larger cache, which affects the speed of accessing frequently used data.

From Table 4.1 it is inferred that the two networks have very similar performance. The first three metrics indicate an increased performance of the geodesic network, though the image related final two metrics indicate an increase performance by the spectral network. The higher RSME compared to MAE indicate both predictions are subject to outliers. To further investigate the differences the two network predictions are visualized in Figure 4.3.



(a) Geodesic network spatial MAE [m] and temporal MAE [m]

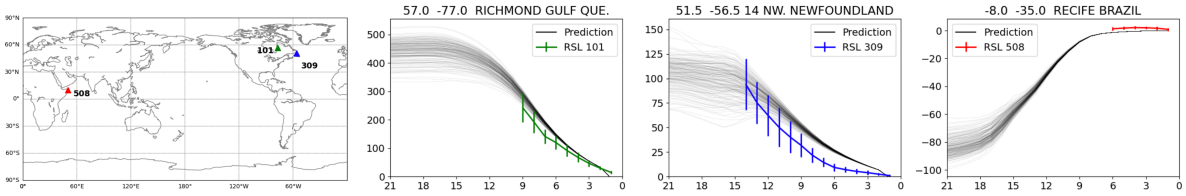
(b) Spectral network spatial MAE [m] and temporal MAE [m]

Figure 4.3: Results of experiment one network predictions providing insight in spatial and temporal learning. The main figures show the temporally-averaged MAE [m] for the test prediction of 200 samples for the two networks. The plots inside the figures provide the spatially-averaged MAE [m] per timestep of 1,000 years

Figure 4.3's plots illustrate that the temporal prediction performance and trend capture across timesteps is fairly consistent between the two networks, although the geodesic network demonstrates a slightly lower error. Given the differences in the applied normalization methods, this parity seems more coincidental than expected. Interestingly, the temporal scale added back between timesteps for the spectral prediction seems to result in similar temporal prediction performance as the geodesic network, despite the latter learning this in training. This normalization aspect is explained in Section 3.5. However, when assessing spatial performance, a significant difference is observed between the two networks. The disparity can be attributed to two primary factors: (1) the synergy between the normalization method and the applied filters, and (2) structural differences inherent to the functioning of these filters within each network architecture. Concerning the first component, the combination of Sample normalization and the globally applied spectral filter in the spectral method yields a uniform distribution on the global pattern. This can be remarked from the more constant values observed over the oceans (continuous shades of blue) when compared to the geodesic prediction. Alongside this, the magnitude of prediction error aligns with the inherent variation of the RSL data on the locations of former ice sheets (as discussed in Section 3.5). This observation of synergy is similarly observed for the geodesic network with Feature normalization, the scale difference between RSL in North America and on Greenland between Figure 4.3 and Section 3.5 are strong indicators. This observed phenomenon is further substantiated by the higher PSNR and SSIM values for the spectral network. These values are a combined effect of the Sample normalization and the globally applied spectral filter. PSNR and SSIM determine the degree to which to images are visually similar, here representing the relative color error between prediction and target. In the training phase, this combination allows the spectral network to learn a continuous manifold. However, as differences are normalized across features and timesteps, the prediction denormalization step results in an amplification of these learned differences. On the other hand, the geodesic network is preprocessed using Feature normalization and applies its respective filter locally. The DiNe-filter has a smaller connectivity compared to the spectral filter, one-ring opposed to multiple ring consideration, ensuring the geodesic network learns the spatial relations for each timestep more specifically.

The results from the Figure 4.3a's main figure have been modified by subtracting the precomputed mapping error per sample of the testing set associated to the Feature normalization. Originally the most considerable prediction errors of the geodesic prediction appeared on the coastlines near the poles. The Sample normalization did not need this subtraction. Instead the spectral network offers a unique benefit during denormalization: the feature map per timestep, calculated on the regular grid data, is added back to the prediction. This approach mitigates consistent errors at the poles, effectively concealing them at the south pole by averaging out the error through the added back mean and standard deviation per timestep.

In summary, the geodesic network is better at capturing the long-wave GIA signal and has a lower prediction error than the spectral method, However, the performance of the network has not been verified. Verification is conducted using a selection of specific RSL site location data from van der Wal et al., 2010. For brevity only the geodesic network is evaluated, shown in Figure 4.4.



(a) RSL site locations

(b) Prediction evaluations of geodesic model, From left to right: RSL location 101, 309 and 508

Figure 4.4: Geodesic network RSL predictions at selected RSL site locations with latitude and longitude coordinates per timestep of 1,000 years

The 200 test dataset predictions for the geodesic network, displayed in Figure 4.4b, display high precision but relatively lower accuracy in comparison to the RSL data. Several factors could contribute to this observation. Firstly, a difference in resolution exists: while the specified RSL site locations are based on a grid of 0.5 x 0.5 degrees, the prediction resolution is set on a larger grid of 3.75 x 3.75 degrees. This difference in resolution could contribute to the observed discrepancy in precision. Furthermore, the simplifications implemented on the SLE mathematical method, as detailed in Table 2.2, may also play a role in the noted differences. Additionally, potential inaccuracies in the Earth model could be another significant factor. Any changes in the Earth model can substantially alter the sea-level response to ice loading, thus influencing the alignment with the RSL data. For comparative purposes, prediction evaluations for the spectral method are available in Figure B.8. It's important to note that, while the predictions do not align perfectly with the RSL data, this does not necessarily indicate an issue with the performance of the network. The model is not explicitly tuned to fit this data, and a perfectly fitting model can compromise its generalization capability. Consequently, the absence of a perfect fit should not be interpreted as a lack of performance. Rather, the reasonably close alignment observed between the predictions and RSL data, given the complexity of the task and the degree of generalization required, supports the verification of the network's predictions.

4.3.2 Experiment 2

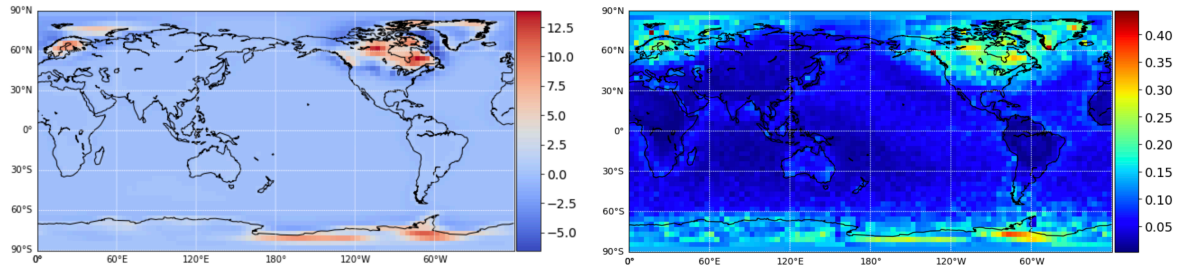
In Table 4.2 the performance of the two convolutional networks are presented. The networks are trained using a 2,000 sample ice load history to GIA uplift rate dataset. The validation and test set are both comprised of 150 samples. The main hyperparameter settings of both methods are unchanged compared to experiment one. However, there a some differences due to the decreased complexity of the output. First, the number of training epochs has been decreased to 20 epochs based on the speed of convergence and the decreased complexity of the two methods. The filter size of the spectral filter has been decreased to a second order Chebyshev polynomial. The decrease in model parameters explains the observed decrease in prediction time compare to experiment one.

Metrics	Architecture	Model Complexity	MSE	MAE	R2	PSNR	SSIM	Avg. Prediction Time
Geodesic	Four-layer	1,905,564 parameters	0.019	0.082	0.9939	49.1	0.9838	0.0254 seconds
Spectral	Three-layer	728,804 parameters	0.069	0.108	0.9772	43.4	0.9762	0.0773 seconds

Table 4.2: Performance comparison table of experiment two. In this table MSE is provided as opposed to RSME because the values of MSE are lower than 1. Values for MSE and MAE are in [mm/yr]

From the results in Table 4.2 is derived that the geodesic network significantly outperformed the spectral network. This difference was anticipated due to the normalization nature of the dataset. It was established in Section 3.5 that the geodesic network performs better on Feature normalized data. In addition, the uplift rate data is more irregular compared to the RSL, for example, in North America. Whereas the RSL in North America resembles a more uniform single dome, the GIA uplift rates often resemble three smaller domes. To evaluate the performance of geodesic network on the second experiment in more detail the MAE of the prediction over the regular grid is provided along with the mean of the prediction targets in Figure 4.5.

When examining the uplift rates prediction error from experiment two in relation to the geodesic network prediction in experiment one, it is found that the MAE across the oceans is significantly larger for the experiment two prediction. However, this is not surprising, given that the distribution of uplift rates globally presents a lower value scale compared to the synthetically generated RSL data, thereby



(a) Mean of GIA uplift rates from test dataset in [mm/yr]

(b) Geodesic network MAE [mm/yr]

Figure 4.5: MAE Prediction error of the geodesic network next to the mean GIA uplift rates of geodesic convolutional network in experiment two

distributing the learning focus more evenly. A comparison of the maximum prediction error and the maximum mean of target values, particularly in the case of North America, reveals similar outcomes for both experiments. This suggests that learning the relationship from ice loading to RSL and from ice loading to uplift rates is comparably feasible, at least to some extent.

To properly evaluate the performance of the geodesic network, the two types of errors need to be distinguished: (1) the prediction error and the model-data error (2). The prediction error of the CNN, which refers to the difference between the CNN prediction and the targets computed using the physics-based GIA-forward model. Second, the model-data error, which considers the difference between the computed targets of a physics-based GIA model (using realistic ice models and Earth histories) and actual measurements and observations. The prediction error is calculated on a separate test dataset not used in the training process, the prediction error is therefore equal to the testing error. The temporally-averaged MAE of the geodesic network prediction, which represents the prediction error, varies on the global map between 0.02 and 0.4 mm/yr. This suggests that the CNN is effectively capturing the behavior of the physics-based GIA-forward model within this range of error. On the other hand, the model-data discrepancy for current GIA models, as reported by Bagge et al., 2023, can be much higher. For regions like Antarctica, Fennoscandia, and North America, this discrepancy spans from 0.03 to 0.98 mm/yr. For areas with limited observational data such as Alaska and Greenland, the discrepancy extends to between 2.07 and 8.63 mm/yr. Comparing these values, it is inferred that while the CNN prediction error remains significantly lower than the typical model-data discrepancy encountered in GIA modeling. This comparison suggests that, even with the approximation error introduced by the CNN, the overall accuracy of the GIA model may not be significantly affected.

For further context, uncertainty quantification (UQ) for GIA uplift rates, as provided by Simon et al., 2020, can also be considered. They used Global Position Satellite (GPS) data to measure vertical land motion (VLM) across Fennoscandia and North America. Their findings suggested that within deglaciation centers, GIA model uncertainty can reach up to $\sim$2 mm/yr (VLM). The estimated GIA uncertainty for RSL change is $\sim$0.3–0.5 mm/yr along the U.S. East Coast and $\sim$0.6–0.8 mm/yr in the North Sea (Simon et al., 2020). These uncertainties present another important perspective on the potential error sources within GIA modeling. The prediction error of the geodesic network is significantly lower than the uncertainty range proposed by Simon et al., 2020, generally indicating robust performance. Nevertheless, it's important to distinguish between accuracy and uncertainty: while accuracy means the difference between target and prediction, uncertainty represents the expected range of targets. Even though the geodesic network from experiment two demonstrates superior accuracy to the estimated uncertainty and comparable accuracy to space geodetic data, it does not directly suggest that the emulator accurately captures all physical processes. The GIA uplift rates have not been directly measured to any observations, therefore the real accuracy depends on the implementation of the numerical SLE method on the generated ice load histories.

Chapter 5

Discussion and Limitations

This chapter presents the discussion on the sub-research questions on the dataset generation, algorithm design and experiments in Section 5.1. Afterwards, the associated broader limitations and statements regarding the works's contribution are detailed in Section 5.2.

5.1 Discussion

he first section provides a discussion on the research questions presented at the beginning of this study, following the chronological order in which the sub-questions were presented. First, the sub-question related to data generation is evaluated. This research aims to explore the capacity of a deep learning neural network for emulating the forward Glacial Isostatic Adjustment (GIA) model. To train the neural network, synthetic datasets were generated using two different methodologies. Similar techniques to the proposed data generation methods have been applied by Lin et al., 2023, although the precise details and contrasts between these approaches are not available, however a WPCA method was also applied. While synthetic datasets were generated through distinct ice load generation methods, these two synthetic datasets were not explicitly evaluated within this study.

The validation of the magnitude and distribution of the ice loading was visually accomplished through a comparison with the original ice models by plotting spatial maps of the differences. This method offers a more comprehensive overview than solely assessing the ice volume per timestep. During the development of the generation methods, normal distributions served as key modulators of the randomized ice load histories. Plotting these spatial difference maps provides insights in assessing the quality and structure of the randomization process. In the 'Select, Shift, and Combine' artificial ice load generation method, for instance, high linear combination weights can induce unrealistic variations in the ice load, even while presenting realistic ice volumes.

The decision to combine both dataset generation methods was adopted to: (1) ensure sufficient variation in the dataset for effective network learning, and (2) create diversity that could ameliorate the networks in identifying patterns, thus adding to their ability to generalize. However, it's important to acknowledge that the influence of such diversity on pattern recognition and subsequent generalization cannot be determined with absolute certainty from data inspection alone. Empirical testing, involving comparing prediction errors of Convolutional Neural Networks (CNNs) trained using both datasets separately, combined, and using the alternate datasets for testing, serves to validate this strategy. The effectiveness of this strategy can be established empirically by comparing the prediction errors of the CNNs. This involves training the networks using both datasets separately, in combination, and testing using the alternate dataset. Based on these considerations, it was discovered that applying the synthetic datasets derived from Method One and Method Two independently led to suboptimal outcomes. Empirical evidence supported this conclusion: when the geodesic and spectral networks were trained on one method's dataset and tested on the other, a deterioration in prediction accuracy was observed. However, training the networks on a combined dataset from both methods resulted in prediction performance on the validation set that was comparable to the performance when the networks were tested on the corresponding testing set from the same method. This observation suggests that combining both datasets enhanced the networks' generalization capabilities, thereby validating the suggested approach of using two distinct data generation methods.

This interpretation suggests a balance must be found between optimizing training and handling dataset variability for optimal generalization. While increased dataset variability could boost generalization, it can also make it more challenging to achieve convergence when training the network to capture this

variability. This issue has been addressed through one iteration of dataset variability reduction, notably from the Large variation ice load set to the Medium, despite its time-consuming nature.

The second sub-research question concerns the design of the network architecture. It is important to note that the architecture of a network can only be preliminarily evaluated based on theory and existing literature. To assert its efficiency, the architecture must be implemented and tested empirically. Although the training optimization process, described in Section 3.4.3, was effective for the particular problem examined in this thesis, it should not be viewed as a universal guideline for training deep neural networks. Numerous factors could influence the outcome of any of the described phases, and minor structural differences in the dataset or network architecture can have significant consequences, as demonstrated by the differing outcomes of the two experiments.
The two network architectures constructed for this thesis draw from the works of Zhao et al., 2019 and Defferrard et al., 2020. The former presents a geodesic spherical CNN with a U-Net architecture, which is a type of CNN configuration designed specifically for biomedical image segmentation that features an encoding (downsampling) path and a symmetrical decoding (upsampling) path, forming a ”U” shape. Meanwhile, the latter introduces a spectral graph CNN, that also uses its own U-Net architecture.

However, these original works were subject to substantial modification due to structural data differences and the designated tasks. Two significant alterations to the networks were identified. The first alteration originates from the lack of any inherent temporal learning associated to the U-Net architecture. Which poses an issue because the GIA problem contains multiple temporal relationships (i.e., deglaciation and subsequent sea-level rise). In response, the architectures were designed with a relatively high U-Net layer depth, residual connections, and multiple point-wise convolutions to simulate a form of temporal learning by enforcing learning across channels, which equal timesteps in the input. This achieved through combined effects of three structural components: (1) the convolutional operations, (2) the U-Net depth and (3) the skip connections. n a CNN, each filter, having unique weights for each input channel, operates across all channels to learn distinctive features. With U-Net’s depth, the network can detect high-level features equating to long-term patterns across sequential channels. For example, in a glaciation sequence, early filters capture low-level features of an ice dome formation, while deeper filters learn the more complex evolution of the dome over time. The U-Net architecture, through encoding and decoding paths, compresses information into abstract representations and reconstructs it while retaining learned features. Skip connections pass early, detailed feature maps to the network’s output, ensuring each input channel’s distinct features are preserved and contribute to the final output. Together, these components form an effective surrogate for learning long-term dependencies in the data.

Despite the exploration of adding a recurrent layer (e.g., an Long short-term memory (LSTM) layer) to the U-Net ’bottleneck’, which is the deepest layer in the U-Net, this modification substantially increased training time and provided no measurable increase in temporal learning. In addition, the performance evaluation of the geodesic network in experiment one confirmed that the tailored U-Net structure is capable of ’indirect’ temporal learning by connecting channel-wise information with hierarchical learning. This capability was confirmed by the consistency of the Mean Absolute Error (MAE) per timestep. The MAE of the emulator prediction of RSL from ice load history, demonstrated a steady range between 0.1 and 2.5 [m] across timesteps. This consistency suggests that the network is effectively learning temporal patterns. Therefore, while there wasn’t explicit implementation of temporal learning, the absence was counterbalanced by the successful application of surrogate methods.

The second major alteration to the networks was related to the speed of convergence. Implementing batch normalization, a technique used to standardize the inputs to a network, resulted in a significant reduction in training time and an increase in generalization. In conjunction with the self-adjusting Adam optimizer, batch normalization enabled a broader range of learning rates. This broader range is beneficial as it provides the model with greater flexibility during the learning process. The model can adjust the learning rate dynamically, adopting a faster rate when the gradient is stable and slowing down when the model nears a minimum to avoid excessive learning. This adaptability enhances the speed of convergence, making the learning process more efficient.

The final sub-research question assesses the GIA emulators performance against the traditional GIA model concerning precision, efficiency, and versatility. Furthermore, it examines the improvements brought about by the emulator to the field of GIA modeling. The versatility is demonstrated by constructing two experiments: one to predict RSL from ice load history, and the second to predict GIA uplift rates from ice load history. This demonstrates the emulators abilities to handle different types of predictions, similar to the numerical SLE method. The versatility is then achieved if the emulators can predict to a reasonable degree. This emulation capability is measured by accuracy and efficiency of the built emulators. The emulator accuracy is quantified through various metrics and visualizations and verified using observations and literature. The MAE is used as the primary metric for evaluating the accuracy of the emulator's predictions against the actual values. The geodesic network emulator used in experiment one achieved an overall MAE of 1.13 m and a MAE of 0.082 mm/yr in experiment two.

In the first experiment, the ice load to RSL emulator with constant Earth model, using the geodesic network, showed an adequate and consistent performance spatially and across time steps. The emulator was surprisingly adept at capturing the long-wavelength GIA signal, despite its lack of temporal learning and its error distribution demonstrated the synergy between the normalization method and the applied filters in the network. When verifying the network's performance using specific RSL site location data, the emulator's predictions showed high precision. Though the accuracy was not uniformly aligned with the measurement error range, which could be attributed to factors such as resolution differences, simplifications in the Sea-Level Equation (SLE) mathematical method, and potential inaccuracies in the Earth model. While the predictions didn't align perfectly with the RSL data, the geodesic network's performance is still satisfactory given the complexity of the task, variability of the data and the degree of generalization achieved. The ice load to RSL machine learning emulator of Lin et al., 2023 achieved a maximum temporally-averaged MAE of $\sim$ 3.5 m, whereas the maximum temporally-averaged MAE of the geodesic network in experiment one is close to 6 m. This result was retrieved from a verbal communication with Lin et al., 2023. It is also known that in Lin et al., 2023 a HEALPix grid is used with around 3,000 pixels as opposed to the 10,242 vertices on an icosahedral mesh in this thesis.

In the second experiment, the geodesic network demonstrated superior predictive accuracy. The MAE for the ice load to uplift rate emulator prediction varied globally between 0.02 and 0.4 mm/yr, effectively capturing the behavior of the physics-based GIA-forward model. Although model-data discrepancies in GIA models can be higher (Bagge et al., 2023), particularly in regions with limited observational data, the prediction error from the geodesic network remained significantly lower. Comparison with uncertainty quantification for GIA uplift rates also showed that the prediction error was substantially lower than estimated uncertainties of GIA uplift rates proposed by (Simon et al., 2020). However, it is important to note that the accuracy of the emulator depends on the implementation of the mathematical SLE method on the generated ice load histories, as there are no direct measurements of GIA uplift rates to validate against, because they are part of the vertical land motion.

The efficiency of the emulators is determined by the computational speed, an important limitation of the mathematical SLE model, particularly for global simulations. In stark contrast, the emulators, once trained, were capable of making predictions in a fraction of the time taken by traditional models. The computation time needed to emulate RSL for one ice load history averaged between 0.5 - 1.5% of the time required for the pseudo-spectral TUDSLE. In absolute time the emulators are able to predict global RSL of 21 timesteps mapped to 10,242 vertices of a spherical icosahedral mesh in 0.04 seconds and one global map of uplift rates in 0.02 seconds on a computer with a 72-core CPU with Intel architecture. This increase in efficiency could allow for more extensive exploration of different scenarios and parameters, contributing to a deeper understanding of the GIA process. The substantial improvement in efficiency extends even as model fidelity and accuracy increase, which typically result in longer computation times in traditional models. Because an emulator does have to be retrained this does not automatically mean an increase in emulator accuracy when trained on improved datasets. Nevertheless, the significant reduction in computation time points to the substantial efficiency improvements brought by the emulators to the field of GIA modeling.

The role of machine learning in addressing the complexities and uncertainties inherent in GIA modeling is important. In this thesis, the creation of a surrogate model, has provided a means to rapidly estimate GIA outputs based on given inputs. This effectively paves the way for replacing the need for computationally demanding physics model calculations, allowing for a more efficient exploration of a wide range of scenarios. This efficiency is particularly beneficial in the context of the inverse model, which requires numerous iterations of the forward model. Moreover, the surrogate model can help reduce the uncertainties linked with GIA modeling by enabling a more comprehensive assessment of the sensitivity of GIA predictions. The sub-research questions in this thesis were formulated to systematically address the main research question, specifically focusing on the design and application of an emulator. While the thesis successfully presented the design of an emulator, assessing its broader contribution can be challenging due to certain computational and data constraints. A key determinant of the emulator's usefulness is its accuracy. This has been reasonably achieved, particularly for the uplift rate emulator. The accuracy of the thesis emulator is less than the emulator created by Lin et al., 2023. To the authors knowledge, there is no benchmark for the ice load to uplift rate emulator. Beyond accuracy and efficiency, the resolution of the emulator's predictions play an important factor in its practical application, particularly in the field of GIA research. To become a viable plug-in tool for GIA research, improvements in the resolution of the emulator's predictions are necessary. The improvement in the resolution would ensure more precise and detailed predictions, thereby enhancing the emulator's applicability and contribution to GIA studies. Still, moving forward, the combination of machine learning techniques and traditional GIA modeling, as explored in this thesis, present promising opportunities to address the ongoing challenges in this field.

5.2 Limitations

This section evaluates the broader limitations of the study, specifically in relation to the sub-research questions and scientific applications. The specific limitations related to emulator design and datasets have been previously discussed in the results and discussion sections. The study presented the construction of two deep learning-based GIA forward models capable of predicting RSL and uplift rates from ice load history. These emulators, however, were built using only a single Earth model, which introduces a major constraint. To adapt the emulator to a different Earth model, the training process would have to be repeated and the RSL output recomputed. If the Earth model is similar the training optimization will not be required. While the limitations of using a singular Earth model apply to the traditional mathematical model it emulates, the latter has the flexibility to adapt to different Earth models over a constant ice loading history, a versatility that the current emulators do not possess. These broader limitations need to be addressed to ensure the emulators' wider applicability and effectiveness in GIA modeling. The implications of these limitations, along with potential ways to address them, are discussed in detail in Chapter 6.

There are several limitations associated with the quality of the randomly generated ice loading histories. Firstly there are limitations associated to the RSL output computation: the GIA physics in the training data set is limited due the assumptions applied SLE to increase computational efficiency such as, no rotational feedback, no lateral variations in Earth model parameters and ignoring changes in the geoid. Secondly, the testing set, which is a subset of the full synthetic dataset, is sufficiently large to provide an insight into general performance trends within this specific dataset. However, it doesn't provide any indication of how the emulators might perform with a completely different ice model. For instance, the emulators' performance in relation to rotation equivariance has not yet been assessed. A network can be trained to understand this concept by augmenting the dataset with Euler rotations. For the GIA signal, this would involve applying one rotation per sample, consistently over time. The spectral method is well-known for its capacity to learn rotated representations of the same data due to the nature of the filter, as noted by Defferrard et al., 2020 However, this is not true for the geodesic network because of its strong localization characteristic. If the emulator is trained with a rotated version of the synthetic dataset, it could potentially predict RSL based on an ice sheet located in Northern Africa, for example.

Furthermore, the emulators' ability to process finer mesh input data remains untested. Addressing this issue would involve increasing the number of vertices and reevaluating the model optimization process,

but such changes would inevitably result in higher computational costs. The versatility demonstrated by traditional GIA models, of allowing for various inputs and parameter adjustments, is only matched and not exceeded by the current emulators. The forward GIA-model can also predict the influence of RSL due to current deglaciation and allow different Earth models as input. However, achieving a similar versatility with a single model that could predict both RSL and uplift rates from a single ice load history presents a conceptual difference that is not particularly desirable.

Chapter 6

Future Work

This chapter outlines recommendations for future work, organized according to the potential time available: from an additional week, to a month, up to an entirely new thesis project. These recommendations are based on the conclusions drawn in the discussion and limitations sections.

If one more week were available, the primary focus should be on further optimizing the training. Ideally, the networks should be implemented on a computer with greater processing power or on a GPU. Transferring the models to a GPU device wouldn't be overly complex as the models were constructed using PyTorch. However, GPUs often have limited memory availability, which might necessitate using a smaller dataset or smaller batch sizes. If this is the case, more extensive optimization would be needed, as the model architecture might need to be revisited. During the training process, it was observed that the models did not achieve overfitting, suggesting that a longer training duration with more epochs could potentially yield improved performance. Since training can continue to improve performance until overfitting; when the validation loss starts deviating from the training loss. An alternate use of the additional week could be more extensive quantification of the model's accuracy, along with further verification of the dataset generation process. To actively compare the precision of the emulators and compare them against network error between predictions and targets, additional RSL and uplift databases would be required.

If the available time extends to one month, then it would be advisable to regenerate the synthetic datasets using less simplified assumptions in the mathematical SLE model, as well as to compute the datasets directly on finer icosahedral meshes from the spherical harmonics domain. By adopting these more complex SLE models with fewer simplifying assumptions, the network may be able to better discern and learn patterns within the data. Consequently, this could improve both the precision of the predictions and the overall training process. Consideration should then also be given to refining the ice loading input through a more comprehensive evaluation and selection of ice models. In the thesis, for instance, the ICE-5G, ICE-6G, and ICE-7G models were all given equal weight during the generation process. However, later generations of ice models were developed using higher-quality data, which suggests they may be more accurate. Developing a method that accounts for the differences in realism and accuracy among these models could improve the ice load histories as well. The second potential area for improvement that fits in the one month timeframe involves the handling of mesh resolution. The number of recursive subdivisions could be increased, which would inherently raise the resolution of the model, providing a more detailed representation of the ice load. However, this step will require a significant reconfiguration of the U-Net architecture, likely necessitating a larger batch size and less channel-wise reduction between the layers and more convolutions within the U-net building blocks. The computational requirements for this step will be substantial, because one more subdivision will quadruple the number of vertices in the mesh. However, a technique called frequency division could be applied as an alternative to recursive subdivisions. Frequency subdivision has been demonstrated to offer a more gradual transition between resolutions, which might be advantageous in discerning and learning intricate patterns within the dataset. This method was suggested by Dahl et al., 2014 and could be particularly beneficial for this study. For instance, frequency subdivision allows for as many as fifteen incremental steps between resolutions of 2,562 and 10,242 vertices. Improving the ice load an RSL datasets and increasing the resolution offer an exciting route for advancing the emulators' capabilities.

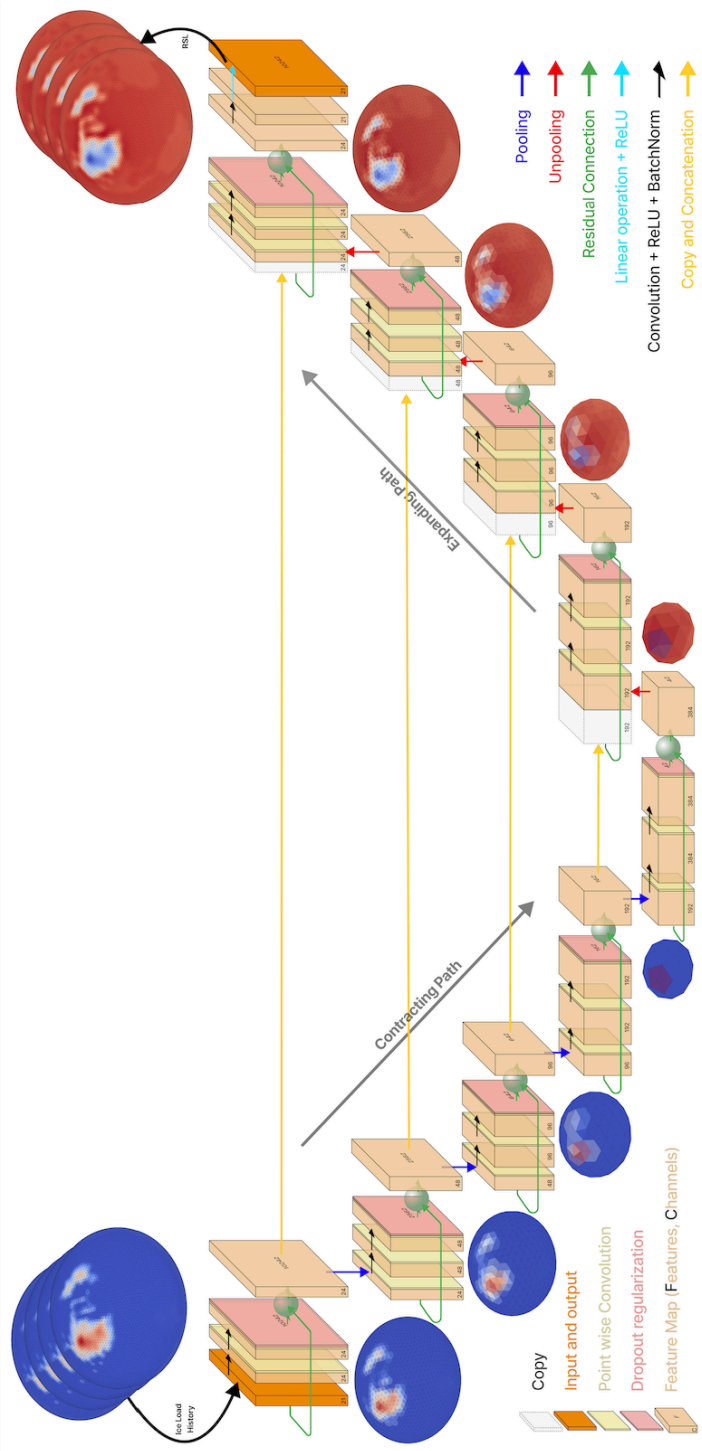
For an even more extended period, such as multiple months or even a full thesis project, the ideal focus would be on addressing the major constraint of Earth parameter variation, as discussed in Section 5.2. This would entail working towards equipping the emulators with the ability to handle different

Earth models for multiple or one ice load history, thereby enhancing their versatility and alignment with traditional GIA models. Including this variation within the emulators would significantly increase their scientific usefulness, because of the impact the Earth model has on the RSL response to ice loading. However, the original U-Net architecture is designed for video-to-video regression and does not readily accommodate the addition of Earth parameters. An initial attempt to incorporate Earth parameter variation was made during the thesis. To address this, a subfield of deep learning called feature fusion was explored. Feature fusion combines or merges features from different layers or branches within a neural network, creating a space for additional parameters. While various strategies exist for feature fusion, ranging from concatenation and branching to attention mechanisms, branching was the preferred method due to its minimal impact on memory requirements and its physical relevance. The branching was attempted to integrate Earth parameters into the GIA emulator using a fully-connected path. This created a feature block equal to the size of the deepest feature block in the U-Net. The experiment was conducted using a single ice load history with RSL output, created using 270 different Earth models. Each Earth model comprised 24 parameters; six for the boundaries of the lithosphere, upper and lower mantle, and four parameters for radius, density, shear modulus, and viscosity. However, the U-Net architecture in this configuration posed a challenge. The skip connections, which transfer learning between equivalent layers in the contracting and expanding paths, resulted in stationary predictions across samples that only change over time and training epochs. This suggested that the main branch of the U-Net was overpowering the additional learning in the side branch during forward, and consequently, back-propagation. While this issue was not fully resolved, an alternative approach is proposed for future work: a combination of feature fusion with transfer learning.

Transfer learning is a another technique in deep learning where a model developed for one task is reused as the starting point for a model on a second task. This is especially effective when the first task is general and the second task is more specific. The method is often implemented on a large-scale image recognition task, like ImageNet, to start a model for identifying specific objects in images (Kornblith et al., 2019). In the context of this research, the trained emulator in the first experiment, which has learned the general patterns of GIA modeling, could serve as the starting point for a new model that incorporates varying Earth parameters. The new model would then 'only' need to learn the more specific patterns relating to the variations in Earth parameters, because the weights in the main branch are pre-trained and frozen. This approach could potentially reduce the computational requirements and the amount of data needed for training. Though the complexity of the contribution of RSL variation induced by the Earth parameter variation should not be underestimated. Feature fusion would then be used in conjunction with transfer learning to integrate the Earth parameters into the new model. The Earth parameters would be inputted via a separate branch in the network, allowing them to influence the model's predictions without overpowering the main branch's learning, as occurred in the initial attempt. One approach would be to create fully-connected branches with a final size equal to the final feature blocks in the layers of the U-Net and connecting them. The branches would then all be separately connected to the Earth parameter branch so they can each learn one respective level of detail in the contracting and expanding paths. Together, transfer learning and feature fusion could potentially provide a robust method for incorporating Earth models into the GIA emulators. This would further enhance their performance, versatility, and alignment with traditional GIA models, making them more valuable for scientific applications.

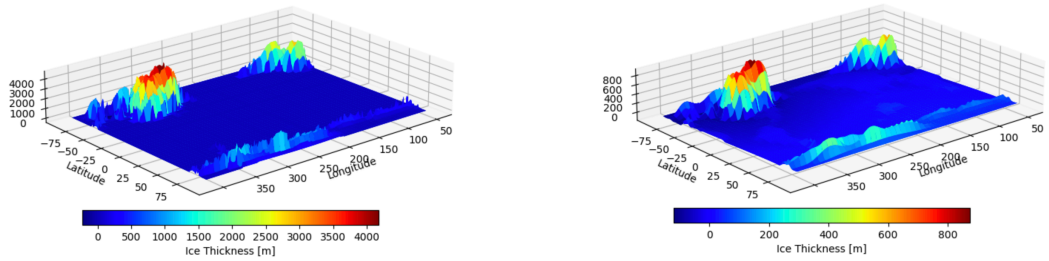
Appendix A

U-Net Architecture Schematic



Appendix B

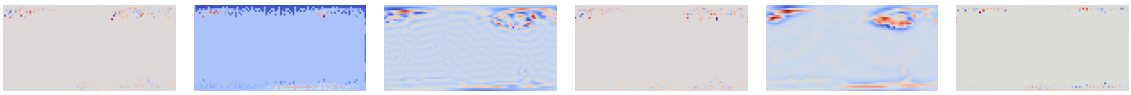
Supplementing Figures



(a) Artificial cumulative ice load history example at 21,000 years BP

(b) Artificial cumulative RSL example at 21,000 years BP

Figure B.1: Surface plot examples showing the smooth RSL data and sharpness of ice loading data indicate distinct characteristics of each dataset. Observed characteristic is inherent to RSL, whereas the ice loading is reconstructed using proxies. RSL and cumulative ice loading are corresponding to one artificial sample at LGM. Ice loading includes values equal to zero, in contrast to RSL data because only continental ice sheets are considered



(a) Nearest-neighbor 10,242 vertices

(b) RectBivariateSpline interpolation

(c) Spherical Harmonics L_{max}= 26

(d) HEALPix 12,288 pixels

(e) Ordinary Kriging interpolation

(f) Nearest-neighbor 40,962 vertices

Figure B.2: Visual comparison of the different interpolation methods. The figures show mapping error for the same random artificial RSL at LGM of the downsampled equiangular map. The L_{max}= 26 degree is related to half of the shortest dimension of the equiangular map. Scales are left out for enhanced visibility, though can be interpreted from the MAE provided in Table 3.1

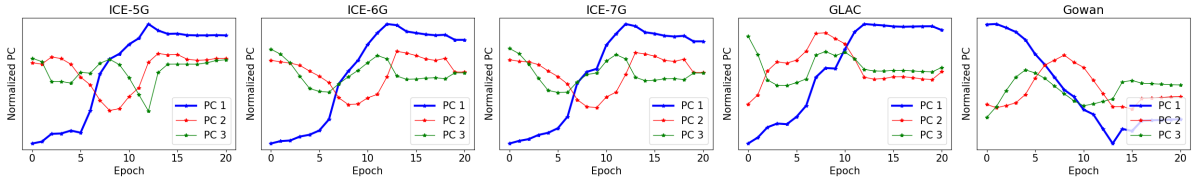


Figure B.3: Normalized PC time series for all ice models. Except for the Gowan ice model it appears as if the first principle component has explained the deglaciation trend. This, however, might be coincidence when examining the Gowan ice model. Though the deglaciation trend for the Gowan ice model is significantly different as can be observed Figure 2.1b

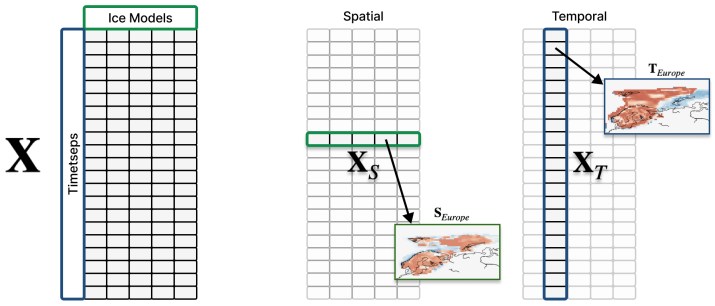


Figure B.4: Schematic indicating the data matrices for WPCA on the spatial and temporal pattern as part of the original data matrix. For the WPCA spatial pattern of X the matrix X_S is applied once for every timestep. Similarly, for the temporal pattern X_T is applied once for every ice model. In the figure each gray block represents a regional map of ice load values, here Fennoscandia. After reconstruction there is a PC map for every timestep and ice model from both the spatial and temporal WPCA

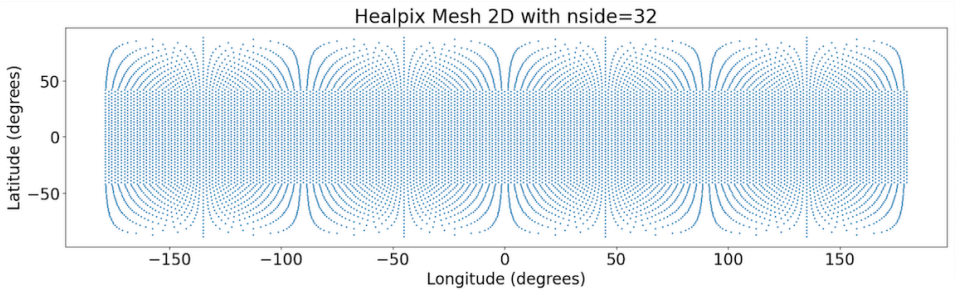


Figure B.5: 2D grid of the HEALPix grid showing the finer and more irregular mesh structure near the poles represented by higher latitudes

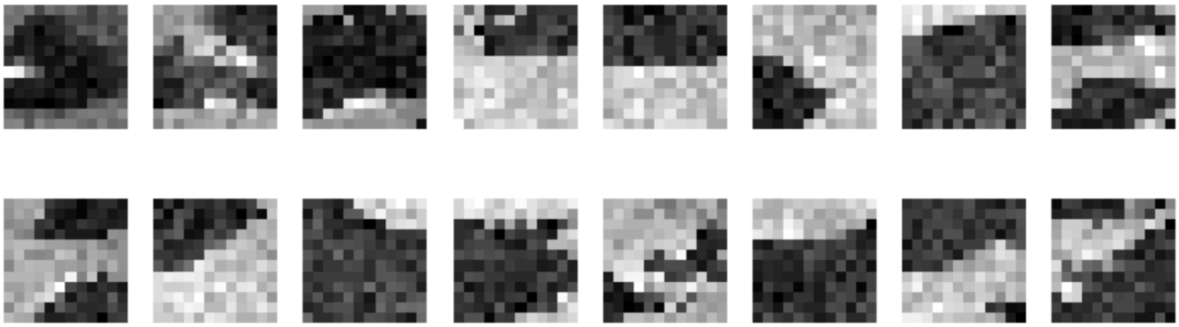


Figure B.6: Visualization of a random selection of sixteen out 24 square 12x12 filters from the first convolution. The convolution is part of a three-layer U-Net 2D CNN that predicts RSL from ice load history using constant Earth properties on a downsampled (48, 96) latitude-longitude grid. The interpretability of the features detected by the filters are difficult to quantify with certainty, though it appears as if the filters learn the edges and slopes of the ice sheet extent. Which is plausible because the filters are of the first convolution, which are applied directly to the ice load input, the filters are thus sliding windows with equiangular size of 12x12 degrees

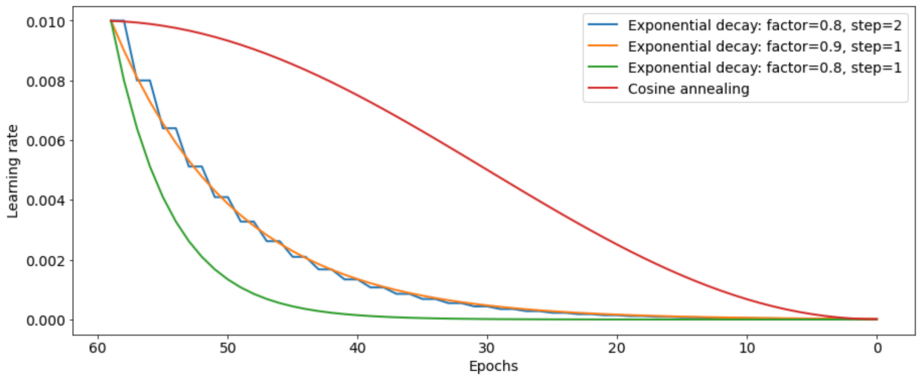
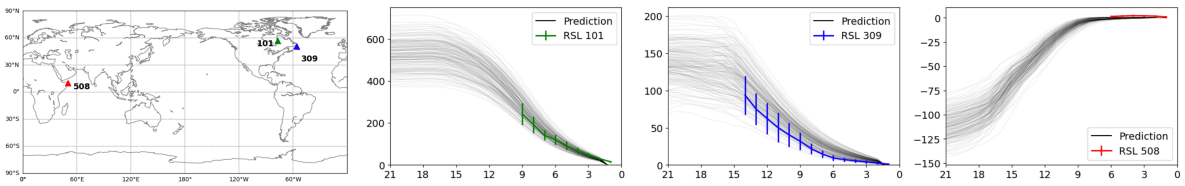


Figure B.7: Figure showing different learning rate schedulers. Cosine annealing poses a more gradual decay compared to the more standard learning rate schedulers that incorporate exponential decay. This allows the learning process to take a smoother, more controlled approach. Cosine annealing is particularly useful for training deep neural networks and is known to improve convergence and generalization. It is also commonly used with Adam optimizer and long training schedules. All of which apply to the training conducted in the thesis



(a) RSL site locations

(b) Prediction evaluations of spectral model. From left to right: location 101, 309 and 508

Figure B.8: RSL predictions at three RSL site locations for the spectral network. Locations defined by latitude and longitude coordinates and depicted on global map. The increasing larger range of the predictions per timestep is explained by the observed prediction error in Figure 4.3b

References

Araya-Polo, M., J. Jennings, A. Adler, and T. Dahlke (2018). Deep-learning tomography. In: *The Leading Edge* 37.1, pp. 58–66. DOI: 10.1190/tle37010058.1. eprint: https://doi.org/10.1190/tle37010058.1. URL: https://doi.org/10.1190/tle37010058.1.

Bagge, M., E. Boergens, K. Balidakis, V. Klemann, and H. Dobslaw (Feb. 2023). Validation of Modelled Uplift Rates with Space Geodetic Data. In: *EGU General Assembly 2023* G3.3.EGU23-3351. DOI: 10.5194/egusphere-egu23-3351. URL: https://doi.org/10.5194/egusphere-egu23-3351.

Barletta, V. R. and A. Bordoni (Nov. 2013). Effect of different implementations of the same ice history in GIA modeling. In: *Journal of Geodynamics* 71, pp. 65–73. ISSN: 02643707. DOI: 10.1016/j.jog.2013.07.002.

Brownlee, J. (Aug. 2019). A Gentle Introduction to Mini-Batch Gradient Descent and How to Configure Batch Size. In: *MachineLearningMastery.com*. URL: https://machinelearningmastery.com/gentle-introduction-mini-batch-gradient-descent-configure-batch-size/.

Cohen, T. S., M. Geiger, J. Koehler, and M. Welling (Jan. 2018). Spherical CNNs. In: URL: http://arxiv.org/abs/1801.10130.

Crowley, T. J. (Sept. 1995). Ice Age terrestrial carbon changes revisited. In: *Global Biogeochemical Cycles* 9.3, pp. 377–389. DOI: 10.1029/95GB01107.

Dahl, V. A., A. B. Dahl, and R. Larsen (2014). "Surface Detection Using Round Cut". In: *2014 2nd International Conference on 3D Vision*. Vol. 2, pp. 82–89. DOI: 10.1109/3DV.2014.60.

Dawson-Haggerty et al. (n.d.). *trimesh*. Version 3.2.0. URL: https://trimsh.org/.

Defferrard, M., M. Milani, F. Gusset, and N. Perraudin (Dec. 2020). DeepSphere: a graph-based spherical CNN. In: URL: http://arxiv.org/abs/2012.15000.

Delchambre, L. (Dec. 2014). Weighted principal component analysis: a weighted covariance eigendecomposition approach. In: DOI: 10.1093/mnras/stu2219. URL: http://arxiv.org/abs/1412.4533%20http://dx.doi.org/10.1093/mnras/stu2219.

Driscoll, J. and D. Healy (1994). Computing Fourier Transforms and Convolutions on the 2-Sphere. In: *Advances in Applied Mathematics* 15.2, pp. 202–250. ISSN: 0196-8858. DOI: https://doi.org/10.1006/aama.1994.1008. URL: https://www.sciencedirect.com/science/article/pii/S0196885884710086.

Farrell, W. E. and J. A. Clark (1976). *On Postglacial Sea Level*, pp. 647–667.

Farrell, W. (1972). Deformation of the Earth by surface loads. In: *Reviews of Geophysics* 10.3, pp. 761–797.

Goodfellow, I., Y. Bengio, and A. Courville (2016). *Deep Learning*. MIT Press. ISBN: 9780262035613. URL: https://lccn.loc.gov/2016022992.

Górski, K. M. et al. (2005). HEALPix: A Framework for High-Resolution Discretization and Fast Analysis of Data Distributed on the Sphere. In: *The Astrophysical Journal* 622 (2), pp. 759–771. DOI: 10.1086/427976. URL: https://doi.org/10.1086/427976.

Gowan, E. J. et al. (2016). ICESHEET 1.0: a program to produce paleo-ice sheet reconstructions with minimal assumptions. In: *Geoscientific Model Development* 9.5, pp. 1673–1682. DOI: 10.5194/gmd-9-1673-2016. URL: https://gmd.copernicus.org/articles/9/1673/2016/.

Gowan, E. J. (2019). *Global ice sheet reconstruction for the past 80000 years*. data set. Supplement to: Gowan, Evan J; Zhang, Xu; Khosravi, Sara; Rovere, Alessio; Stocchi, Paolo; Hughes, Anna L C; Gyllencreutz, Richard; Mangerud, Jan; Svendsen, John Inge; Lohmann, Gerrit (2021): A

new global ice sheet reconstruction for the past 80 000 years. Nature Communications, 12, 1199, https://doi.org/10.1038/s41467-021-21469-w. DOI: 10.1594/PANGAEA.905800. URL: https://doi.org/10.1594/PANGAEA.905800.

He, K., X. Zhang, S. Ren, and J. Sun (2015). Deep residual learning. In: *Image Recognition* 7.

Hexmoor, H. (2015). "Chapter 6 - Diffusion and Contagion". In: H. Hexmoor, ed. *Computational Network Science*. Emerging Trends in Computer Science and Applied Computing. Boston: Morgan Kaufmann, pp. 45–64. ISBN: 978-0-12-800891-1. DOI: https://doi.org/10.1016/B978-0-12-800891-1.00006-8. URL: https://www.sciencedirect.com/science/article/pii/B9780128008911000068.

Jin, K. H., M. T. McCann, E. Froustey, and M. Unser (Sept. 2017). Deep Convolutional Neural Network for Inverse Problems in Imaging. In: *IEEE Transactions on Image Processing* 26 (9), pp. 4509–4522. ISSN: 10577149. DOI: 10.1109/TIP.2017.2713099.

Jolliffe, I. T. and J. Cadima (2016). Principal component analysis: a review and recent developments. In: *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* 374.2065, p. 20150202. DOI: 10.1098/rsta.2015.0202. URL: https://royalsocietypublishing.org/doi/abs/10.1098/rsta.2015.0202.

Jordan, J. (Mar. 2023). Setting the learning rate of your neural network. In: URL: https://www.jeremyjordan.me/nn-learning-rate/.

Karpatne, A., I. Ebert-Uphoff, S. Ravela, H. A. Babaie, and V. Kumar (2019). Machine Learning for the Geosciences: Challenges and Opportunities. In: *IEEE Transactions on Knowledge and Data Engineering* 31.8, pp. 1544–1554. DOI: 10.1109/TKDE.2018.2861006.

Kingma, D. P. and J. Ba (2014). Adam: A method for stochastic optimization. In: *arXiv preprint arXiv:1412.6980*.

Kornblith, S., J. Shlens, and Q. V. Le (2019). "Do better imagenet models transfer better?" In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 2661–2671.

Krachmalnicoff, N. and M. Tomasi (Aug. 2019). Convolutional neural networks on the HEALPix sphere: a pixel-based algorithm and its application to CMB data analysis. In: *Astronomy and Astrophysics* 628, A129. DOI: 10.1051/0004-6361/201935211. arXiv: 1902.04083. URL: https://ui.adsabs.harvard.edu/abs/2019A%5C&A...628A.129K.

Lemke, P. et al. (2007). "Observations: Changes in Snow, Ice and Frozen Ground". In: *Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate Change*. Ed. by S. Solomon et al. Cambridge, United Kingdom and New York, NY, USA: Cambridge University Press.

Levie, R., F. Monti, X. Bresson, and M. M. Bronstein (May 2017). CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters. In: URL: http://arxiv.org/abs/1705.07664.

Lin, Y., P. Whitehouse, and A. Valentine (May 2023). A Deep-learning Based Glacial Isostatic Adjustment Sea-level Emulator. In: *EGU General Assembly 2023*.

Martinec, Z. et al. (Oct. 2018). A benchmark study of numerical implementations of the sea level equation in GIA modelling. In: *Geophysical Journal International* 215 (1), pp. 389–414. ISSN: 1365246X. DOI: 10.1093/gji/ggy280.

Mitrovica, J. X. and W. R. Peltier (1991). On postglacial geoid subsidence over the equatorial oceans. In: *Journal of Geophysical Research* 96 (B12). ISSN: 01480227. DOI: 10.1029/91jb01284.

Nair, V. and G. E. Hinton (2010). "Rectified linear units improve restricted boltzmann machines". In: *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 807–814.

Odena, A., V. Dumoulin, and C. Olah (2016). Deconvolution and checkerboard artifacts. In: *Distill* 1.10, e3.

Paszke, A. et al. (2017). Automatic differentiation in PyTorch. In:

Pedregosa, F. et al. (2011). Scikit-learn: Machine Learning in Python. In: *Journal of Machine Learning Research* 12, pp. 2825–2830.

Peltier, W., D. F. Argus, and R. Drummond (2015). Space geodesy constrains ice age terminal deglaciation: The global ICE-6G-C (VM5a) model. In: *Journal of Geophysical Research: Solid Earth* 120 (1), pp. 450–487. ISSN: 21699356. DOI: 10.1002/2014JB011176.

Peltier, W. R. (2004). Global glacial isostasy and the surface of the ice-age Earth: the ICE-5G (VM2) model and GRACE. In: *Annual Review of Earth and Planetary Sciences* 32.1, pp. 111–149. DOI: 10.1146/annurev.earth.32.082503.144359.

Preethi, P. and H. Viswanath (2021). Denoising and Segmentation of Epigraphical Scripts. In: *arXiv preprint arXiv:2107.11801*. DOI: https://doi.org/10.48550/arXiv.2107.11801. URL: https://arxiv.org/abs/2107.11801.

Ronneberger, O., P. Fischer, and T. Brox (2015). "U-net: Convolutional networks for biomedical image segmentation". In: *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*. Springer, pp. 234–241.

Roy, K. (2017). *High quality constraints on the glacial isostatic adjustment process over North America: The ICE-7G NA (VM7) model*.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning representations by back-propagating errors. In: *nature* 323.6088, pp. 533–536.

Santurkar, S., D. Tsipras, A. Ilyas, and A. Madry (May 2018). How Does Batch Normalization Help Optimization? In: URL: http://arxiv.org/abs/1805.11604.

Schmidt, P., B. Lund, J. O. Näslund, and J. Fastook (May 2014). Comparing a thermo-mechanical Weichselian Ice Sheet reconstruction to reconstructions based on the sea level equation: Aspects of ice configurations and glacial isostatic adjustment. In: *Solid Earth* 5 (1). Comparing Ice Models, pp. 371–388. ISSN: 18699529. DOI: 10.5194/se-5-371-2014.

Schotman, H. (2007). *Shallow-Earth Rheology from Glacial Isostasy and Satellite Gravity a sensitivity analysis for GOCE*. Faculty of Aerospace Engineering , Delft University of Technology, pp. 17–28.

Simon, K. M. and R. E. M. Riva (2020). Uncertainty Estimation in Regional Models of Long-Term GIA Uplift and Sea Level Change: An Overview. In: *Journal of Geophysical Research: Solid Earth* 125.8, e2019JB018983. DOI: https://doi-org.tudelft.idm.oclc.org/10.1029/2019JB018983. URL: https://agupubs-onlinelibrary-wiley-com.tudelft.idm.oclc.org/doi/abs/10.1029/2019JB018983.

Smith, S. L. (Nov. 2017). *Don't Decay the Learning Rate, Increase the Batch Size*. URL: https://arxiv.org/abs/1711.00489.

Spada, G. (2017). Glacial Isostatic Adjustment and Contemporary Sea Level Rise: An Overview. In: *Surveys in Geophysics* 38, pp. 153–185.

Suganuma, Y. et al. (Oct. 2020). Perspectives on a Seamless Marine-lake Sediment Coring Study in East Antarctica. In: *Journal of Geography (Chigaku Zasshi)* 129 (5), pp. 591–610. ISSN: 0022-135X. DOI: 10.5026/jgeography.129.591.

Tarasov, L. and W. R. Peltier (2002). Greenland glacial history and local geodynamic consequences. In: *Geophysical Journal International* 150 (1), pp. 198–229. ISSN: 0956540X. DOI: 10.1046/j.1365-246X.2002.01702.x.

Tarasov, L., A. Hughes, et al. (2014). "The global GLAC-1c deglaciation chronology, melwater pulse 1-a, and a question of missing ice". In: *IGS Symposium: Contribution of Glaciers and Ice Sheets to Sea-Level Change, Chamonix, France*, pp. 26–30.

Targ, S., D. Almeida, and K. Lyman (2016). Resnet in resnet: Generalizing residual architectures. In: *arXiv preprint arXiv:1603.08029*.

Van der Plas, J. (Feb. 2016). *GitHub - jakevdp/wpca: Weighted Principal Component Analysis (PCA) in Python*. URL: https://github.com/jakevdp/wpca.

van der Wal, W., P. Wu, H. Wang, and M. G. Sideris (2010). Sea levels and uplift rate from composite rheology in glacial isostatic adjustment modeling. In: *Journal of Geodynamics* 50.1, pp. 38–48. ISSN: 0264-3707. DOI: https://doi.org/10.1016/j.jog.2010.01.006. URL: https://www.sciencedirect.com/science/article/pii/S0264370710000190.

Virtanen, P. et al. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. In: *Nature Methods* 17, pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

Whitehouse, P. L. (2018). Glacial isostatic adjustment modelling: historical perspectives, recent advances, and future directions. In: *Earth Surface Dynamics* 6.2, pp. 401–429. DOI: 10.5194/esurf-6-401-2018. URL: https://esurf.copernicus.org/articles/6/401/2018/.

Zhao, F. et al. (2019). Spherical U-Net on Cortical Surfaces: Methods and Applications. In: *Information Processing in Medical Imaging*. Ed. by A. C. S. Chung, J. C. Gee, P. A. Yushkevich, and S. Bao.