FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

High performance computing machine learning architecture for the LISA Global Fit

Tomás Miranda de Figueiredo Sarmento

Mestrado em Engenharia Informática e Computação

Supervisor: Prof. Paulo Garcia

Second Supervisor: Prof. Jaime Cardoso

July 22, 2026

High performance computing machine learning architecture for the LISA Global Fit

Tomás Miranda de Figueiredo Sarmento

Mestrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

President: Prof. Pedro Souto
Referee: Prof. Paulo Garcia
Referee: Prof. Arianna Foschi

July 22, 2026

Abstract

BACKGROUND: The upcoming Laser Interferometer Space Antenna (LISA) mission will initiate a new era in gravitational wave astronomy, observing a signal-dominated regime characterized by tens of thousands of overlapping sources. Simultaneously identifying, separating, and characterizing these signals requires a complex, modular Bayesian inference pipeline known as the Global Fit. However, traditional Markov Chain Monte Carlo (MCMC) methods in this pipeline are heavily bottlenecked by millions of computationally expensive, physics-based likelihood evaluations, presenting significant performance limitations even when executed on modern supercomputing infrastructures like the Deucalion High-Performance Computer (HPC).

OBJECTIVE: This thesis designs and evaluates a prototype machine learning surrogate model to accelerate these evaluations for a single Galactic Binary (GB) in a localized 2.95–3.00 mHz frequency band.

METHODS: The core of the work involves developing a dual-branch feed-forward neural network that maps physical waveform parameters and pre-processed frequency-domain data directly to predicted log-likelihood values. To ensure the model generalizes effectively, we systematically analyze different data generation and dataset balancing strategies. To validate performance, the trained surrogate was integrated as a prototype into an active ensemble MCMC sampler.

RESULTS: Our results show that high structural diversity in simulated signal backgrounds is essential to prevent the network from memorizing individual noise realizations. Standalone testing demonstrates a clear computational speedup over the analytical baseline, though end-to-end integration reveals system-level overheads typical of mixed-framework pipelines.

CONCLUSIONS: Ultimately, this prototype successfully establishes the feasibility of neural surrogates for accelerating the LISA Global Fit, while defining a clear, practical path for future GPU-bound scaling.

Keywords: MCMC • LISA • Machine Learning • Galactic Binary • Gravitational-Waves

UN Sustainable Development Goals

Traditional Bayesian inference frameworks, such as the Markov Chain Monte Carlo (MCMC) algorithms used for the Laser Interferometer Space Antenna (LISA) Global Fit are highly energy-intensive, requiring millions of repetitive, expensive physics-based likelihood evaluations. This thesis directly addresses this challenge by introducing portable, lightweight machine learning surrogate models to substitute these expensive analytical calculations.

By optimizing the software architecture and leveraging GPU-accelerated frameworks, this research significantly reduces the overall core-hours and energy required to process complex astrophysical datasets. Consequently, this work directly contributes to global sustainability targets in energy efficiency and technological innovation.

The specific Sustainable Development Goals mentioned have the following names:

SDG 9 Build resilient infrastructure, promote inclusive and sustainable industrialization and foster innovation

SDG 12 Ensure sustainable consumption and production patterns

SDG 13 Take urgent action to combat climate change and its impacts

SDG	Target	Contribution	Performance Indicators and Metrics
9	9.5	Upgrading the technological capabilities of scientific research infrastructures by creating a portable, highly parallelizable, and scalable machine learning framework.	- High scaling efficiency across diverse HPC partitions.
12	12.2	Maximizing computational resource efficiency by ensuring that expensive public supercomputing nodes are not wasted on slow, analytical loop evaluations.	- Reduction in CPU/GPU core-hours spent on idle or slow processes.
13	13.3	Mitigating the carbon footprint associated with data center electricity draw by reducing the computational complexity and total execution time of the Global Fit pipeline.	- Reduction in estimated carbon footprint (CO_2 equivalent) per data analysis run.

Acknowledgements

I would like to express my gratitude to Professors Paulo Garcia, Francisco Pimenta and Jaime Cardoso for the guidance given during the development of this master's thesis.

I am grateful to the LISA consortium, who provided a prototype and the documentation needed for my work, as well as my thanks to Dr. Antoine Basset for his contextual support and for the broad guidance on what is already being worked on and possible challenges.

I also wish to express my gratitude for the access to computational resources on the Deucalion supercomputer, awarded under the EuroHPC Development Access Call (Proposal No. EHPC-DEV-2026D02-075).

I would also like to express my gratitude to my close friends who helped me to keep it together during a challenging part of the thesis: Tomás Câmara, João Belshior, Rodrigo Moucho and Rodrigo Povoa.

Another acknowledgment to the friends whom I met during my Erasmus at TUM and my roommates.

I would also like to thank my family members, who always believed in me, especially my sisters and my twin.

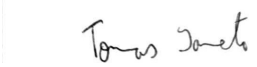
Last but not least, I would like to thank my cute pets, Sabi, as well as Kiwi (a friend's pet), for the emotional support they gave me during this challenging part of my life.

Tomás Sarmento

Declaration of honour

I declare that this dissertation is my own work and has not previously been submitted or assessed at this or any other institution. References to other authors (statements, ideas, thoughts), as well as the use of published works, strictly comply with the rules of attribution and are duly identified in the text and in the bibliographic references, in accordance with the applicable referencing standards. I am aware that the practice of plagiarism or self-plagiarism constitutes an academic offence, as established in the U.Porto Code of Ethics.
I also declare that I have used Artificial Intelligence tools to complete part(s) of this dissertation, and that this use and its results are duly noted and have been carefully checked for errors, inaccuracies, unfounded attributions or other forms of bias in content and arguments, as well as determining authorship.

Tomás Miranda de Figueiredo Sarmento



"A man cannot step into the same river twice, because it is not the same river, and he is not the same man."
"Δεν μπορείς να μπεις στο ίδιο ποτάμι δύο φορές, γιατί δεν είναι το ίδιο ποτάμι και δεν είσαι ο ίδιος άνθρωπος."

Heraclitus

Contents

List of Figures

List of Tables

List of Acronyms

CNN Convolutional Neural Network
CPU Central Processing Unit
DFT Discrete Fourier Transform
DRAM Dynamic Random Access Memory
DSRM Design Science Research Methodology
ECSS European Cooperation for Space Standardisation
ESA European Space Agency
FFT Fast Fourier Transform
GB Galactic Binary
GPU Graphics Processing Unit
GW Gravitational Wave
HPC High-Performance Computer
LISA Laser Interferometer Space Antenna
LLM Large Language Model
MAE Mean Absolute Error
MAF Masked Autoregressive Flow
MBHB Massive Black Hole Binary
MCMC Markov Chain Monte Carlo
ML Machine Learning
MLP Multi-Layer Perceptron
MSE Mean Squared Error
NRPT Non-Reversible Parallel Tempering
PT Parallel Tempering
RAM Random Access Memory
ReLU Rectified Linear Unit
RJ-MCMC Reversible Jump MCMC
SNR Signal-to-Noise Ratio
TDI Time-Delay Interferometry
VRAM Video Random Access Memory

Chapter 1

Introduction

1.1 Context

The Laser Interferometer Space Antenna (LISA) is a groundbreaking mission as the sole space-based gravitational wave observatory. LISA will be launched in 2035 as a European Space Agency (ESA) project. LISA will reveal a whole new window to the Universe by detecting low-frequency Gravitational Wave (GW)s, subtle ripples in spacetime as predicted by Einstein's general theory of relativity. This will enable scientists to trace the growth and development of cosmic structures over time and explore fundamental physics in unprecedented ways [11, 13]. LISA is not just one spacecraft but a constellation of three. They will trail Earth in its orbit around the Sun, forming an exquisitely accurate equilateral triangle in space. Each side of the triangle will be 2.5 million km long (more than six times the Earth-Moon distance), and the spacecraft will exchange laser beams over this distance. Every component of the project must integrate seamlessly to achieve its ambitious goals [10].

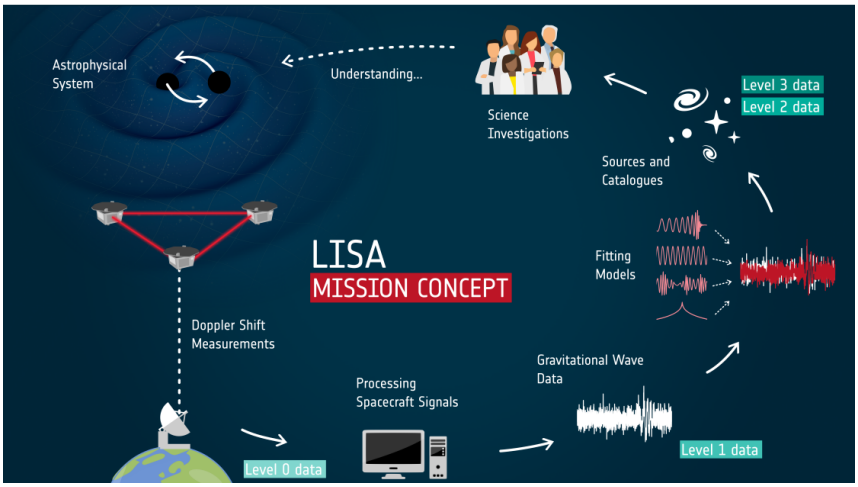


Figure 1.1: Overview of the mission data analysis concept. Extracted from [7]

1.2 Motivation

Driving the research are two key dimensions:

Scientific Opportunity: The LISA mission will open a new window into the universe by observing low-frequency GWs, revealing mysteries of supermassive black holes, cataclysmic cosmic events, and Galactic Binaries. Extraction of these findings is entirely dependent on being able to decouple the coupled signals in the data, and that is what the Global Fit framework aims to do.

Technological Requirement: Global Fit today is a computationally massive method to execute on expensive supercomputers. This is a major restraint that can delay scientific results. The work aims to optimize this process through high-performance computing and machine learning to allow the mission to produce timely results, and to save energy [28].

1.3 Problem

Gap in Engineering: The prototype lacks the robust engineering that will support production deployment. Given the lack of documentation, standardized interfaces, and fault tolerance, it cannot be safely incorporated into the distributed data processing system. The project integrates European Cooperation for Space Standardisation (ECSS)[12] and ESA coding standards in the prototype development.

- **Performance Limitations**: The current implementation exhibits significant performance constraints, characterized by excessive execution times and high memory consumption. These limitations render the software computationally prohibitive and incompatible with the operational throughput required for the mission.
- **Issues in Integration with High-Performance Computer (HPC)**: The system is not tuned to performance for the Deucalion supercomputer environment. Scheduling jobs, parallel processing, and data administration problems reduce operational efficiency.
- **Algorithmic Bottlenecks**: Traditional log-likelihood computation methods are computationally expensive, creating a major bottleneck in the analysis pipeline for data.
- **Markov Chain Monte Carlo (MCMC)**: Traditional MCMC algorithms take a majority of the execution time to converge to the true parameters.

1.4 Research questions

- **Optimization of Bottlenecks:** Which components of the pipeline have higher priority for optimization, and evaluating the feasibility of their optimization.
- **Computational Performance Characterization**: What are the results of computational bottlenecks in the current Global Fit implementation when executed on realistic LISA data

simulations, and how do they change with increasing source complexity and observation time?

- **Machine Learning Acceleration:** To what extent can machine learning models (Multi-Layer Perceptron (MLP), Convolutional Neural Network (CNN)), numerically approximate the log-likelihood function in the Global Fit setting, and what potential acceleration of the overall MCMC computation results in the required scientific accuracy?
- **Optimization Validation:** How does the computational efficiency and scientific accuracy of the optimized Global Fit framework compare to the original prototype when applied to benchmark data sets with known source parameters?

1.5 Contributions

The primary goal of this dissertation is to design, develop, and evaluate a portable machine learning surrogate to accelerate likelihood evaluations in the LISA Global Fit pipeline. To this end, the main contributions of this work are:

- **High-Diversity Dataset Generation:** The development and execution of an automated data generation pipeline on the Deucalion HPC cluster, resulting in a novel dataset of 200,000 samples configured with various signal-to-parameter ratios to study the effects of signal diversity on surrogate generalization.
- **Dual-Branch Neural Surrogate Architecture:** The design and implementation of a custom dual-branch feed-forward neural network that successfully fuses low-dimensional physical wave parameters with high-dimensional frequency-domain instrument representations.
- **Target-Based Resampling Analysis:** A systematic evaluation of data-balancing and resampling techniques (downsampling, oversampling, and stratified binning) to manage highly skewed log-likelihood spaces near the noise floor.

1.6 Structure of the Document

The remainder of this dissertation is organized as follows:

- **Chapter 1** (this chapter) provides the context, motivation, engineering gaps, and research questions guiding the thesis.
- **Chapter 2 (Background)** reviews the fundamental concepts of Bayesian inference, MCMC methods, gravitational-wave source types, and deep learning techniques relevant to the surrogate model.

- **Chapter 3 (State of the Art)** examines related work in machine learning acceleration for gravitational-wave data analysis, focusing on neural surrogates, autoencoders, Normalizing Flows and HPC execution environments.
- **Chapter 4 (Methodology)** details the design of the dual-branch neural network, the dataset creation pipeline on the Deucalion HPC, feature extraction steps, and the target-based resampling strategies.
- **Chapter 5 (Results and Discussion)** evaluates the surrogate model's approximation accuracy, analyzes dataset diversity and hyperparameter configurations, and discusses the performance speedups and integration on the pipeline.
- **Chapter 6 (Conclusions and Future Work)** summarizes the key findings, identifies the primary computational constraints discovered, and outlines future pathways for Graphics Processing Unit (GPU)-bound optimization.

Chapter 2

Background

2.1 Machine Learning

2.1.1 Conditional Normalizing Flows

A Normalizing Flow transforms a simple base distribution (e.g., a standard Gaussian) into a complex target density through a series of invertible, differentiable mappings. A *conditional* flow extends this by making the transformations depend on a conditioning variable, like the parameters θ of a Galactic Binary (GB) and the Fast Fourier Transform (FFT) of the GW. Thus, a Conditional Normalizing Flow defines a parametric family $q_\phi(\mathbf{x} \mid \theta)$ that can be trained to approximate the true log-likelihood $p(\mathbf{x} \mid \theta)$. Once trained, evaluating the log-likelihood for any $(\mathbf{x}, \theta)$ pair is fast and differentiable.

2.1.2 Autoencoder Architectures

- **The Encoder (E_ψ):** A deep neural network that projects the high-dimensional input data x (e.g., the concatenated A and E Time-Delay Interferometry (TDI) channels or an FFT) into a lower-dimensional latent space vector $z = E_\psi(x)$. For example, an input dimension of 14,400 can be compressed to a latent vector of size 128.
- **The Decoder (D_ξ):** A symmetric network that attempts to reconstruct the original input from the latent vector, producing an output $y = D_\xi(z)$.

2.1.3 Model Evaluation Functions

Loss functions quantify the discrepancy between the true physical log-likelihoods and the surrogate model predictions.

2.1.3.1 Mean Squared Error (MSE)

Mean Squared Error (MSE) measures the average squared difference between the predicted values and the actual values. It is highly sensitive to outliers:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.1}$$

where:

- y_i is the actual physical log-likelihood for the i-th observation,
- $\hat{y}_i$ is the predicted log-likelihood generated by the surrogate model for the i-th observation,
- n is the total number of observations

2.1.3.2 Mean Absolute Error (MAE)

Mean Absolute Error (MAE) measures the average difference between the predicted values and the actual values. It is more robust dealing with outliers than MSE:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.2}$$

2.1.3.3 Coefficient of Determination (R^2)

The Coefficient of Determination, denoted as R^2, measures the proportion of variance in the target variable that is predictable from the surrogate model's predictions. Unlike absolute error metrics like MSE or MAE, R^2 is a dimensionless, relative metric that evaluates the model's performance against a simple baseline that constantly predicts the mean of the actual values:

$$R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2} \tag{2.3}$$

where:

- $\bar{y}$ is the mean of the actual physical log-likelihoods.

An R^2 value of 1.0 indicates that the model perfectly predicts the actual values, whereas a value of 0.0 indicates that the model's predictions are no better than simply guessing the mean ($\bar{y}$). Negative R^2 values are also possible, indicating that the surrogate model performs worse than this simple baseline mean.

2.2 The Frequency Domain and the Fast Fourier Transform

While a time-domain representation tracks how a signal's amplitude changes over time, a frequency-domain representation decomposes that same signal into its individual spectral components [29]. This shift in perspective reveals which frequencies are present in the signal and at what strengths.

Mathematically, a discrete time-domain signal $x[n]$ of length N is mapped to the frequency domain using the Discrete Fourier Transform (DFT):

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i\frac{2\pi}{N}nk} \tag{2.4}$$

Evaluating this equation directly is computationally demanding, requiring $\mathcal{O}(N^2)$ arithmetic operations. The FFT is simply a highly efficient algorithm designed to compute the exact same DFT, reducing the computational complexity to $\mathcal{O}(N \log N)$. This logarithmic scaling makes high-throughput spectral analysis practical for modern data processing pipelines.

2.3 Gravitational Wave Source Types

The modular global-fit pipeline handles two main source populations [10]:

- **Galactic Binaries (GBs):** Millions of compact binaries in the Milky Way. Their signals are *quasi-monochromatic* (nearly constant frequency) and persist for the entire mission. They are analysed exclusively in the *frequency domain*, where each source appears as a narrow peak, each GB is described by *eight parameters* as seen in Table 4.1. Most GBs are unresolved and form a stochastic foreground.

- **Massive Black Hole Binaries (MBHBs):** Merging supermassive black hole systems. Their signals are *transient* (hours to weeks) and *broadband*. Detection uses frequency-domain methods, but parameter estimation is performed in the *time domain* using heterodyning. Massive Black Hole Binary (MBHB)s are the loudest sources and are subtracted early to avoid contaminating the analysis of fainter signals, each MBHB is described with *eleven parameters* [21].

2.4 LISA Data Challenges and the "Sangria" Dataset

To facilitate the development, benchmarking, and collaborative validation of data analysis pipelines, the LISA Consortium's LISA Data Challenge working group organizes standardized mock data challenges [3]. These challenges provide simulated time-series and frequency-domain datasets of increasing complexity, integrating realistic instrumental noise with overlapping signals from various astrophysical populations.

The second major iteration of these challenges, codenamed LISA Data Challenge "Sangria", is specifically designed to simulate a "mild source confusion" regime [3]. The Sangria dataset comprises Gaussian instrumental noise superimposed with simulated waveforms from over 30 million Galactic white dwarf binaries (GBs), verification binaries, and several loud merging MBHBs [3].

Because LISA Data Challenge Sangria represents the consensus baseline for evaluating the modular Global Fit pipeline, the data generation pipeline designed in this thesis (Section 4.3.2) is

intentionally configured to mimic the exact signal mixing, TDI specifications, and noise characteristics of the Sangria dataset.

2.5 Computational Infrastructure: Deucalion HPC

The computational experiments in this work are conducted on *Deucalion*, a petascale supercomputer hosted at the Minho Advanced Computing Center in Portugal. As part of the EuroHPC Joint Undertaking, Deucalion provides a theoretical peak performance of 10 PetaFLOPS through a unique heterogeneous architecture that integrates ARM, x86, and GPU-accelerated technologies.

2.5.1 Compute Node Architectures

Deucalion is organized into three distinct partitions designed to accommodate diverse computational requirements:

- **ARM Partition:** The primary component consists of 1,632 nodes equipped with Fujitsu A64FX 48-core processors. These nodes utilize 32 GB of High Bandwidth Memory (HBM2) and are optimized for energy-efficient, vector-parallel workloads, similar to the architecture used in the Fugaku supercomputer.
- **x86 Partition:** For general-purpose high-throughput computing, the system offers 500 nodes, each featuring dual AMD EPYC 7742 64-core processors (128 cores per node) and 256 GB of RAM, providing a robust environment for standard Central Processing Unit (CPU)-bound simulations.
- **Accelerated GPU Partition:** This partition comprises 33 nodes designed for Artificial Intelligence and massively parallel tasks. Each node is equipped with two AMD EPYC 7742 CPUs and four NVIDIA A100 Tensor Core GPUs (interconnected via NVLink), providing high-density floating-point performance and accelerated memory throughput.

2.5.2 Workload Management

Resource allocation and job execution are managed via the **Slurm** (Simple Linux Utility for Resource Management) workload manager. Slurm ensures efficient distribution of tasks across the heterogeneous partitions, allowing for precise control over CPU affinity, GPU allocation, and multi-node communication over the high-speed InfiniBand HDR network. This environment provides the scalability required for the intensive machine learning models and to run the Global Fit pipeline in this study.

Chapter 3

State of the Art

3.1 Literature Review

This literature review adopts a semi-systematic approach, balancing methodological rigor with the flexibility required for emerging, interdisciplinary fields like AI-driven code optimization for HPC.

Beginning with exploratory seed papers and broad queries, the search was progressively refined through structured database queries, specific eligibility criteria, and AI-assisted retrieval tools. This process successfully identified a diverse range of literature, spanning specialized optimizations from CNNs and gravitational wave detection to broader HPC code improvements.

3.1.1 Initial Research Methodology

The review process was iterative, combining supervised seed guidance, search engine queries, and reference-based expansion, it included both manual screening and AI-assisted querying (Gemini Deep Research), the queries were refined over time based on the findings in early-stage papers. The final set of papers reflects the diversity in model architecture, bias source, and methodological contribution.

Filters were applied later on to decrease the amount of results and to increase the quality of the results, publication date, and Web of Science.

3.1.2 Search Strategy

Initial seed papers were recommended by the dissertation supervisor and reviewed in depth after consulting the references of those papers. After those steps, 10 important papers were found.

General search queries were run on IEEE Xplore, web of science and ACM Digital Library sorted by Relevance.

3.1.3 Results of the queries with filters applied

Each database has its own filters, which can help greatly in reducing irrelevant results without losing the relevant results. The following filters were used in the search engines:

- **IEEE Xplore** There were not many results from this database, so we will not apply filters.
- **Web of Science** The search target was changed for Title, abstract, keyword plus, and author keywords. For the first query, a temporal filter was added to show articles from 2021 and newer for the first query, on the third query, a temporal filter of 2016 and newer was used.
- **ACM Digital Library** A temporal filter was added for the first query to show articles newer than 2021.

Table 3.1: Number of Results per query, without website filters

	IEEE Xplore	**ACM Digital Library**	**Web of Science**	**Total**
Number of Papers	4	30	98	132

3.1.3.1 Screening and Selection Criteria

Following the collection of 132 records from the databases, a two-stage screening process, modelled on the PRISMA guidelines, was applied to identify relevant, high-quality studies. By adding the 10 recommended papers and books and by removing 4 duplicates papers using EndNote's automated feature, 138 unique articles are left. Subsequent filters were then applied to further refine the dataset, as detailed below:

- **Title and Abstract Screening**: The titles of the remaining 138 articles were screened for relevance. This led to the exclusion of 72 articles, resulting in 66 for the next stage.
- **Full-Text Availability**: We then confirmed that the full text was accessible for all articles. As all were available, no papers were excluded at this stage.

The remaining parts of the selection, involve reviewing the initial image, abstract, introduction, along with the Content Assessment and Analysis and State of the Art chapters. The entire process that resulted in these final eligible articles can be seen in Figure 3.1.

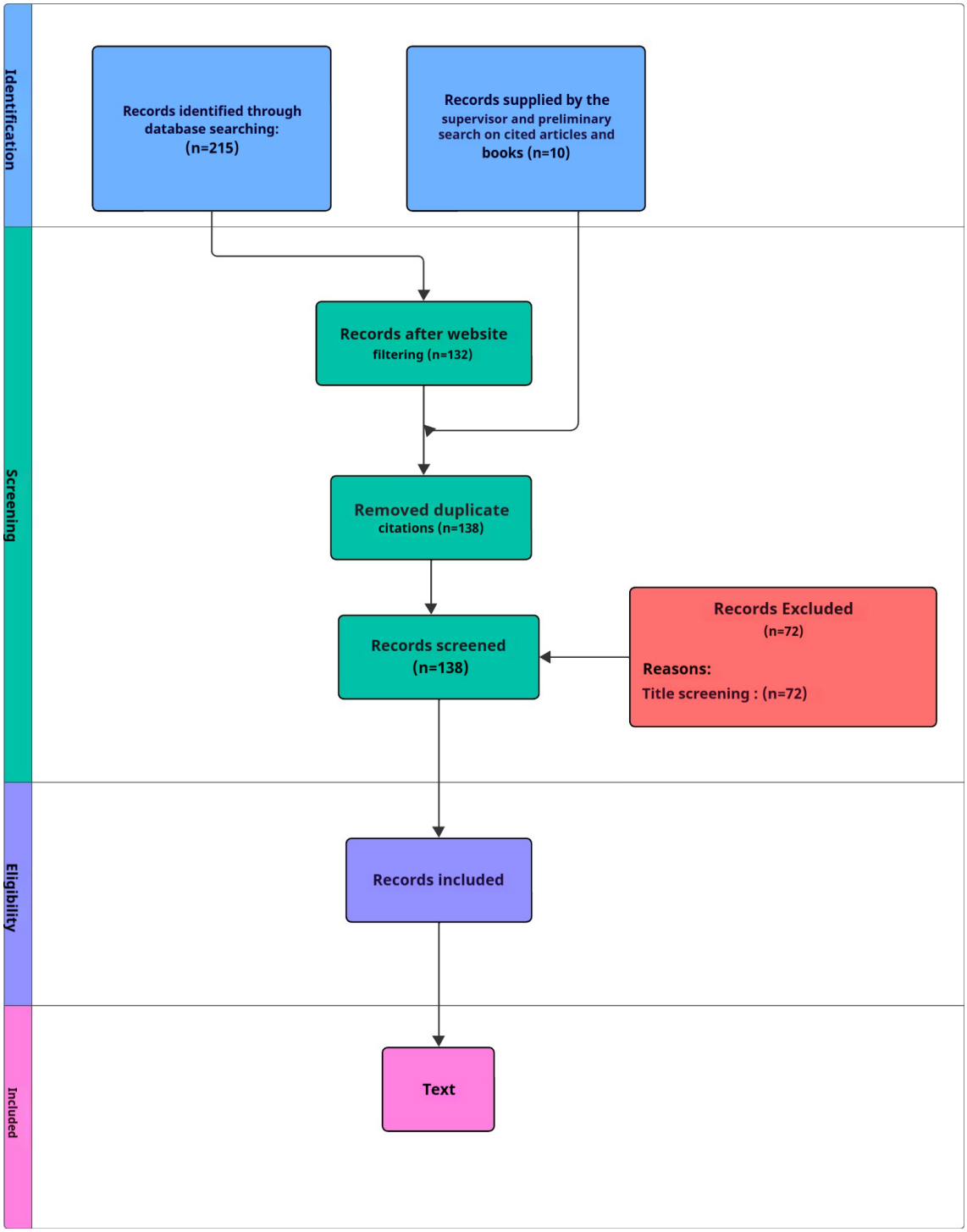


Figure 3.1: Flow chart showcasing the Literature review process

3.2 The Classical LISA Global Fit Framework (Analytical Baseline)

The data analysis requirements for the LISA mission present a unique computational challenge. Unlike ground-based detectors, the data stream is signal-dominated, containing thousands of overlapping signals from different astrophysical populations (e.g., GB, MBHB) and non-stationary instrument noise [22]. Consequently, the state of the art in data processing has shifted from isolated signal extraction to complex, hierarchical Bayesian inference pipelines that require significant HPC resources.

3.2.1 Global Fit Pipeline Architecture

One of the current architectural standards for analysing data is the Global Fit Wheel, implemented in pipelines such as *Erebor*. This architecture employs a Blocked Gibbs Sampling approach, where the global parameter space is partitioned into blocks corresponding to specific source types (e.g., GB, MBHB, Noise). These blocks update iteratively, exchanging data residuals to condition the inference of other blocks [10].

From a software engineering perspective, this modularity is critical for resource management. It allows different hardware backends to be targeted for different source types; for instance, GB blocks, which involve analysing thousands of sources simultaneously, benefit significantly from massive parallelization on GPUs [22].

The analysis of LISA data represents a signal-processing challenge of unprecedented scale. Unlike ground-based detectors, LISA's data stream is signal-dominated, containing tens of thousands of overlapping GW sources. The "Global Fit" is the process of simultaneously identifying, separating, and characterizing every resolvable source in the data. The current architectural standard for this task is a modular, iterative framework often implemented as the "Global Fit Wheel".

3.2.1.1 Iterative Strategy: The "Wheel of Fortune"

Because the total number of sources is unknown and signals are deeply covariant, the pipeline employs a cyclic, iterative strategy known as the *Wheel of Fortune* mechanism [22]. The functional workflow of this architecture is illustrated in Figure 3.2 that contain the main components:

1. **Kick-off Phase:** As seen at the top of Figure 3.2, the process begins by ingesting raw TDI data to perform preliminary detections. High-Signal-to-Noise Ratio (SNR) candidates, primarily MBHBs and bright Verification Binaries, are identified using maximum-log-likelihood searches. These preliminary reconstructions serve as the starting point for the Global Fit.

2. **Parallel Source Blocks:** Within the main **Iteration** cycle (pink box), the pipeline spawns dedicated modules for different source classes. The **MBHB Block** (green) and **GB Block** (blue) operate in parallel to perform Bayesian inference. These blocks do not act on the raw data directly but on *residuals*, effectively exchanging data so that each block sees the TDI stream minus the current best-fit templates from all other known sources.

3. **Live Catalogues:** Each source block maintains a **Live Catalogue** (yellow stacks). These are dynamic data structures that store the current best-fit parameters for every identified binary. As the MCMC samplers in the blocks progress, they continuously update these catalogues to refine the global solution.

4. **Noise Characterization and Feedback:** The residuals from the updated source catalogues are fed into the **Noise Block**. This module characterizes the **Noise Model**, accounting for both instrumental effects and the unresolvable Galactic foreground. As indicated by the circular arrow in Figure 3.2, this updated noise model is then fed back into the source blocks, allowing for a more accurate likelihood evaluation in the subsequent iteration.

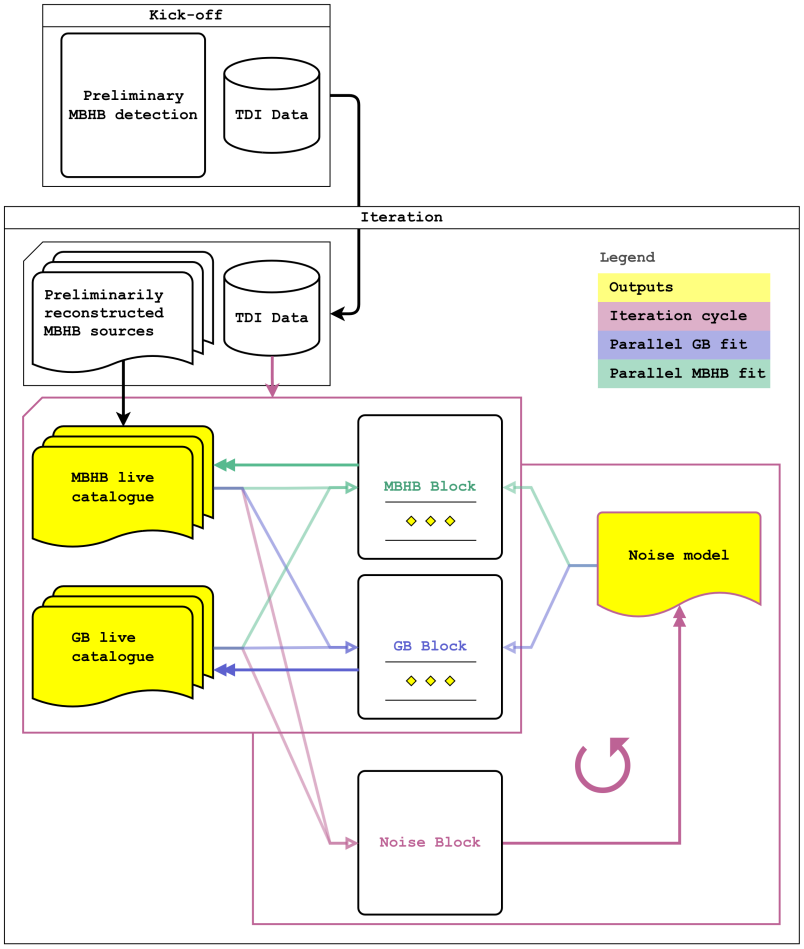


Figure 3.2: Architecture of the modular global-fit pipeline [10].

3.2.2 Bayesian Inference and Sampling Algorithms, MCMC Methods

The core computational engine of the Global Fit is MCMC sampling. A simple MCMC algorithm, such as the Metropolis-Hastings sampler, generates samples from a target probability distribution $p(\theta \mid d)$ by constructing a Markov chain.

Starting from an initial parameter vector θ_0, the algorithm proposes a new state θ^* from a proposal distribution $q(\theta^* \mid \theta_i)$. The proposed state is accepted with probability

$$\alpha = \min\left(1, \frac{p(\theta^* \mid d)\, q(\theta_i \mid \theta^*)}{p(\theta_i \mid d)\, q(\theta^* \mid \theta_i)}\right).$$

If accepted, the chain moves to θ^*; otherwise it remains at θ_i as is represented in Figure 3.3. Under mild conditions, this procedure guarantees that the chain's stationary distribution is the target posterior. In practice, an initial *burn-in* period is discarded to reduce sensitivity to the starting point, and the remaining samples are used to approximate posterior expectations. However, when the posterior contains multiple well-separated modes or strong parameter degeneracies, a single Metropolis-Hastings chain can become trapped, leading to extremely long autocorrelation times and unreliable estimates.

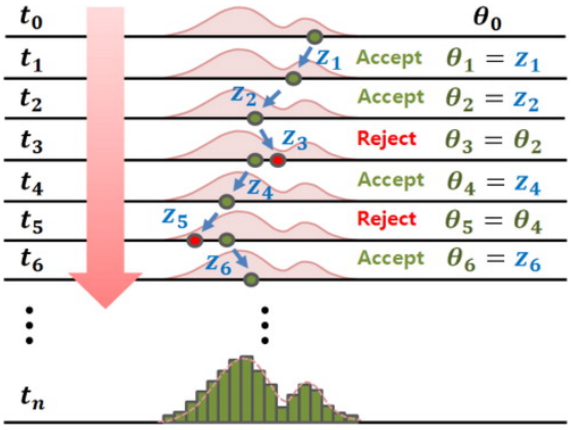


Figure 3.3: Illustration of a MCMC walker [18]

3.2.2.1 The Two phases of MCMC Burn-in and the Stationary phases

Distinguishing the difference between the transient **burn-in phase** and the **stationary phase** is crucial to ensure that the MCMC posterior distribution is unbiased, the MCMC chains are highly influenced by the starting point of the chains rather than the distribution, to mitigate the influence of the starting point on the result, the job of the *burn-in* is to take out the bias from the starting point of the chain [15]. The chain runs until the *burn-in* phase is complete, after which it discards the initial biased samples and begins building the posterior afresh.

3.2.2.2 Parallel Tempering

To prevent chains from becoming trapped in local optima, Parallel Tempering (PT) is employed. This involves running multiple chains at different *temperatures* to smooth the likelihood surface, allowing high-temperature chains to explore the global space while low-temperature chains refine local fits [1, 25]. However, standard PT can suffer from diffusive behaviour where chains linger at specific temperature levels. Recent advancements suggest the use of Non-Reversible Parallel

Tempering (NRPT), which enforces a deterministic flow of chains between temperatures, thereby increasing the *round trip rate* and improving convergence efficiency by orders of magnitude compared to reversible methods [39].

3.2.2.3 Ensemble Sampling

State-of-the-art pipelines, such as those utilizing the *Eryn* software package, utilize Ensemble MCMC methods [25]. By maintaining a population of "walkers" rather than a single chain, the sampler can leverage the internal structure of the ensemble to generate affine-invariant proposals. This approach is particularly advantageous for HPC implementations because the likelihood evaluations for the multiple walkers in the ensemble can be vectorized and executed in parallel on GPUs, significantly increasing computational throughput [20, 21].

3.2.2.4 Reversible Jump Markov Chain Monte Carlo

MCMC methods assume a fixed number of parameters. For LISA GBs, this would require knowing the exact number of resolvable sources a priori, which is not possible. Reversible Jump MCMC (RJ-MCMC) extends the classical algorithm to sample from parameter spaces of varying dimensionality [14].

In RJ-MCMC, the chain can propose jumps that increase or decrease the dimension of the state vector. For GBs, this corresponds to adding a new source (with its eight parameters) or removing an existing source. The acceptance ratio is modified to include a Jacobian term that accounts for the change in volume between spaces of different dimensions, ensuring detailed balance [14].

The sampler uses three types of proposals:

- **Update:** This is the standard Metropolis-Hastings move. The number of sources k stays the same; the chain proposes a new parameter vector for one existing source (e.g., perturbing its frequency or amplitude of the GBs). The dimensionality is unchanged.
- **Birth:** The chain proposes to increase k by one. It generates a new set of parameters for an additional source and appends them to the current state. The reverse move is a death. The acceptance ratio includes a Jacobian term and the probability of the reverse death move.
- **Death:** The chain proposes to decrease k by one. It selects one of the current sources at random and removes its parameters. This allows the sampler to discard sources.

The chain cycles through these moves, allowing it to automatically determine the most plausible number of sources while simultaneously estimating their parameters.

3.3 Deep Learning in Gravitational Waves

To address the limitations of traditional MCMC, deep learning techniques are increasingly integrated into gravitational wave analysis to serve as fast approximations or feature extractors.

3.3.1 Neural Surrogate Modeling

Applying Neural Surrogate Modeling involves replacing the expensive mathematical likelihood calculation with a neural network approximation during the initial *burn-in* or search phases [6, 25].

3.3.2 Normalizing Flows

A Normalizing Flow is a sequence of invertible neural layers that maps a simple base distribution (e.g., a Gaussian) to a complex target distribution, enabling efficient density estimation and sampling [24]. In the LISA Global Fit, flows can be used to accelerate Bayesian inference by providing better proposal distributions or approximate posteriors. Their application differs between GB and MBHBs only in the input representation: GBs operate on *frequency-domain* data (narrow spectral peaks), while MBHBs require *time-series* processing, often preceded by a 1D-CNN or autoencoder for dimensionality reduction [26]. The flow architecture itself remains generic, but the front-end feature extractor must be tailored to each data domain.

3.3.3 Preprocessing of the Frequency-Domain Data

When processing a discrete, real-valued time-series signal $x[n]$ of length N using the FFT, the output is a sequence of complex coefficients $X[k] = X_{\mathrm{re}}[k] + iX_{\mathrm{im}}[k]$. Rather than feeding the FFT Cartesian complex numbers directly into a neural network, we extract magnitude and phase features as described below.

3.3.3.1 The Srinivasan et al. Compression Scheme

To reduce the dimensionality of the LISA frequency series ($\sim 6 \times 10^6$ complex points), Srinivasan et al. [37] first retained the log-absolute value and the real part of each complex data point, and then subsequently rebinned the series by averaging over 10 consecutive frequencies, followed by subdivision into 1024 logarithmically spaced bins. Within each bin, they performed a linear fit of the values as a function of frequency, storing the slope and intercept. The standard deviation of the residuals captured the bin's dispersion, while the L_2 norm of outliers (residuals $> 3\sigma$) quantified the contribution of loud, resolvable sources. This yielded four features per bin per component ($\log_{10}|D|$ and $\Re(D)$), compressing the original data by a factor of $\sim 1.5 \times 10^3$ into an 8×1024 summary.

3.3.3.2 Magnitude and Phase Extraction

To facilitate stable neural network training, the Cartesian complex coefficients representing the FFT of the input data—are transformed into polar coordinates (magnitude and phase):

$$A[k] = \sqrt{X_{\mathrm{re}}[k]^2 + X_{\mathrm{im}}[k]^2}, \qquad \phi[k] = \operatorname{atan2}\big(X_{\mathrm{im}}[k], X_{\mathrm{re}}[k]\big).$$

While this coordinate transformation preserves the number of parameters per frequency bin (retaining two real values). This change of representation does not compress the data, but it significantly improves gradient flow and model convergence, as demonstrated in [45].

3.3.4 Compression of the Frequency-Domain Data

While Normalizing Flows provide a framework for bypassing expensive analytic likelihoods, they struggle when applied directly to high-dimensional raw data. The TDI channels (A, E, and T) used in LISA data analysis are time-series containing thousands of data points [26]. Training density estimators directly on such high-dimensional vectors is computationally infeasible and prone to overfitting. To mitigate the curse of dimensionality, feature extractors are employed to compress raw data into low-dimensional summary statistics.

3.3.4.1 1D-CNNs

Unlike the 2D CNNs typically used in computer vision for image classification, such as the architectures presented in [16] and [35], gravitational wave analysis requires the processing of sequential time-series or FFT data. Consequently, 1D-CNNs are used.

A 1D-CNN applies a set of learnable filters (kernels) that slide across the temporal dimension of the input strain data. This operation allows the network to detect local features such as merger peaks (MBHB), or ringdown oscillations (GB) regardless of their position in the time series [26].

However, standard 1D-CNNs face fundamental limitations when applied to LISA data analysis. First, their receptive field is inherently local, each convolutional layer only captures information within a fixed window size, requiring many stacked layers to model long-range temporal dependencies. For LISA's months-long observations containing thousands of overlapping GB signals, these long-range correlations are precisely what encode the frequency evolution ($\dot{f}$) and phase relationships that distinguish individual sources. Second, CNNs lack an explicit mechanism for learning which parts of the input are most relevant for a given prediction task. The same learned filters are applied uniformly across the entire time series. This brittleness is particularly problematic for LISA, where the final detector configuration, noise characteristics, and waveform models continue to evolve during the mission's development.

3.3.4.2 Autoencoder Architectures: From 1D-CNNs to GraviBERT

Within LISA Global Fit pipelines, CNNs are often used inside an autoencoder framework for non-linear dimensionality reduction. The encoder compresses the input (e.g. TDI channels) into a latent vector, and the decoder reconstructs the original signal. A denoising variant forces the network to learn noise-invariant features [26].

GraviBERT [4] improves upon plain 1D-CNN autoencoders by combining a multi-scale CNN feature extractor with a Transformer encoder and a BERT-style self-supervised pretraining phase. The CNN captures local patterns at different time scales, while the Transformer's self-attention

models long-range dependencies across the entire sequence. Pretraining on a masked reconstruction task learns general-purpose GW representations without labels.

GraviBERT is particularly well-suited for the LISA Global Fit pipeline. Its self-attention mechanism captures long-range temporal dependencies across months-long observations, which is essential for accurately estimating parameters such as the frequency derivative $\dot{f}$. The pretrained representations enable efficient transfer learning: models adapted to new detector noise profiles or waveform approximants require minimal fine-tuning, reducing retraining overhead as mission specifications evolve. Moreover, LISA will observe GWs across a wide frequency range; the self-supervised pretraining phase allows the model to learn general frequency-domain features from unlabeled data, significantly reducing the amount of labeled training data required for downstream inference tasks. The masked pretraining objective also helps separate coherent gravitational wave signals from the confusion noise that dominates LISA's data stream. Finally, GraviBERT operates directly on raw time-domain TDI channels, preserving phase information that frequency-domain methods often discard. Additionally, Monte Carlo Dropout, a technique that keeps dropout active during inference and performs multiple stochastic forward passes through the network, each with different randomly dropped neurons, can be applied to quantify predictive uncertainty, providing a measure of confidence [4]. With the given drawbacks of GraviBERT for LISA:

- **Computational cost of training:** While inference is fast, the self-supervised pretraining phase requires large amounts of unlabeled data and significant GPU resources (e.g., multiple days on high-end GPUs), which may be expensive or time-consuming before deployment.
- **Memory footprint:** Transformers have high memory requirements due to the self-attention mechanism, which scales quadratically with sequence length. LISA's long-duration time-series may require sequence truncation or efficient attention variants, potentially losing some long-range information.
- **Black-box nature and interpretability:** As a deep learning model, GraviBERT offers little insight into why it makes certain predictions. In a high-stakes scientific mission like LISA, this lack of interpretability may hinder validation and trust from the astrophysics community.
- **Risk of bias or systematic error:** If the training data does not fully cover the true parameter space or waveform variations, the model may introduce biases that propagate through the Global Fit, potentially affecting scientific conclusions.
- **Generalization to unseen source populations:** The model is trained on simulated waveforms with specific priors. If the true LISA data contains unexpected source types or noise artifacts, the pretrained model may fail gracefully, requiring fallback mechanisms.
- **Overhead of uncertainty calibration:** While MC Dropout (different from MCMC) provides uncertainty estimates, these need to be properly calibrated, otherwise, the model may be overconfident or underconfident, undermining hybrid surrogate-MCMC schemes.

Consequently, GraviBERT offers a transferable and computationally efficient foundation for accelerating the Global Fit pipeline, and its integration represents a promising direction for future work.

3.3.4.3 Denoising Autoencoders

A significant engineering challenge in GW detection is distinguishing weak signals from instrumental noise. Standard autoencoders may learn to reconstruct the noise profiles rather than the underlying physical signal. To address this, *Denoising Autoencoders* are employed [26].

In this approach, the network is trained on pairs of data where the input $\tilde{x}$ is a noisy waveform (Signal + Noise), but the reconstruction target is the clean, noise-free waveform. This forces the encoder E_ψ to learn robust, noise-invariant features of the gravitational wave signal, effectively acting as a learned non-linear filter [26].

3.4 Optimizing Bayesian Inference via Deep Learning

The primary computational bottleneck in Bayesian inference for gravitational wave detection is the MCMC *burn-in* phase. During this stage, the sampler performs a computationally expensive random walk through low-likelihood parameter spaces before converging on the high-likelihood mode. To mitigate this, contemporary research focuses on replacing physical likelihood evaluations with Machine Learning (ML) approximations and utilizing deep learning to provide informed initializations [8, 19].

3.4.1 Surrogate-Assisted Burn-in with MLPs

A direct engineering optimization involves the deployment of *Surrogate Models* to intercept calls to expensive physics-based waveform generators and use a machine learning model such as MLPs as lightweight proxies for the likelihood function $L(\theta)$.

Experimental results depicted in [25] using LISA-like datasets have demonstrated that using an MLP surrogate low-likelihood as a substitute of the full physics computation proposals can yield up to a $2.92\times$ speedup in total convergence time by short-circuiting the "Brute Force" search state.

In another architecture [6], the ML model is implemented and trained "online" using a manager-worker pattern. As parallel MCMC replicas explore the parameter space, the manager collects (θ, L) pairs to retrain a global surrogate model at fixed intervals. To ensure robustness against the sparse data typical of early-stage sampling, architectures often employ high dropout rates (e.g., 0.4) to prevent overfitting.

3.4.2 Search-to-Proposal via Dimensionality Reduction

Raw LISA data, which may exceed 14,000 dimensions for the MBHB, is unsuitable for direct MLP processing [26]. State-of-the-art pipelines utilize Denoising Autoencoders to compress this

data into a compact latent representation z. By coupling this latent vector with a Normalizing Flow, such as a Masked Autoregressive Flow (MAF), the system can generate a rough approximation of the posterior $p(\theta|z)$. Sampling from this neural posterior provides the MCMC walkers with starting coordinates already situated within the high-likelihood region, effectively reducing the required *burn-in* length to nearly zero [26].

3.4.3 Algorithmic Tuning and Reasoning via LLMs

While traditional ML models handle the numerical heavy lifting, Large Language Model (LLM)s are emerging as "Reasoning Agents" for algorithmic design and meta-parameter tuning. Recent frameworks, such as Evo-MCTS, utilize LLMs to act as mutation operators within an evolutionary search tree, automatically discovering and optimizing signal processing code [42].

In the context of MCMC optimization, LLMs can be employed to dynamically tune the *Temperature Ladder* in Parallel Tempering schemes. By analyzing swap acceptance rates between chains in real-time, an LLM-based agent can rewrite the temperature schedule or propose new proposal distribution widths, automating the heuristic tuning processes that previously required significant human expertise [42].

3.4.4 Sequential Neural Likelihood (SNL)

Sequential Neural Likelihood [26] is a method for learning a surrogate model of an intractable likelihood function $p(\mathbf{x} \mid \theta)$ when only simulations from the model are possible. Unlike a standard regression network that directly outputs $\log p(\mathbf{x}_o \mid \theta)$ for a single observed dataset $\mathbf{x}_o$, Sequential Neural Likelihood learns a *conditional density estimator* that can approximate the likelihood for any $\mathbf{x}$. This is achieved using a Conditional Normalizing Flow.

3.4.5 Amortized Population Inference via Simulation-Based Inference (SBI)

A recent and significant paradigm shift in gravitational wave data analysis is the move toward Simulation-Based Inference, also referred to as likelihood-free inference. While the traditional LISA Global Fit is designed for source-level characterization, it faces extreme scaling challenges when applied to population-level studies. Srinivasan et al. [37] proposed a method to bypass the Global Fit entirely, inferring astrophysical population parameters directly from the frequency-domain strain data. While completely bypassing the Global Fit using amortized population inference simplifies the data pipeline, it requires significantly larger training sets to resolve degeneracies.

3.4.5.1 Amortization and Computational Efficiency

The primary engineering advantage of the Simulation-Based Inference approach is its *amortized* nature. In traditional MCMC approaches, the computational cost is incurred during the inference

phase for every new dataset. In contrast, Simulation-Based Inference shifts the burden to a one-time training phase. Once the neural network (typically a Normalizing Flow) is trained on a large suite of simulations, the inference on real LISA data can be performed in seconds. This is particularly relevant for the operational requirements of the mission, where rapid updates to astrophysical models may be required as data accumulates. However, this architecture suffers from high training data requirements and substantial GPU computational overhead during the training phase.

3.4.5.2 HPC-Accelerated Forward Modeling

The scalability of Simulation-Based Inference is tied directly to the performance of the forward simulator. Srinivasan et al. [37] demonstrated a GPU-accelerated implementation using `CuPy` and `Numba` (JIT compilation), achieving a three-order-of-magnitude speedup over previous CPU-based population synthesis codes. For an engineer, this highlights a critical design pattern [9, 37]: by optimizing the forward pass to run in under a minute for millions of GB sources, it becomes feasible to generate the millions of realizations required to train deep density estimators. This aligns with the objectives of this thesis regarding the utilization of the Deucalion HPC environment for high-throughput astrophysical simulations.

3.5 Hardware Acceleration and Profiling Tools

Deploying machine learning models and data pipelines requires fast hardware and careful performance tracking. This section covers current methods for accelerating calculations using GPUs and managing data movement between processors. It also introduces the standard performance metrics and profiling tools used to find and fix speed bottlenecks in supercomputing environments.

3.5.1 Heterogeneous Architectures (CPU-GPU Scheduling)

The computational cost of waveform generation, particularly for the thousands of GBs, has necessitated a transition from CPU-centric to GPU-accelerated architectures.

3.5.1.1 GPU Kernel Optimization

Modern pipelines leverage GPUs to perform massive batches of likelihood calculations. For example, the *gbgpu* package allows for the parallel computation of GB waveforms, achieving significant speedups over CPU implementations [25]. However, maximizing GPU utilization requires minimizing non-computation overheads, such as kernel launching and memory transfers. Techniques such as “monolithic” kernel fusion have been proposed to merge multiple operations (e.g., neural network layers or mathematical operations) into single kernels, reducing context switching and optimizing on-chip resource usage [46]. Furthermore, automated tuning of kernel parameters (e.g., block sizes and tiling) via algorithms like dual annealing is essential to adapt code to specific hardware architectures (e.g., NVIDIA A100 vs. consumer cards) [33].

3.5.1.2 Memory and Scheduling

Efficiently managing data movement between host CPU and device GPU is critical. Architectures such as *FusionFlow* demonstrate that decoupling data preprocessing from model training and utilizing idle CPU cycles for data preparation can prevent GPU starvation [23]. In the context of LISA, memory management strategies involve storing pre-computed waveform components in GPU buffers to avoid redundant calculations, necessitating careful management of Video Random Access Memory (VRAM) to prevent overflows during transdimensional (Reversible Jump) updates [22].

3.5.2 Benchmarking Metrics and Profiling Tools

To rigorously evaluate the computational efficiency of the pipeline, it is essential to define standardized performance metrics and utilize specialized profiling tools that can identify bottlenecks across heterogeneous (CPU and GPU) hardware.

3.5.2.1 Performance Metrics

The evaluation of HPC and deep learning applications relies on several key metrics to quantify efficiency and scalability:

- **Speedup and Scalability:** Speedup measures the ratio of execution time between a baseline (often a single CPU core or serial implementation) and the accelerated parallel implementation [17]. Strong scaling evaluates performance as the number of processors increases while the total problem size remains fixed, whereas weak scaling measures performance as the problem size increases proportionally with the number of processors [41].
- **Throughput and Latency:** For inference and data processing tasks, throughput (measured in samples per second or TFLOPS) indicates the rate at which data is processed [2]. Latency measures the time required to process a single unit of data [43], which is critical for real-time applications. Tail latency (e.g., P90, P95) is often analyzed to understand performance variability under load [46].
- **Hardware Utilization:** This metric, often expressed as a percentage, indicates how effectively the compute resources (e.g., CUDA cores or Tensor Cores) are occupied. Low utilization (e.g., $< 50\%$) often signals bottlenecks in memory bandwidth or data loading rather than compute capacity [2]. Occupancy specifically refers to the ratio of active warps on a GPU relative to the maximum number of warps supported, serving as a key indicator of latency hiding capability [40].

3.5.2.2 Profiling and Tuning Tools

Identifying specific computational bottlenecks requires advanced profiling tools that can visualize the execution timeline and memory hierarchy usage.

- **Intel Profiling Tools:** For CPU-bound operations, Intel Advisor is a primary tool used to perform Roofline analysis, which visualizes the trade-off between floating-point performance and memory bandwidth. It helps determine if a kernel is compute-bound or memory-bound on CPU architectures. Additionally, Intel Parallel Studio provides a suite of tools for analyzing threading design and vectorization efficiency on Intel Xeon processors [38].
- **NVIDIA Nsight Systems (nsys):** This is a system-wide profiling tool used to visualize the timeline of CPU-GPU interactions. It is essential for identifying non-computation overheads, such as kernel launch latencies, API calls, and memory transfers (H2D/D2H) that cause GPU idle time. It is also used to capture peak memory footprints during execution [2, 44].
- **NVIDIA Nsight Compute (ncu):** For granular kernel analysis, ncu provides detailed metrics on GPU instruction execution. It is used to generate Roofline models for GPU kernels, calculating arithmetic intensity (Floating-Point Operations/byte) to pinpoint whether specific CUDA kernels are limited by Dynamic Random Access Memory (DRAM) bandwidth or compute capability.
- **TensorBoard:** TensorBoard is used to monitor training dynamics, visualize computational graphs, and track loss metrics over time, often in conjunction with TensorFlow or PyTorch profilers Sim, Shin, and Lee [34].

3.6 Research Gaps and Thesis Focus

A critical gap in existing literature is the lack of research that trains a log-likelihood predicting surrogate model and successfully integrates it directly within an active, operational LISA Global Fit pipeline (detailed in Section 3.2).

To address this, our approach trains the surrogate without detector noise (as detailed in the simplified setup in Section 4.2.3), focusing instead on various overlapping background signal configurations (discussed in Section 5.2). This prevents the model from memorizing specific noise patterns, forcing it to learn the physical parameters of the target waveform (as designed in Section 4.5). This thesis shows that even though the model is trained on noise-free data, it can still successfully predict likelihoods when integrated into the real pipeline where noise is present.

Chapter 4

Methodology

This chapter explains the methodology for the solution to the limitations identified in the State of the Art regarding the computational efficiency of the LISA Global Fit. The goal is to move away from computationally expensive likelihood evaluations by implementing specific Machine Learning accelerators.

Crucially, while the Deucalion HPC infrastructure will be used as the development testbed, this methodology prioritizes *algorithmic portability* over hardware-specific micro-optimization. The focus is on identifying architectural bottlenecks and resolving them with ML surrogates that can be deployed across various supercomputing clusters, rather than tuning kernels specifically for the A64FX or A100 architectures.

4.1 Research Methodology

This research follows the Design Science Research Methodology (DSRM) proposed by Peffers et al. [31], which provides a process model for conducting design science research. The methodology consists of six activities that are represented in Figure A.1, which are mapped to this thesis:

- **1. Problem identification and motivation:** As outlined in Chapter 3, prototype implementations of the LISA Global Fit pipeline present significant computational challenges due to the scale of the required sampling, which involves mathematically expensive, physics-based likelihood evaluations that must be computed throughout the MCMC process to resolve thousands of overlapping signals. This high computational overhead limits the pipeline's efficiency in processing massive data streams and increases the resource demands on high-performance computing infrastructures.

- **2. Define Objectives for a Solution:** Our objective is to design a portable ML accelerator capable of numerically approximating the GB likelihood function. The target objectives defined in Section 4.2.3 require that the ML-based surrogate model achieves a convergence speedup of $\geq 2\times$ during the initial MCMC search phase while preserving astrophysical parameter estimation accuracy comparable to the original jaxified physical pipeline.

- **3. Design and Development:** This phase focuses on creating the design artifacts. We develop a dual-branch neural network architecture (detailed in Section 4.5) that maps an 8-dimensional physical parameter space $\theta \in [0,1]^8$ and a $1,556$-dimensional pre-processed TDI data vector $\mathbf{D}$ to predict log-likelihood values $\log \hat{\mathscr{L}}(\theta \mid \mathbf{D})$.
- **4. Demonstration:** We demonstrate the neural surrogate by wrapping the PyTorch model inside a wrapper (Section 5.6) and integrating it into the production-level pipeline. End-to-end sampler functionality is demonstrated using a simulated GB source within the narrow-band frequency spectrum (Section 5.7).
- **5. Evaluation:** The surrogate is evaluated quantitatively across both physical and computational dimensions in Chapter 5. We assess test-set accuracy across different dataset diversity regimes (Section 5.2), evaluate model accuracy above critical sampling thresholds (Section 5.5), and compare estimated parameter posteriors to the physical baseline (Section 5.7.1). Lastly, pipeline execution speeds and integration bottlenecks are profiled in Section 5.7.2.
- **6. Communication:** The design, performance characteristics, and unresolved integration constraints of this hybrid deep learning architecture are communicated and documented throughout this Master's thesis (Chapter 6).

The entry point for this DSR project is problem-centered, as the research originated from the observation of a concrete performance bottleneck in an existing system, rather than from a pre-existing solution seeking a problem or a client-initiated request.

Following vom Brocke et al. [5], evaluation is conducted concurrently throughout the process: problem validity is assessed in Phase 1, design feasibility in Phase 2, artifact correctness in Phase 3, and solution effectiveness in Phase 4. This formative evaluation approach allows for iterative refinement and risk mitigation.

4.2 Galactic Binary (GB) Block

Only a small fraction (approximately 10,000) will be individually resolvable; the rest form a stochastic foreground that must be jointly modelled with the resolvable signals [37]. The data analysis challenge is therefore not to detect a single source in isolation, but to perform a *Global Fit* that simultaneously identifies, separates, and characterizes the parameters of all resolvable GBs while accounting for the unresolved background and other source classes.

4.2.1 GB Signal Model and Parameter Space

For the purpose of LISA data processing, a GB signal is described by a waveform model that depends on eight parameters (see Table 4.1) [37].

Table 4.1: The eight parameters describing a GB waveform.

Parameter	**Symbol**
Amplitude	A
Inclination	ι
Initial Phase	ϕ_0
Polarisation Angle	ψ
Frequency	f
Frequency Derivative	$\dot{f}$
Ecliptic Longitude	λ
Ecliptic Latitude	β

4.2.2 Key Challenges for the GB Block

From a software engineering perspective, the GB block imposes several constraints that make direct, full-scale global fitting difficult:

- **High dimensionality and source overlap** Even a single GB has an 8-dimensional parameter space. When 10–20 resolvable sources occupy the same frequency band and signal overlap creates strong parameter degeneracies.
- **Unknown number of sources** The pipeline must simultaneously infer how many resolvable GBs are present (model selection) and their parameters, typically using trans-dimensional MCMC.
- **Computational cost of likelihood evaluation** Evaluating the likelihood for a single candidate GB requires generating its waveform and computing inner products over the entire time-series (millions of samples). This operation is called millions of times during MCMC.
- **Prototype codebase** The current GB analysis block is still a research prototype and the interfaces are evolving, documentation is incomplete, and fault tolerance is not yet implemented making optimisation and integration challenging.

Figure 4.1 illustrates a key aspect of these challenges: even with only six injected GB signals, the observed time-series (black) contains a superposition of their waveforms plus noise. The true combined magnitude of the GBs alone is overlaid in other colours, showing how strongly the signals overlap and how easily they are masked by noise. This makes disentangling individual sources and recovering their parameters difficult in practice.

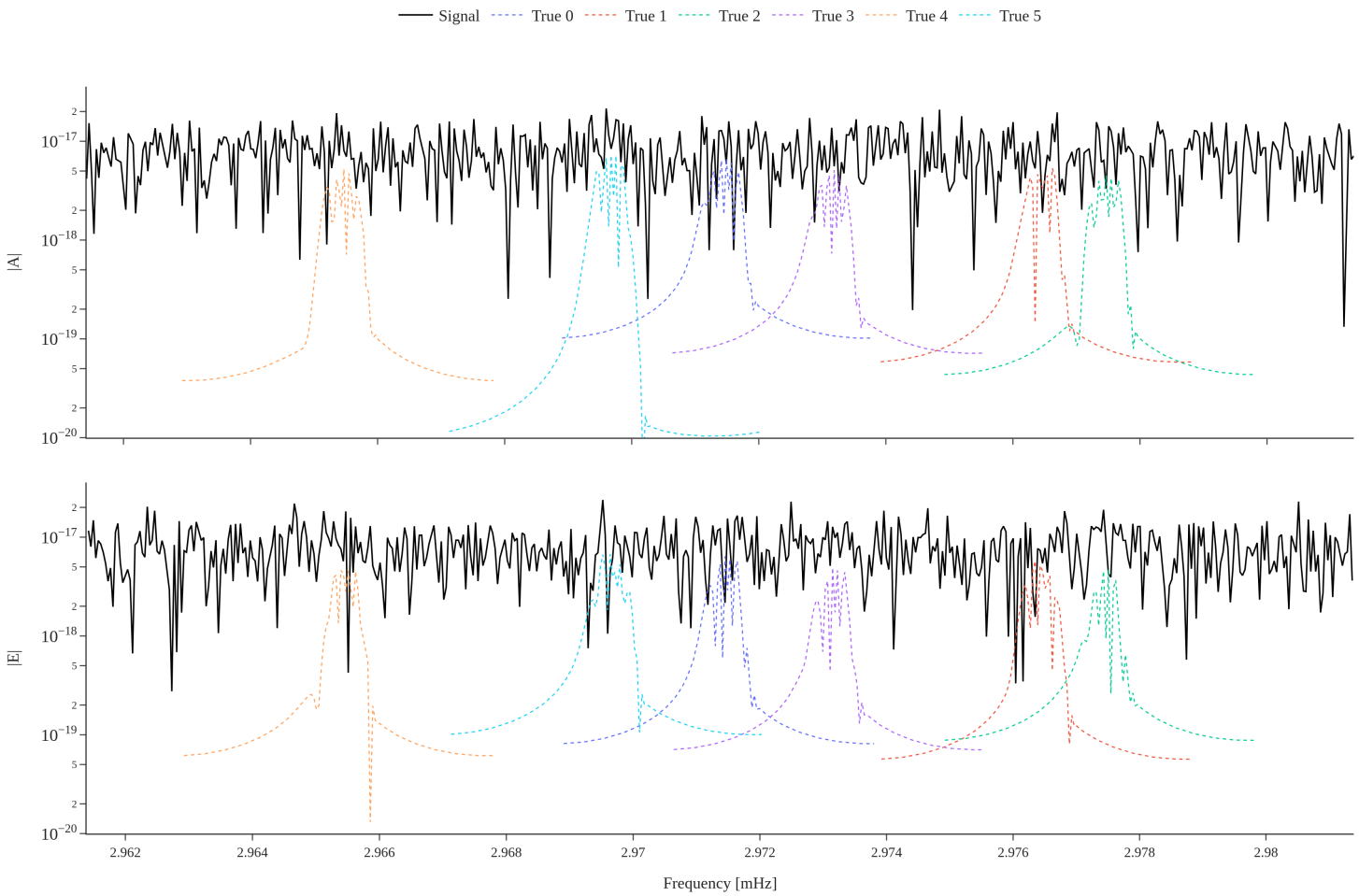


Figure 4.1: Magnitude of the synthetic time-series (black) with six injected GB signals. The true combined magnitude of the GBs alone is overlaid with other colours. Strong overlap and the presence of noise make it difficult to identify individual sources or recover their parameters.

4.2.3 Scope and Simplification: The Single-Source Narrow-Band Sub-Problem

Solving the full LISA GB parameter estimation challenge which requires simultaneously identifying and fitting tens of thousands of deeply covariant, overlapping sources across the entire sensitive frequency spectrum remains a major open problem in gravitational wave astronomy. Designing a single machine learning architecture capable of resolving this global, trans-dimensional problem directly from raw data is computationally and conceptually intractable.

To make the problem amenable to neural surrogate acceleration, this thesis focuses on a highly targeted, localized sub-problem. Rather than attempting a full-band, multi-source Global Fit, we isolate a narrow frequency window between 2.95 mHz and 3.0 mHz. Within this band, we simplify the inference task to a *single-source parameter* estimation problem: the surrogate is trained to predict the log-likelihood of a single target GB, evaluated against a synthetic data stream that has a small, overlapping background of 5 to 10 additional injected GBs to simulate a realistic confusion foreground. This localized, single-source focus is highly aligned with the modular architecture of

state-of-the-art LISA pipelines like *Erebor* [37]. In these pipelines, the Global Fit is decomposed into a Blocked Gibbs Sampling scheme, where individual sources or small clusters are updated iteratively while keeping the residuals of all other source types fixed.

4.3 Data Generation for the Surrogate Model

The surrogate model is trained to predict the log-likelihood of a single GB from a short, fixed frequency band of LISA data. The data generation pipeline follows an amortized approach: we pre-compute a large set of parameter data log-likelihood triplets and reuse them across multiple training runs.

4.3.1 Parameter sampling.

Each training example corresponds to a single GB with eight parameters, drawn from prior ranges summarized in Table 4.2. Unlike a full-band simulation, we restrict the frequency to a narrow band between 2.95mHz and 3.0mHz. This ensures that at most one resolvable binary falls inside the band, which simplifies the surrogate task.

Table 4.2: Prior ranges for GB parameters used in data generation. The frequency is constrained to a narrow band (2.95–3.00mHz).

Parameter	**Symbol**	**Sampling distribution**
Frequency	f	Uniform in $[2.95 \times 10^{-3},\ 3.00 \times 10^{-3}]$ Hz
Frequency derivative	$\dot{f}$	Uniform in $[10^{-16},\ 10^{-14}]$ Hz/s
Amplitude	A	Log-uniform in $[10^{-23},\ 10^{-21}]$
Ecliptic Latitude	β	$\arcsin(\text{Uniform}(-1,1))$
Ecliptic Longitude	λ	Uniform in $[0,\ 2\pi]$
Inclination	ι	$\arccos(\text{Uniform}(-1,1))$
Polarisation	ψ	Uniform in $[0,\ \pi]$
Initial phase	ϕ_0	Uniform in $[0,\ 2\pi]$

After sampling, the parameters are converted to *Hypercube coordinates* $\in [0,1]^8$. This transformation is required because the MCMC chain used in the Global Fit expects all parameters to be defined on a unit hypercube. Converting to hypercube coordinates also acts as a preprocessing step that balances the dynamic range of the parameters, improving the stability and convergence of the neural network models that are trained on these values.

4.3.2 Signal and log-likelihood sampling

For each data realisation, we first generate a synthetic LISA observation over 90 days with a 5-second cadence. To mimic the confusion foreground present in a realistic LISA data stream, we inject between 5 and 10 additional Galactic Binaries (with the parameters sampled from Table

4.2) and then the function `make_sangria_like_tdi()` is used to generate the TDI, to prevent the machine learning model from overfitting to specific noise realizations, no noise is added to the signal, only the 5 to 10 GBs.

The time-domain data are transformed to the frequency domain via an FFT function already included in the Global Fit prototype used, and the log-likelihood of the target binary is computed using a baseline likelihood function that operates on the A and E TDI channels. This implementation leverages automatic differentiation and just-in-time compilation, allowing efficient batch evaluation on GPUs. The output log-likelihood values range from approximately $-80\,000$ to $-1\,000$ (see Figure 4.2). The key statistical properties of this target log-likelihood distribution are summarized in Table 4.3.

Statistic	**Value**
Minimum (y_{min})	$-79{,}712.00$
Maximum (y_{max})	$-1{,}484.30$
Mean (μ)	$-9{,}342.54$
Standard Deviation (σ)	7,538.05

Table 4.3: Statistical distribution of the target log-likelihood values within the generated training dataset.

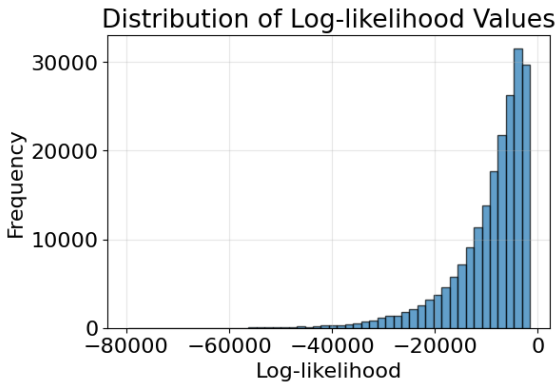


Figure 4.2: Histogram with the values of the log-likelihood for a training dataset with 200,000 samples and 15 samples per signal (13,334 signals)

4.3.3 Dataset Generation

The generation of the dataset is controlled by two parameters: `num_samples` (total number of training examples) and `samples_per_signal` (the number of different target parameter vectors that are evaluated against the signal generated). For each generated signal (i.e., each fixed noise and background configuration), we repeatedly sample new target parameters and compute their log-likelihoods. Multiple training batches were made with `samples_per_signal` values of 15, 50, 100, and 350. The resulting $(\boldsymbol{\theta}, \mathbf{D}, \log \mathscr{L})$ triplets are stored as compressed `.npz` files,

enabling fast reloading for later training runs. The generated data comprises three independent subsets: a training set of 200,000 samples, a validation set of 24,000 samples, and a testing set of 24,000 samples ($\approx$ 80% for training, $\approx$ 10% for testing, $\approx$ 10% for validation). Figure 4.3 illustrates the complete generation pipeline, from parameter sampling to cached feature vectors.

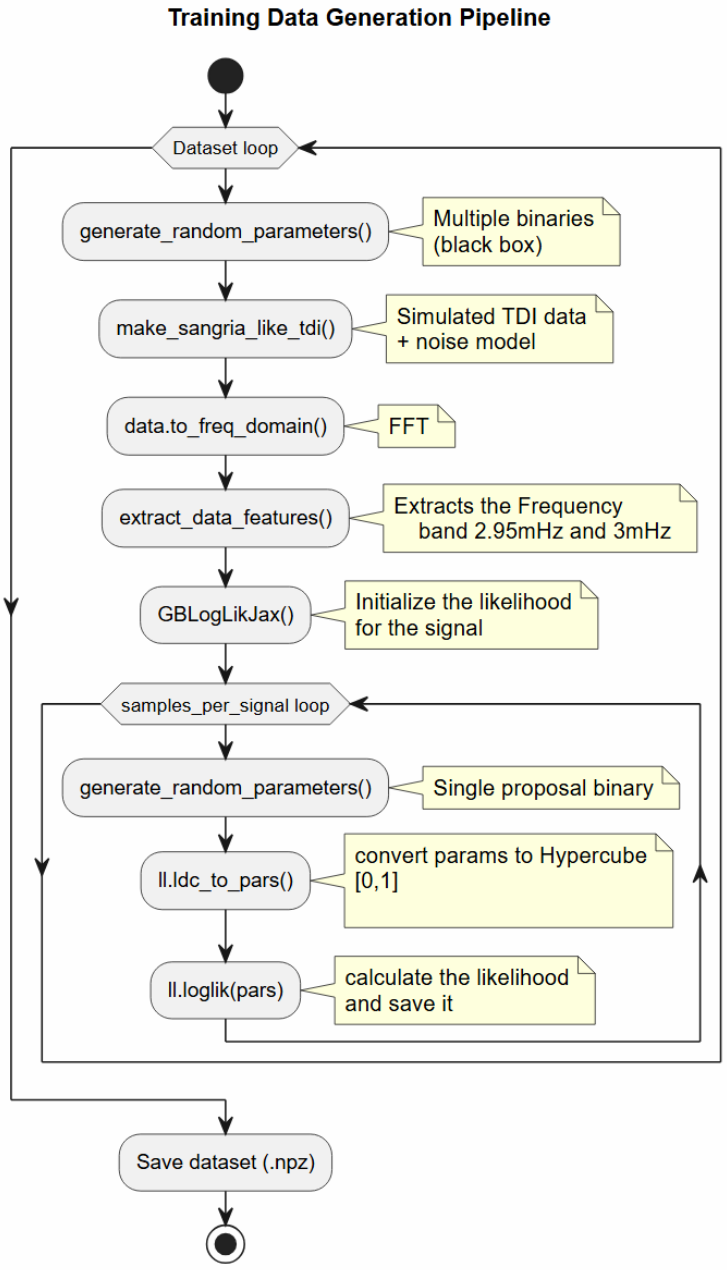


Figure 4.3: Data Generation Pipeline in detail.

4.3.4 Deucalion Data Generation

The data generation pipeline was executed on the Deucalion supercomputer (see Section 2.5) using its x86 partition. Multiple adapted scripts were submitted as independent Slurm jobs, each running

on a single node. Although only one node was used per job, the script took full advantage of the many available cores and the plentiful Random Access Memory (RAM) on the HPC node.

A single generation script required $\approx$ 35 hours to complete for each dataset a runtime that would be impossible on a local workstation due to the large memory footprint (the dataset generation requires holding multiple TDI realisations and log-likelihood evaluations in RAM). In total, three datasets were generated, with the following configurations: 200,000 with 5, 15 and 100 samples per signal, each accompanied by the corresponding validation datasets. The resulting triplets were cached as compressed files on the shared file system for subsequent training. The model training itself was not performed on Deucalion because its training was relatively fast, the computationally expensive and resource-intensive part was the dataset creation.

4.4 Feature Extraction and Preprocessing

The raw frequency-domain TDI data are high-dimensional: each channel contains approximately 777,601 complex points for a 90-day observation at 5s cadence. To reduce this dimensionality, we apply band selection:

1. **Band selection:** We extract only the frequency band from 2.95mHz to 3.0mHz. This yields 1,556 real numbers, formed by concatenating the real and imaginary parts of the A_{TDI} and E_{TDI} channels (389 frequency bins $\times$ 4 components). Figure 4.4 shows an example of the raw (real/imag) representation.

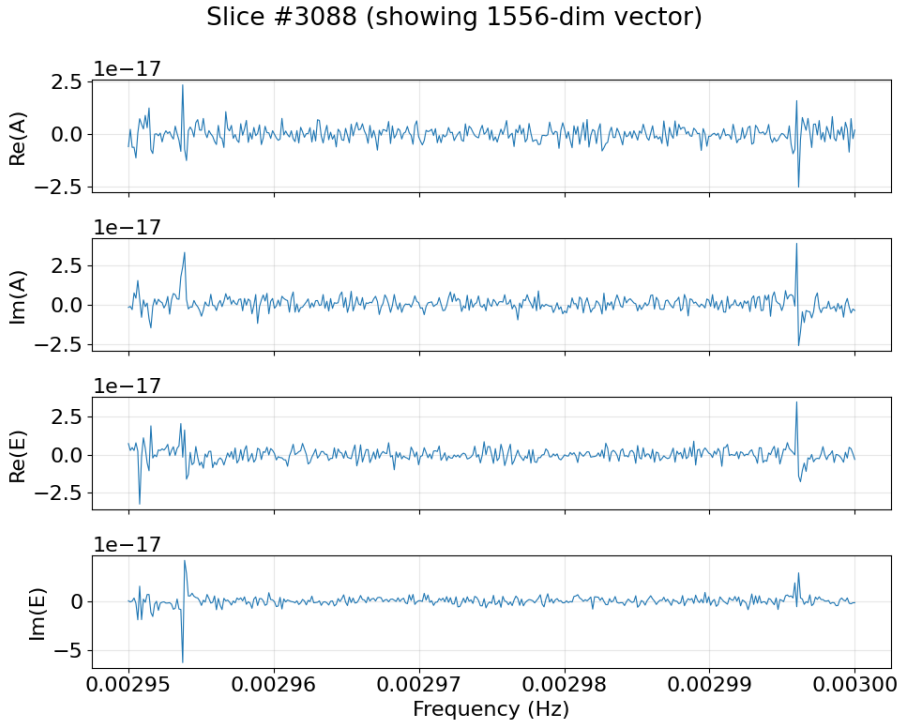


Figure 4.4: Raw FFT of simulated LISA data containing only GBs.

2. **Magnitude and phase conversion:** The function `convert_raw_to_mag_features` transforms the complex data into magnitude and phase (as previously discussed in Section 3.3.3.2). For each channel, it computes:

$$\text{amp} = |A_{\text{chan}}|, \qquad \text{phase} = \text{angle}(A_{\text{chan}}),$$

and similarly for the E channel. This conversion does not change the dimensionality (still 1,556 numbers), but it compresses the dynamic range of the signal (which spans several orders of magnitude) and isolates the phase information. The resulting feature vector **D** contains $[|A|,\ \angle A,\ |E|,\ \angle E]$. Figure 4.5 shows the same data after this conversion.

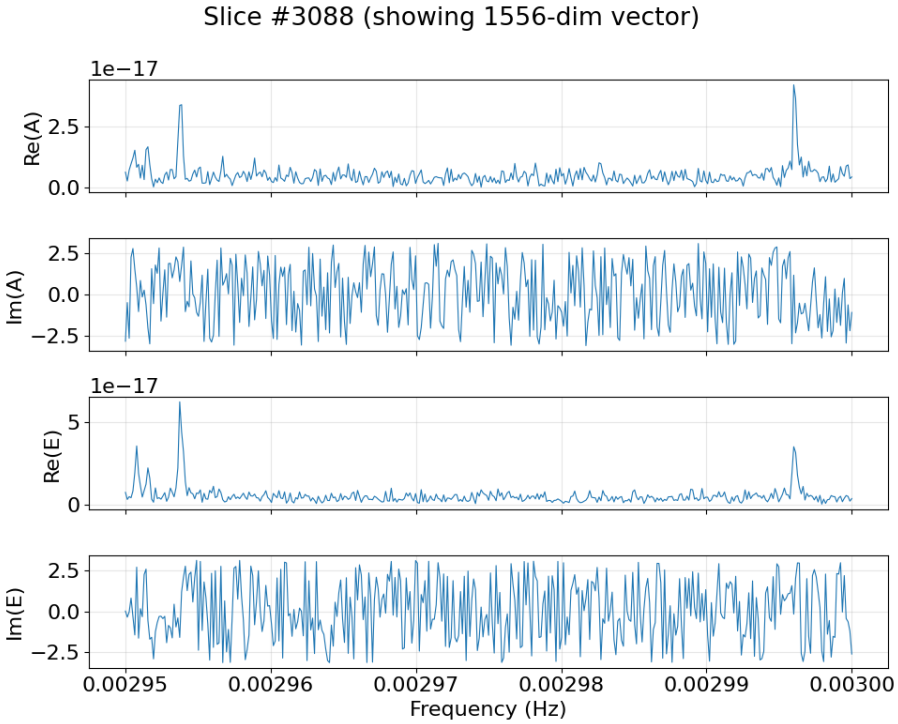


Figure 4.5: Magnitude and phase components extracted from the raw FFT (A and E channels).

After conversion, the magnitude components are standardized (zero mean, unit variance) using `StandardScaler` from `sklearn`, fitted only on the training set. The phase components are left unscaled because they already lie in $[-\pi, \pi]$. Figure 4.6 illustrates the effect of this standardisation. This step is crucial for stable neural network training, as it brings all input features onto a similar scale.

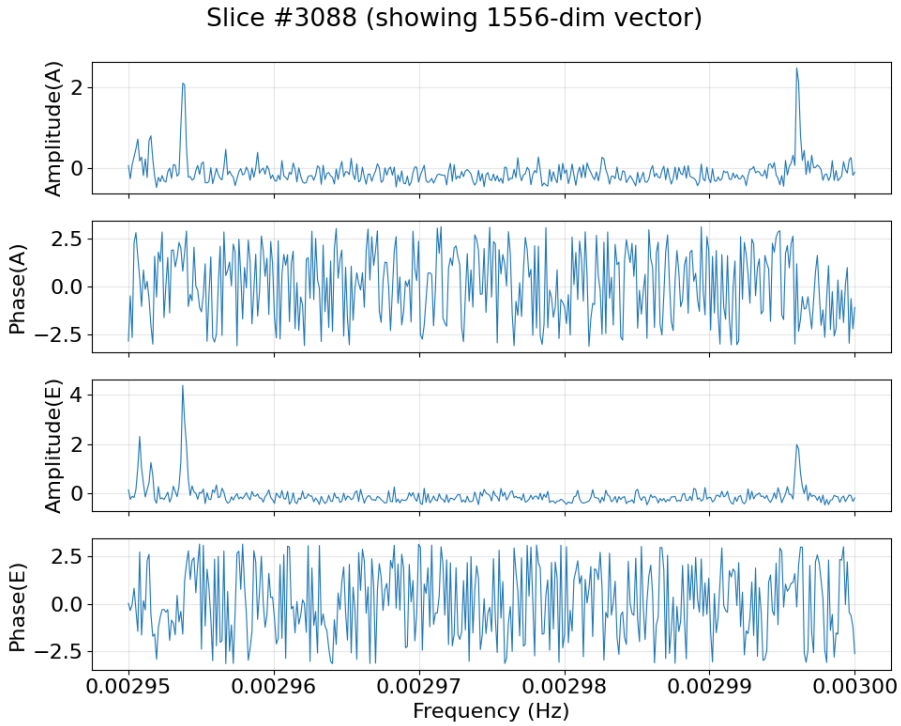


Figure 4.6: Standardized magnitude and original phase features after applying `StandardScaler`.

4.5 Machine Learning Model

The core of our computational acceleration strategy is a neural surrogate model designed to predict the log-likelihood of a single GB given its physical parameters and the pre-processed TDI data segment. The model is a feed-forward regression network with a *two-branch architecture*, as illustrated in Figure 4.7.

The dual-branch design is chosen to handle the multi-modal nature of the inputs: the physical parameters θ occupy a bounded, low-dimensional continuous space, whereas the pre-processed data segment $\mathbf{D}$ is a high-dimensional representation of the instrument's output. Processing these inputs through distinct pathways allows the network to extract independent latent features before fusing them for the final regression task.

GB Parameters Branch (θ) This branch ingests the 8-dimensional physical parameter vector $\theta \in [0,1]^8$, represented in unit hypercube coordinates. The vector is mapped through a series of three fully connected layers containing 256, 512, and 256 hidden units, respectively. Each linear transformation is followed by batch normalisation and a Rectified Linear Unit (ReLU) activation. The primary objective of this pathway is to project the low-dimensional parameter coordinate space into a rich, 256-dimensional latent representation $\mathbf{h}_\theta \in \mathbf{R}^{256}$ that captures non-linear parameter dependencies.

Signal Encoder Branch (D) This pathway processes the high-dimensional instrument representation $\mathbf{D} \in \mathbf{R}^{1556}$, containing the concatenated magnitude and phase features of the A and E channels. Given the dimensionality of this input relative to the parameter vector, a three-layer bottleneck architecture with 256, 128, and 64 hidden units is employed. This sequential compression forces the network to isolate the most salient physical features of the signal, outputting a compact latent feature vector $\mathbf{h}_D \in \mathbf{R}^{64}$.

Fusion and Regression Trunk To fuse the extracted representations, the latent outputs of both branches are concatenated to form a joint latent vector:

$$\mathbf{h}_{\text{concat}} = [\mathbf{h}_\theta \,||\, \mathbf{h}_D] \in \mathbf{R}^{320} \tag{4.1}$$

This 320-dimensional joint vector is then passed directly into the shared regression trunk for the final mapping.

The trunk consists of three dense layers with dimensions $320 \to 256 \to 256 \to 128$. To ensure stable training and mitigate overfitting, each layer incorporates batch normalisation, a dropout rate of 0.4, and ReLU activations.

To facilitate gradient flow through the deep regressor, a linear skip connection is introduced. Because the dimension of the concatenated vector (320) differs from the output of the third trunk layer (128), a learnable linear projection matrix $\mathbf{W}_{\text{proj}} \in \mathbf{R}^{128\times320}$ is applied to the shortcut connection:

$$\mathbf{h}_{\text{out}} = \text{ReLU}\left(\mathbf{h}_{\text{trunk3}} + \mathbf{W}_{\text{proj}}\mathbf{h}_{\text{concat}}\right) \tag{4.2}$$

This residual pathway allows the network to retain direct access to the initial fused features, mitigating vanishing gradients during backpropagation and allowing the model to learn identity mappings where additional depth is redundant [30].

The final linear layer maps the 128-dimensional representation down to a single continuous scalar, outputting the predicted log-likelihood value $\log \hat{\mathscr{L}}(\theta \mid \mathbf{D})$. This balanced distribution of capacity between feature extraction and joint regression allows the surrogate to map the complex, highly non-linear log-likelihood topology of overlapping GBs with high fidelity.

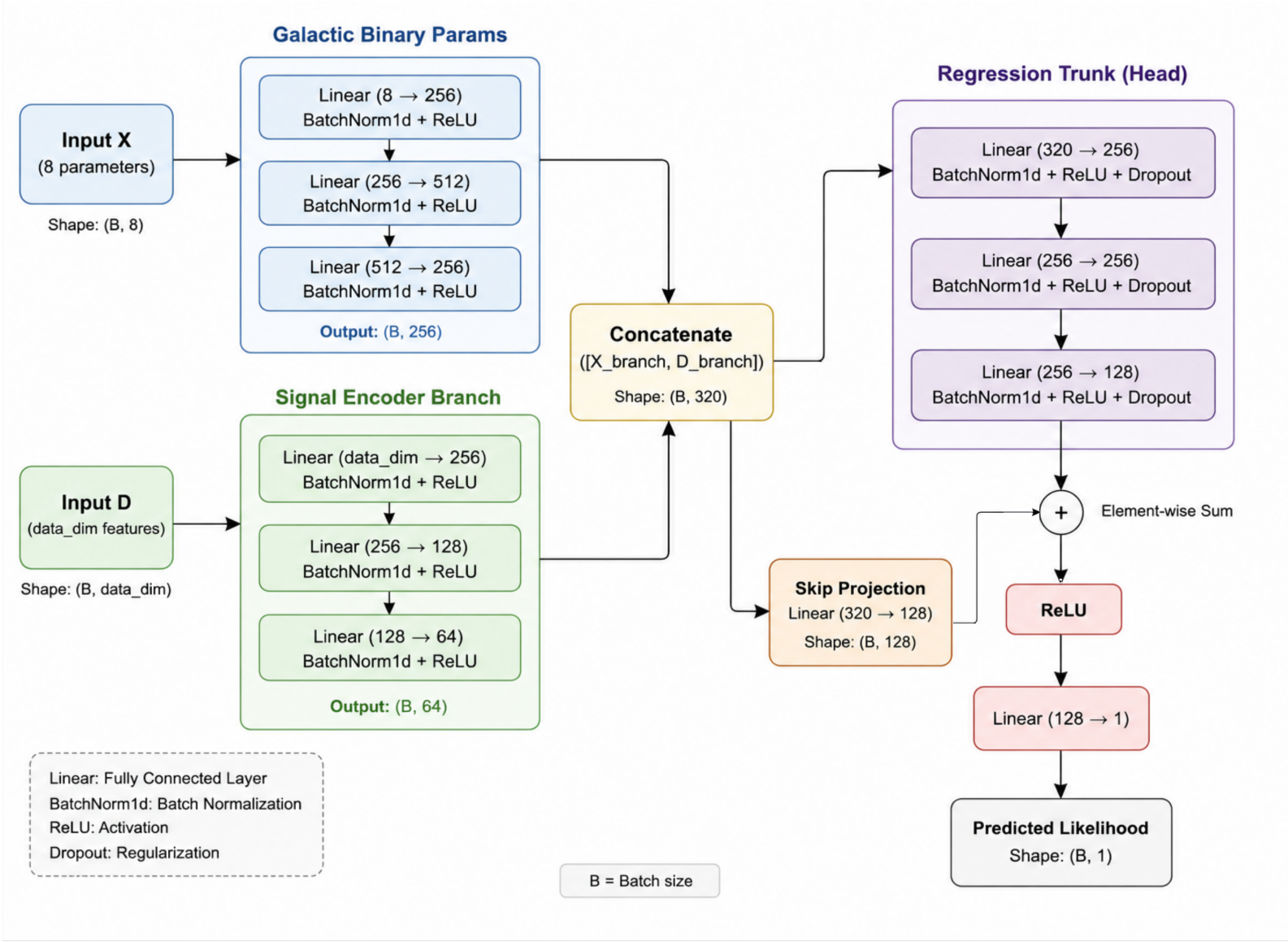


Figure 4.7: Architecture of the Machine learning model used to predict the log-likelihood the `data_dim` is 1556 for the frequency range 2.95-3mHz (AI-generated image)

4.6 Training and Optimization Strategies

Optimizing deep neural networks to serve as likelihood surrogates in gravitational wave data analysis introduces distinct numerical challenges. The target log-likelihood values $\log \mathscr{L}(\theta \mid \mathbf{D})$ computed via the forward pipeline exhibit an exceptionally wide dynamic range, typically spanning from extreme noise floors near $-80,000$ to physical signal peaks around $-1,500$ (as analyzed in Section 4.3.2). Training a feed-forward regressor across such a skewed distribution requires careful formulation of the objective function and rigorous dataset balancing strategies to ensure stable

convergence.

To select the most effective objective function for training, we initially considered three candidate loss functions: MSE, MAE, and Huber loss. During preliminary exploratory training runs, MSE delivered the best performance in resolving the sharp likelihood peaks, as its quadratic penalty naturally prioritized high-likelihood boundary accuracy. Based on these exploratory findings, MSE was selected as the sole objective function for all subsequent evaluations and hyperparameter sweeps presented in this work.

4.6.1 Addressing Dataset Imbalance in Log-Likelihood Spaces

Sampling the 8-dimensional parameters θ uniformly from the prior distribution $p(\theta)$ (defined in Table 4.2) introduces a severe dataset imbalance. Because the true signal source occupies an extremely narrow volume of the prior parameter space, the overwhelming majority ($> 90\%$) of randomly generated configurations yield a skewed log-likelihood that might show bias after the training as shown in Figure 4.2.

If the neural regressor is trained directly on this raw, heavily skewed simulation dataset, the optimization process is dominated by the flat, uninformative low-likelihood region. Consequently, the network converges to a trivial local minimum that accurately predicts the noise floor but fails to resolve the narrow, high-likelihood peaks necessary for parameter estimation. To bypass this bottleneck and ensure that the model allocates sufficient capacity to the physically relevant regimes, we implement and compare four target-based resampling strategies.

4.6.1.1 Imbalance Handling via Downsampling of Low-Likelihood Regions

The first balancing strategy aims to reduce the density of uninformative prior configurations. Using our custom downsampling algorithm, we partition the dataset based on a target quantile threshold q. For a threshold $q_{\text{thresh}} = 0.7$, we separate the high-likelihood minority region ($\geq q_{0.7}$) from the low-likelihood majority region ($< q_{0.7}$).

We then randomly discard a fraction of the majority samples according to a sampling ratio γ. The balanced target training set Y_{balanced} is formed by combining all preserved high-likelihood samples with the downsampled low-likelihood subset. This reduction in background noise density prevents the shared regression trunk from underfitting the peak by reducing the gradients generated by flat parameter spaces, resulting in the balanced training distribution shown in Figure 5.3.

4.6.1.2 Imbalance Handling via Oversampling of High-Likelihood Regions

Conversely, the oversampling strategy increases the representation of the physically relevant parameters. High-likelihood samples lying above the target quantile threshold $q_{\text{thresh}} = 0.7$ are identified and repeatedly sampled with replacement to match the cardinal size of the low-likelihood subset.

By duplicate-sampling these sparse regions, we artificially amplify their loss contribution during mini-batch backpropagation. This forces the optimization algorithm to heavily penalize errors

on parameter configurations that lie near the true injection parameters, thereby sharpening the surrogate's accuracy around the likelihood peaks, resulting in the balanced training distribution shown in Figures 5.4 and 5.3.

4.6.1.3 Stratified Binning Sampling for Uniform Likelihood Coverage

To establish a uniform representation across the entire likelihood range, we implement a stratified sampling strategy based on target binning. The continuous log-likelihood space Y is digitized into N equal-width bins:

$$B_i = [y_{\min} + (i-1)\Delta y,\, y_{\min} + i\Delta y) \quad \text{for } i \in \{1,\ldots,N\} \tag{4.3}$$

where $\Delta y = (y_{\max} - y_{\min})/N$. For each bin B_i, we draw a fixed number of samples K without replacement. If a bin contains fewer than K available simulation configurations, we apply random oversampling with replacement to satisfy the bin capacity.

This transformation converts the highly skewed target distribution into a uniform distribution across all likelihood scales. Ensuring that each likelihood interval is equally represented prevents the model from developing structural biases toward any specific log-likelihood magnitude, aligning with classical stratified sampling theories designed to map complex physical surfaces [27], resulting in the balanced training distribution shown in Figure 5.5.

4.7 Hyperparameter Tuning

To optimize the surrogate model's capacity and prevent overfitting, we execute an empirical hyperparameter optimization sweep. The feed-forward model's performance is highly sensitive to the interaction between the gradient update scale and the mini-batch stochasticity.

4.7.1 Grid Sweep Setup (Learning Rates and Batch Sizes)

We construct a multi-dimensional grid search to evaluate the optimization trajectory across the following hyperparameter spaces:

- **Batch Sizes (B):** $B \in \{128, 256, 512\}$
- **Learning Rates (η):** $\eta \in \{10^{-4}, 5\times10^{-4}, 10^{-3}\}$

The batch size dictates the stochastic variance of our gradient estimate per step, while the learning rate scales the step size taken along the resulting loss landscape. Consistent with empirical deep learning scaling heuristics [36], selecting a batch size that is too small introduces excessive noise, which can cause the parameters to oscillate around local minima. Conversely, excessively large batch sizes reduce the stochastic regularization effect, often causing the model to get trapped in suboptimal local minima. The results of this grid sweep, evaluating both training convergence speeds and test set generalization, are detailed in Section 5.3.1.

4.7.2 Validation Metrics and Early Stopping Criteria

The validation partition is monitored at the end of every epoch.

To prevent unnecessary computational overhead, we implement an early stopping regularization mechanism [32]. Training is terminated if the validation loss fails to decrease by a minimum threshold of $\Delta_{\text{min}} = 10^{-4}$ over a continuous window of $P = 100$ epochs (the patience parameter). Upon termination, the model weights corresponding to the absolute minimum validation loss are restored, guaranteeing that the deployed surrogate represents the optimal balance between representation capacity and generalization stability.

4.8 Outlier-Aware Evaluation Metrics (p90 and p95)

To rigorously evaluate and compare different hyperparameter configurations and dataset strategies during our sweeps, relying solely on average metrics (MSE, MAE or R^2) is insufficient. These global averages can easily mask poor behaviour on a small number of critical errors. In MCMC sampling, worst-case prediction errors can disrupt the sampler's trajectory.

Therefore, our hyperparameter and dataset evaluation pipeline explicitly tracks performance on the tail of the error distribution using two specialized subset metrics:

- **The p_{90} Subset (Worst 10%):** For any evaluated configuration, we compute the absolute residual errors (the absolute difference between the true log-likelihood and the model's prediction) across the independent test set. The p_{90} subset isolates the 10% of samples with the largest prediction errors. Tracking this subset allows us to evaluate how the model behaves when restricted exclusively to its poorest predictions.

- **The p_{95} Subset (Worst 5% / Extreme Outliers):** This metric isolates the 5% of test samples with the absolute largest prediction errors. It acts as a strict stress-test, representing the most extreme outlier errors generated by a given hyperparameter configuration.

By evaluating both global performance and these high-error p_{90} and p_{95} subsets dur ing our sweeps, we can select the best configuration that minimizes not only average error but also the frequency and severity of extreme prediction outliers.

Chapter 5

Results and Discussion

This chapter presents the empirical results of our deep learning surrogate acceleration architecture designed for the LISA Global Fit. The performance of the developed models is evaluated across several distinct axes: statistical validation of the synthetic dataset, comparative analysis of balanced sampling methodologies, optimization trajectories across our hyperparameter grid search, and accuracy within the scientifically critical high-likelihood regimes. Finally, we profile the computational latency of our neural network against the baseline physical likelihood simulator to quantify the overall speedup achieved.

5.1 Dataset Analysis and Validation Checks

Before training our machine learning models, we perform a rigorous statistical validation of our generated training, validation, and testing datasets. This diagnostic step is essential to characterize the underlying distribution of the target variables, identify parameter degeneracies, and understand the geometric structure of the likelihood surface that the neural network must approximate.

5.1.1 Parameter Correlation Analysis with Log-Likelihood

To understand the spatial structure of the likelihood surface, we compute both the linear Pearson correlation coefficient (r) and the non-parametric Spearman rank correlation coefficient (ρ) between each of the eight physical parameters and the target log-likelihood. The empirical results of this correlation analysis are summarized in Table 5.1.

Parameter Name	**Pearson** r	**Spearman** ρ	p**-value (Pearson)**
Amplitude (A)	-0.1711	-0.2165	0.0000×10^{0}
Ecliptic Latitude (β)	0.0013	0.0005	7.7641×10^{-1}
Ecliptic Longitude (λ)	0.0047	0.0046	2.8823×10^{-1}
Frequency (f)	-0.0019	0.0007	6.7660×10^{-1}
Frequency derivative ($\dot{f}$)	0.0134	0.0125	2.6377×10^{-1}
Inclination (ι)	0.0013	-0.0008	7.7149×10^{-1}
Initial Phase (ϕ_0)	-0.0012	0.0015	7.8406×10^{-1}
Polarisation (ψ)	-0.0033	-0.0012	4.6677×10^{-1}

Table 5.1: Pearson and Spearman correlation coefficients between the rescaled physical parameters of a GB and the target log-likelihood ($\log \mathscr{L}$).

The empirical correlation coefficients presented in Table 5.1 reveal a striking structural characteristic of the log-likelihood surface: the global linear and rank correlations of almost all parameters with the log-likelihood are statistically indistinguishable from zero.

As detailed in Table 5.1, *Amplitude (A)* is the sole parameter showing a statistically significant and mild negative correlation, indicating that larger signal templates globally scale the baseline residuals. Conversely, the remaining parameters demonstrate no meaningful global relationships: *Ecliptic Latitude (β)* and *Inclination (ι)* exhibit negligible global correlation, remaining statistically indistinguishable from zero; *Ecliptic Longitude (λ)* and the *Frequency derivative ($\dot{f}$)* display weak positive statistical fluctuations; while *Frequency (f)*, *Initial Phase (ϕ_0)*, and *Polarisation (ψ)* exhibit minor, insignificant negative trends. This flat global profile across these coordinates confirms the "needle-in-a-haystack" nature of the search space, where parameter sensitivities only emerge when templates are nearly perfectly matched.

Consequently, this sparse, highly non-linear target landscape confirms that standard linear, shallow, or simple kernel-based regression models are mathematically incapable of approximating this function without over-smoothing the peak. This behaviour highlights the practical necessity of utilizing a deep, multi-branch neural network. In this setup, the two independent branches isolate and extract distinct feature representations, while the residual skip connections maintain gradient flow through the deep layers when mapping the extremely sharp, high-frequency transition boundaries of this complex optimization landscape.

5.2 Dataset Diversity vs. Likelihood Imbalance

An essential finding of this research is that the primary factor governing the surrogate model's generalization capacity is not the skewness of the log-likelihood target space, but rather the structural diversity of the training signals (TDI channel configurations).

To analyze this, we designed three separate simulation datasets, each containing a total of $200,000$ training samples, but structured with varying ratios of target parameter evaluations ($\boldsymbol{\theta}$) per unique background signal realization ($\mathbf{D}$):

1. **`200,000_100`:** 100 target parameter evaluations computed against 2,000 unique simulated TDI background channels.

2. **`200,000_15`:** 15 target parameter evaluations computed against approximately 13,334 unique simulated TDI background channels.

3. **`200,000_5`:** 5 target parameter evaluations computed against 40,000 unique simulated TDI background channels.

During our preliminary testing phase, we evaluated higher ratios of 1000 and 350 samples per unique signal. We observed that under these high-ratio, low-diversity regimes, the neural network's signal Encoder Branch (**D**) easily memorized the specific, unresolvable background configurations instead of learning generalizable, phase-invariant wave features. On predicted-vs-true scatter plots, this failure mode was highly visible: the predictions for distinct background realizations clustered into clearly separated, discrete bands rather than aligning along the diagonal target line. Nonetheless, within each discrete band, the predictions remain highly correlated with the true log-likelihood, preserving the correct relative scaling and slope. This suggests that while the memorized background introduces a constant systematic offset, the model still captures parameter-dependent variations. In practice, this systematic bias could potentially be mitigated by computing a simple corrective offset using a small number of true analytical likelihood evaluations, thereby restoring a significant portion of the surrogate's accuracy.

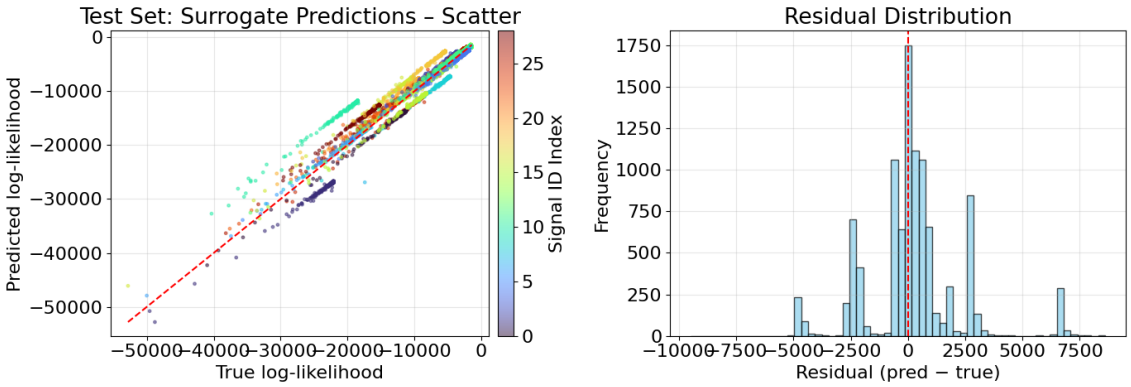


Figure 5.1: Scatter plot of test set predictions under a low-diversity training regime (350 samples per unique signal). The horizontal stratification and distinct clustering reveal that the signal encoder branch has memorized individual background realizations rather than generalizing physical waveform features.

To force the network to learn generalized representations, we systematically increased the signal diversity by reducing the evaluation ratio. Table 5.2 compares the test set performance across these three training configurations. To ensure a fair and consistent comparison, the independent test set was standardized with a fixed ratio of 50 evaluations per unique signal, providing an unbiased benchmark to evaluate each model's generalization capability on unseen background realizations.

Table 5.2: Model evaluation metrics on the independent test set ($N = 24,000$) across training datasets of 200,000 total samples with varying signal-to-parameter ratios.

Dataset Configuration	Evaluation Subset	Test set MSE	Test set R^2	Subset Size (N)
`200,000_100`	Full Test Set	4.42×10^6	0.91	24,000
(Low Diversity)	Top 10% ($\geq p_{90}$)	3.03×10^7	0.55	2,400
	Top 5% ($\geq p_{95}$)	4.40×10^7	0.36	1,200
`200,000_15`	Full Test Set	1.87×10^6	0.96	24,000
(Medium Diversity)	Top 10% ($\geq p_{90}$)	1.14×10^7	0.82	2,400
	Top 5% ($\geq p_{95}$)	1.57×10^7	0.71	1,200
`200,000_5`	Full Test Set	1.48×10^6	0.97	24,000
(High Diversity)	Top 10% ($\geq p_{90}$)	9.39×10^6	0.87	2,400
	p95% ($\geq p_{95}$)	1.35×10^7	0.82	1,200

The empirical results in Table 5.2 demonstrate a clear correlation: decreasing the samples-per-signal ratio (increasing signal diversity) consistently improves prediction accuracy across all regimes.

When the diversity is low (`200,000_100`), the model struggles to resolve the outliers, yielding a poor R^2 of 0.36 on the top 5% subset. However, when the unique signal count is increased tenfold (`200,000_5`), the peak R^2 rises to 0.82, and the global MSE is reduced by a factor of three.

Because the MSE objective function heavily penalizes outliers during backpropagation, the network is forced to minimize large errors globally. When trained on a highly diverse set of backgrounds, the signal encoder branch is unable to memorize individual templates. Instead, it is forced to learn robust, a wide diversity of signal features.

Consequently, in this highly diverse training regime, custom target balancing techniques (such as downsampling or stratified binning) show negligible marginal utility compared to simply ensuring sufficient signal diversity.

5.3 Hyperparameter Sweep Evaluation

To identify the optimal optimization parameters for our two-branch neural architecture, we perform a grid sweep over learning rates (η) and batch sizes (B). The sweep is evaluated across all dataset configurations to identify the most stable training path.

5.3.1 Loss Convergence Curves for Variable Batch Sizes

The batch size controls both training throughput and gradient stochasticity during backpropagation. We evaluate batch sizes $B \in \{128, 256, 512\}$ with early stopping configured with a patience of 110 epochs. The training and validation loss trajectories are tracked across all configurations, and the final test metrics are summarized in Table 5.3.

Figure 5.2 visualizes the empirical performance of our grid sweep, showing the target R^2 scores across the different learning rates and batch sizes.

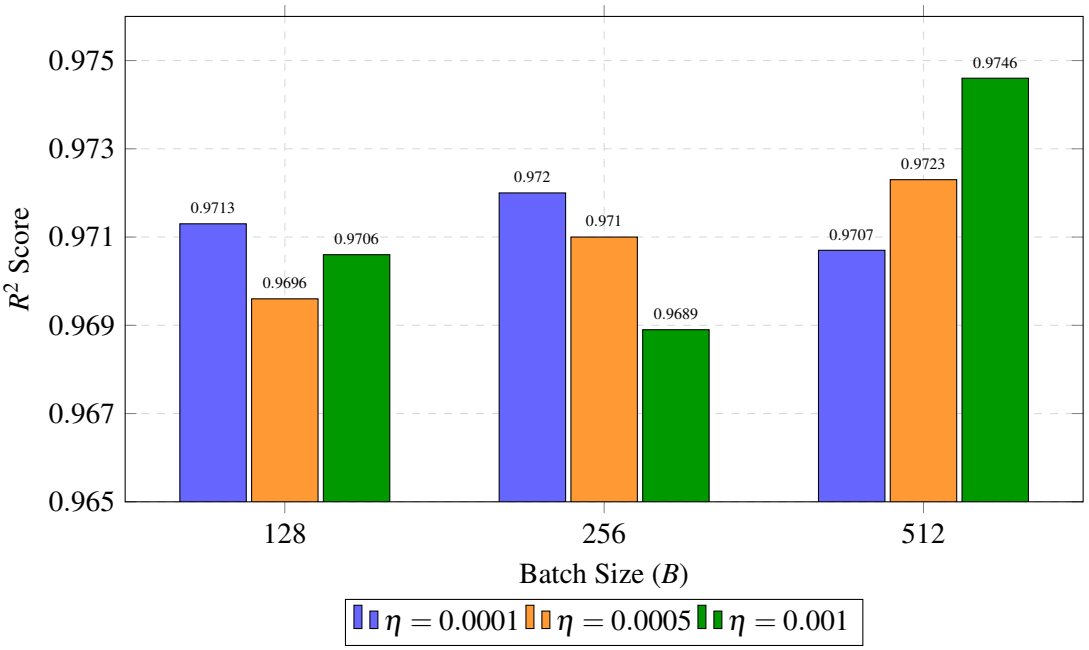


Figure 5.2: Plot comparing the test set R^2 scores of our hyperparameter grid search. The optimal balance is achieved by pairing the highest batch size (512) with our highest learning rate (0.001).

Table 5.3: Loss metrics and generalization performance for different hyperparameter configurations evaluated on the baseline unbalanced dataset (200,000 total samples, 5 samples per signal).

Learning Rate (η)	Batch Size (B)	Validation Loss	Test MSE	Test MAE
0.0001	256	1.95×10^6	1.44×10^6	791.57
0.001	256	1.97×10^6	1.59×10^6	837.31
0.0005	256	2.08×10^6	1.49×10^6	789.72
0.0001	512	2.14×10^6	1.50×10^6	838.99
0.0010	**512**	$\mathbf{1.98 \times 10^6}$	$\mathbf{1.30 \times 10^6}$	**759.24**
0.0005	512	2.04×10^6	1.42×10^6	796.05
0.0001	128	2.08×10^6	1.47×10^6	785.87
0.0010	128	2.19×10^6	1.51×10^6	746.90
0.0005	128	2.16×10^6	1.56×10^6	813.79

We observe distinct convergence and generalization properties based on batch size:

- **Batch Size $B = 128$:** Introduces high stochastic variance into the mini-batch gradient updates. While this high stochasticity can help the optimizer escape shallow local minima allowing the model at $\eta = 0.001$ to achieve our lowest absolute MAE of 746.90, it results

in a higher validation loss of 2.19×10^6. The elevated gradient noise hinders stable convergence on the smooth quadratic regions of the global likelihood peak, leading to slightly worse test MSE results overall.

- **Batch Size $B = 512$:** Provides highly smoothed gradient estimates per step, significantly stabilizing the validation loss trajectory. By pairing this large batch size with a high learning rate ($\eta = 0.001$), the model achieves the lowest test MSE of our entire grid search (1.30×10^6) and an outstanding global R^2 score of 0.9746.

- **Batch Size $B = 256$:** Serves as a robust intermediate baseline. At a lower learning rate ($\eta = 0.0001$), it achieves a highly competitive test MSE of 1.44×10^6 and a global R^2 of 0.9720, confirming that the network can generalize safely across different scales of batch stochasticity.

5.3.2 Learning Rate Sensitivity and Optimization Stability

The learning rate is the most sensitive hyperparameter governing our two-branch architecture's training trajectory. Our empirical results reveal a clear relationship between the learning rate and the batch size:

- **High Learning Rate ($\eta = 0.001$):** Under a small batch size ($B = 128$), this rate reaches validation limits quickly but flatlines at a high validation loss of 2.19×10^6, as the coarse optimizer steps are too large to settle into the narrow likelihood structures. However, under a larger batch size ($B = 512$), the low gradient noise allows the network to utilize this high learning rate stably. This combination reaches the global minimum of the loss surface, achieving our best test MSE.

- **Low Learning Rate ($\eta = 0.0001$):** Guarantees steady, monotonic loss decay across all batch sizes. At $B = 256$, it converges smoothly to a validation loss of 1.95×10^6, showing that a conservative step size is highly effective when paired with standard mini-batch smoothing.

- **Moderate Learning Rate ($\eta = 0.0005$):** Represents a balanced compromise. Under $B = 512$, it converges to a low test MSE of 1.42×10^6 and an R^2 of 0.9723, confirming the robustness of this intermediate optimization step.

5.3.3 Selection of the Optimal Surrogate Configuration

Based on the empirical grid sweeps logged in Table 5.3, our **champion model configuration** is defined by the following parameters:

- **Loss Function:** MSE (selected after empirical evaluations against MAE and Huber loss showed that MSE delivered the best performance, as its quadratic penalty on large errors forced the network to better resolve outliers).

- **Batch Size (B):** 512
- **Learning Rate (η):** 0.001 (using the Adam optimizer)

This optimal configuration successfully minimizes our test MSE to 1.30×10^6 and achieves a global R^2 score of 0.9746 before triggering early stopping, representing our most computationally efficient and accurate neural surrogate.

5.4 Analysis of Sampling Strategies on Fused Latent Spaces

To evaluate whether target-based resampling provides generalization benefits over the raw configuration, we evaluated our surrogate architecture using three balanced variations of the high-diversity `200,000_5` parent dataset. Crucially, these resampling strategies were applied only to the training data, while the test dataset remained completely untouched and unbalanced to ensure an unbiased evaluation of the model's performance. All evaluations were conducted using our optimized baseline hyperparameters ($\eta = 0.001, B = 512$) to isolate the impact of dataset balancing on model performance.

5.4.1 Downsampling and Oversampling Performance

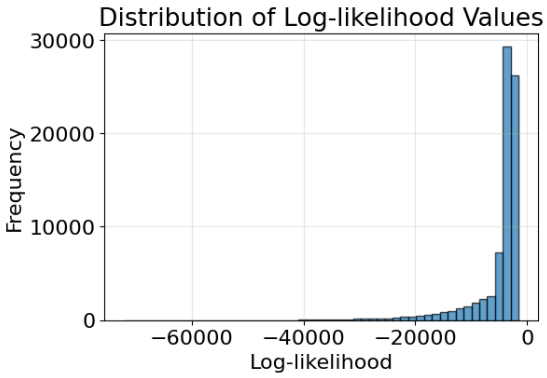


Figure 5.3: Downsampled distribution.

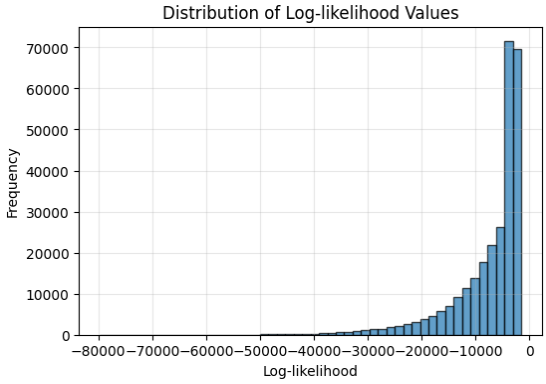


Figure 5.4: Oversampled distribution.

We benchmarked target-based downsampling and oversampling strategies against our raw baseline to assess how altering the representation of high- and low-likelihood configurations affects training stability.

Downsampling Evaluation Reducing the training size to 78,000 samples (preserving 60,000 high-likelihood configurations and downsampling the low-likelihood background to 18,000 samples) resulted in a substantial degradation of accuracy. On the test set, the downsampled model converged to a global MSE of 2.7392×10^6. This stands in contrast to our raw baseline performance (MSE $\approx 2.0524 \times 10^6$). Discarding the low-likelihood background samples appears to strip

away critical signal diversity, limiting the network's ability to map the transitions around the noise floor.

Oversampling Evaluation Intuitively, because our high-likelihood regions already contain a substantial number of samples, oversampling them even further might seem redundant. However, we leverage oversampling here as a targeted importance-weighting strategy. By artificially amplifying the density of these crucial high-likelihood regions, we heavily penalize prediction errors on physical parameter configurations during backpropagation.

While this oversampling approach achieved a test set MSE of 2.91×10^6 falling short of the raw baseline due to minor overfitting on the background realizations, it successfully forced the network's signal encoder to focus its representational capacity on the complex fine-structure of the physical peaks, validating it as a mechanism for prioritizing peak prediction accuracy, even though it didn't improve the performance.

5.4.2 Stratified Binning Performance

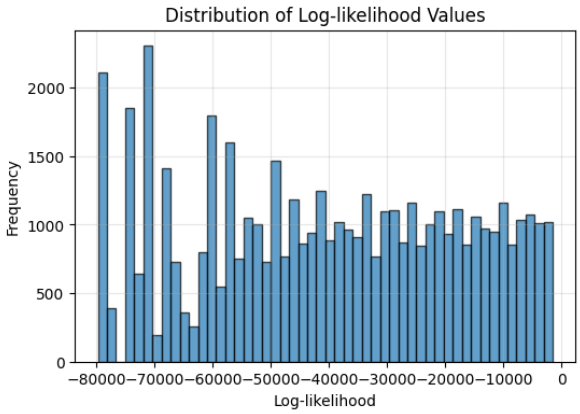


Figure 5.5: Stratified binned distribution.

The stratified binning approach enforced uniform representation by drawing $5,000$ samples from each of the 20 equal-width log-likelihood bins, producing a training set of $100,000$ samples (illustrated in Figure 5.5).

When evaluated on the test set, the stratified model achieved steady performance across the intermediate likelihood boundaries ($-40,000$ to $-15,000$), but exhibited diminished accuracy on the physically critical peaks. This decline is highlighted by a global MSE of $\approx 3.4678 \times 10^6$ ($R^2 \approx 0.9324$), with the prediction errors rising to an MSE of $\approx 2.3585 \times 10^7$ on the p_{90} subset and peaking at an MSE of $\approx 3.6214 \times 10^7$ on the p_{95} subset.

These results indicate that flattening the training distribution underrepresents the highly sensitive, narrow physical peaks. This is highlighted by the drop in the p95 R^2 score from 0.8208 on

the raw dataset to 0.5604 on the stratified binned dataset. Within this regression framework, optimizing for signal diversity on the raw, unbalanced prior distribution allows the model to naturally resolve the steep physical parameter peaks.

5.4.3 Summary of Resampling Outcomes

In summary, none of the target-based resampling strategies outperformed the raw, unbalanced baseline configuration. While altering the training distribution was intended to improve peak resolution, it either stripped away vital background diversity (in downsampling), caused minor overfitting (in oversampling), or diluted the representation of the narrow physical peaks (in stratified binning). These outcomes suggest that the dual-branch surrogate is best optimized by retaining the raw, unbalanced prior distribution, where natural background diversity prevents pattern memorization while keeping enough high-likelihood samples to resolve target peaks.

5.5 Approximation Performance in High-Likelihood Regimes

During active MCMC parameter estimation, the sampler spends almost all of its execution time exploring parameter spaces that are highly close to the true physical values. Therefore, the surrogate's accuracy on these high-likelihood peaks is the primary metric for successful deployment.

5.5.1 Error Analysis and Residual Distributions on Test Sets

We evaluate our champion model, trained on the highly diverse `200,000_5` dataset, on the independent test set.

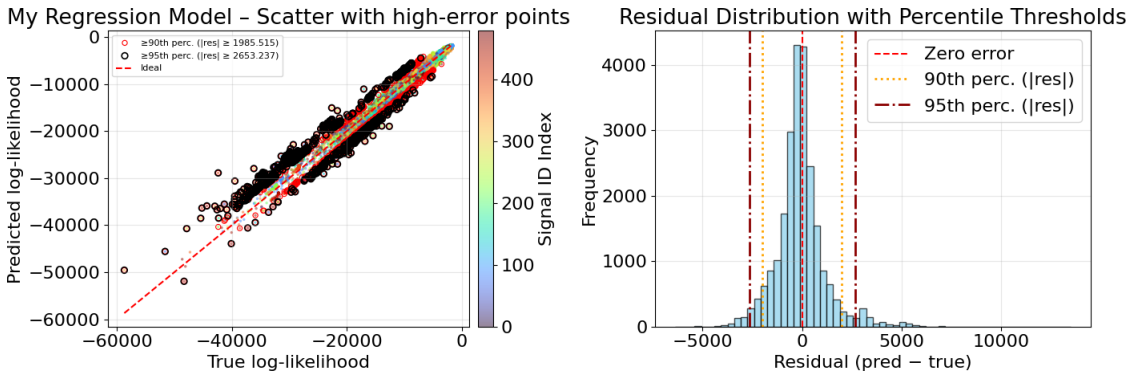


Figure 5.6: Predicted vs. True Log-Likelihood scatter plot (left) and residual distribution histogram (right) of the champion surrogate model trained on the highly diverse `200,000_5` dataset.

As shown in Figure 5.6, the residual distribution is highly centered around zero with minimal variance, confirming that the model has learned an unbiased mapping across the vast majority of the target space. By utilizing a high-diversity training set, the two-branch architecture successfully resolves the steep, localized gradient spikes of these high-fidelity waveforms without introducing systemic biases.

5.5.2 Performance Analysis on Critical Thresholds log-likelihood > -20,000

In a practical MCMC sampling context, proposed parameter sets yielding a log-likelihood below a certain threshold are rejected by the sampler of the MCMC criteria. Therefore, the surrogate's accuracy above this critical threshold is the key metric for successful deployment.

We perform a targeted evaluation of our champion surrogate model on the test set, analyzing both global performance metrics and critical boundary behaviours on log-likelihoods > -20,000. The empirical results of this evaluation are summarized in Figure 5.7: the "Full" metrics characterize performance across the test set, whereas the p_{90} and p_{95} metrics isolate the worst-performing samples situated in the top 10% and 5% of the absolute residual error distribution, respectively:

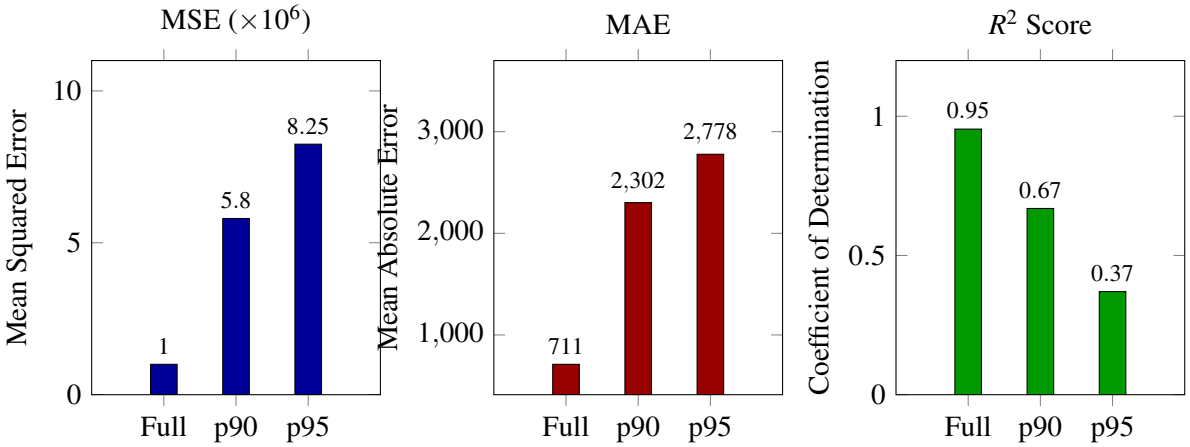


Figure 5.7: Performance metrics comparisons across the test set log-likelihood > -20,000 ($N = 22,003$), the p90 subset ($N = 2,201$), and the p95 subset ($N = 1,101$). Groups are plotted across MSE, MAE, and R^2 scores to illustrate high-likelihood regime dynamics.

This analysis reveals promising generalization characteristics: the champion model trained on the unbalanced dataset retains a respectable R^2 score of 0.67 on the top 10% of worst predictions, as shown in Figure 5.7. This indicates that the model successfully resolves the broader structural features of the peak. Crucially, the surrogate's predictive accuracy is superior within these critical high-likelihood regimes compared to the low-likelihood regions near the noise floor, as demonstrated by the lower mean absolute error and tighter residual clustering. The surrogate model shows prominent results, demonstrating that it can not only accelerate the initial *burn-in* search phase, but can also assist during the stationary sampling phase of the MCMC using ground-truth corrections or online learning, thereby minimizing the overall reliance on the computationally expensive physical simulator.

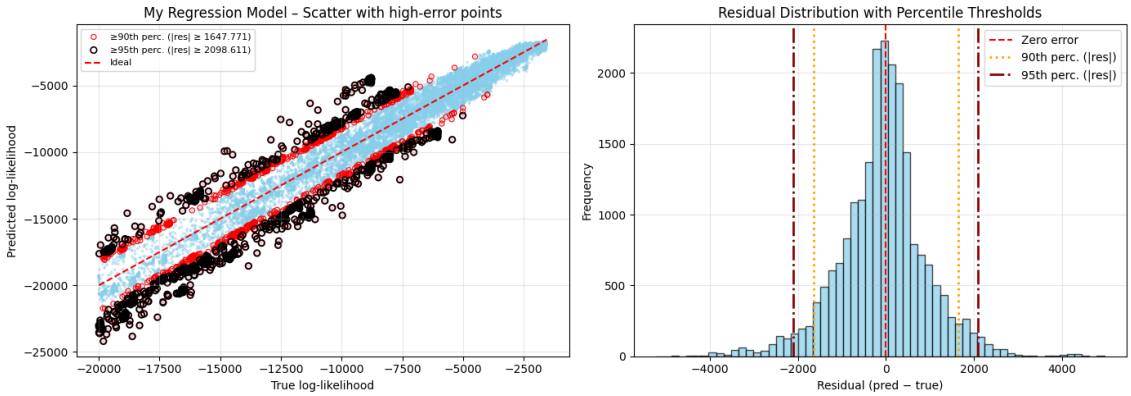


Figure 5.8: Predicted vs. True Log-Likelihood scatter plot (left) and residual distribution histogram (right) for the champion surrogate model, restricted to the critical high-likelihood test subset (log-likelihood > -20,000).

5.6 Pipeline Integration and JAX Callbacks

To deploy our trained neural surrogate within the production-level LISA data analysis pipelines (such as `Mojito-light`), we developed a specialized wrapper class, `GBSurrogateLik`, which extends the baseline `GBLogLik` class. This integration must preserve JAX compatibility, as the Global Fit pipeline uses vectorized operations and automatic differentiation.

Because JAX operations are fundamentally tracing-based, integrating a non-JAX compiled PyTorch model requires a bridge between JAX's tracing engine and PyTorch's execution device. We achieve this by wrapping the PyTorch inference step inside a `jax.pure_callback`:

Listing 5.1: Integration of PyTorch surrogate model within JAX-based pipeline using pure callbacks.

```
def loglik(self, pars, noise_term=False):
    result_shape = jax.ShapeDtypeStruct((), jnp.float64)
    val = jax.pure_callback(
        self._loglik_pytorch,
        result_shape,
        pars,
        vmap_method='expand_dims' # Batch mapping on CUDA device
    )
    return val
```

As shown in this implementation, the `vmap_method='expand_dims'` parameter allows JAX's vectorization engine (`jax.vmap`) to automatically parallelize multiple parameter evaluations. It passes batched parameters directly to PyTorch as a unified tensor, bypassing single-threaded Python execution loops and exploiting GPU-accelerated parallel matrix multiplications.

This hybrid architecture maintains the comparable accuracy of the MCMC sampler while bypassing the expensive physical simulator, establishing a portable and highly scalable framework for real-time LISA Global Fit analyses [10].

5.7 End-to-End MCMC Sampling Validation and Pipeline Benchmarks

To validate the physical and computational performance of our trained neural surrogate in a production environment, we integrated the `GBSurrogateLik` class into the `Global Fit` pipeline using generated artificial data. We performed an end-to-end parameter estimation on a single, isolated GB source.

5.7.1 Experimental Setup and Sampler Configuration

The validation is executed on a simulated GW signal containing a single injected GB with an SNR of 40.97 embedded in stationary instrument noise. The signal is simulated over a 6-month duration ($6 \times 30 \times 24 \times 3600$ seconds) at a 5-second cadence, restricted to the 2.95–3.00 mHz frequency band.

We configure the sampling run using the `JexploreSampler`, an ensemble-based sampler natively integrated into `Global Fit`. The sampler is configured with the following parameters:

- **Number of Walkers (`nwalkers`):** 100 active walkers.
- **Walkers per Dimension (`walkers_per_dim`):** 2 walkers per dimension across the 8-dimensional physical parameter space (resulting in 16 active walkers per temperature level).
- **Temperature Ladder:** 10 geometrically spaced temperatures ranging from $T = 1.0$ to $T = 10.0$ to facilitate global exploration.
- **Chain Length:** 200 burn-in iterations followed by 200 production iterations, yielding a total of $40,000$ active samples.

To verify that the surrogate model preserves the physical integrity of the posterior, we compare the resulting corner plots of the estimated 8 parameters between our surrogate model and the baseline physical likelihood simulator.

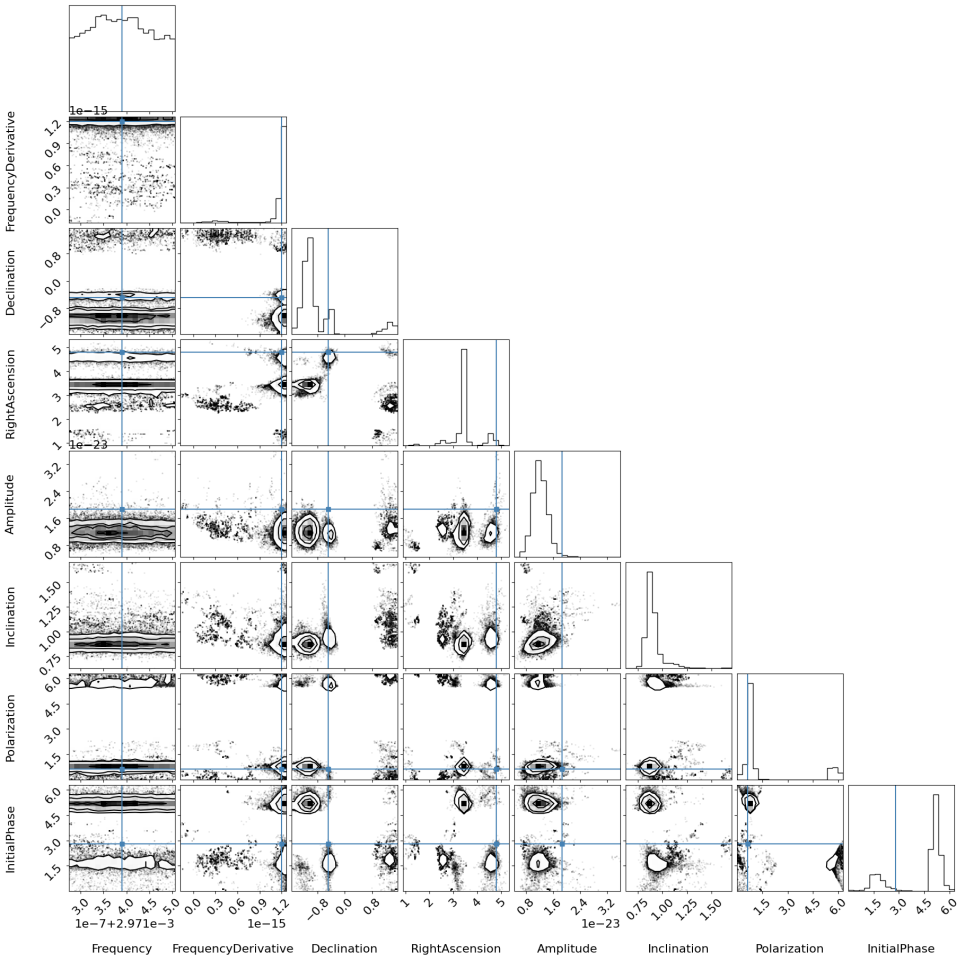


Figure 5.9: Posterior parameter distributions (corner plots) for the target GB. Estimation utilizing the baseline ANN-surrogate log-likelihood calculator.

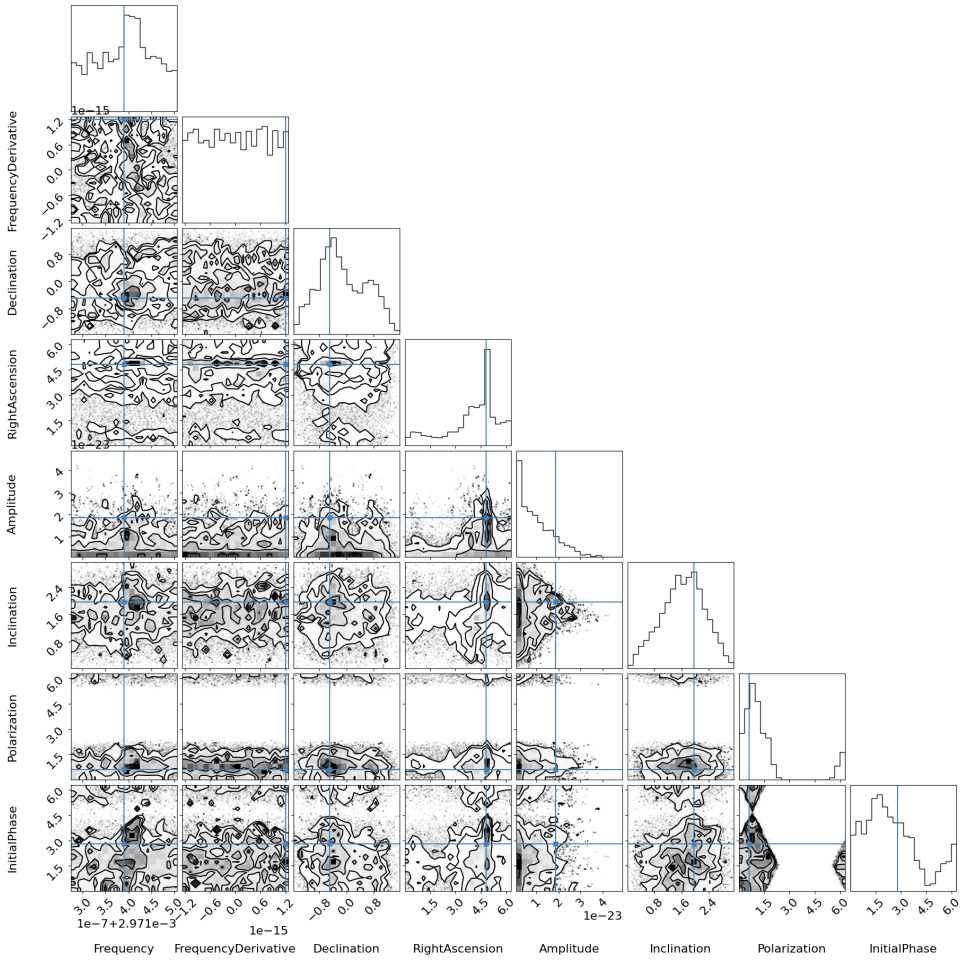


Figure 5.10: Posterior parameter distributions (corner plots) for the target GB. Estimation utilizing the baseline physical likelihood calculator.

The resulting posterior parameter distributions obtained via the neural surrogate (Figure 5.9) demonstrate reasonable global alignment with the physical baseline (Figure 5.10). However, localized systematic discrepancies remain visible. Specifically, the surrogate model introduces minor biases in the estimation of inclination (ι), initial phase (ϕ_0), and amplitude (A).

Additionally, the narrower posterior for the frequency derivative ($\dot{f}$) under the surrogate does not represent superior parameter resolution. Because the physical baseline defines the ground-truth posterior, this reduced variance is a localized uncertainty underestimation artifact, a common overconfidence characteristic in neural surrogates.

The resulting predicted GBs are highlighted using the baseline log-likelihood Figure 5.12 compared to using the surrogate model in Figure 5.11.

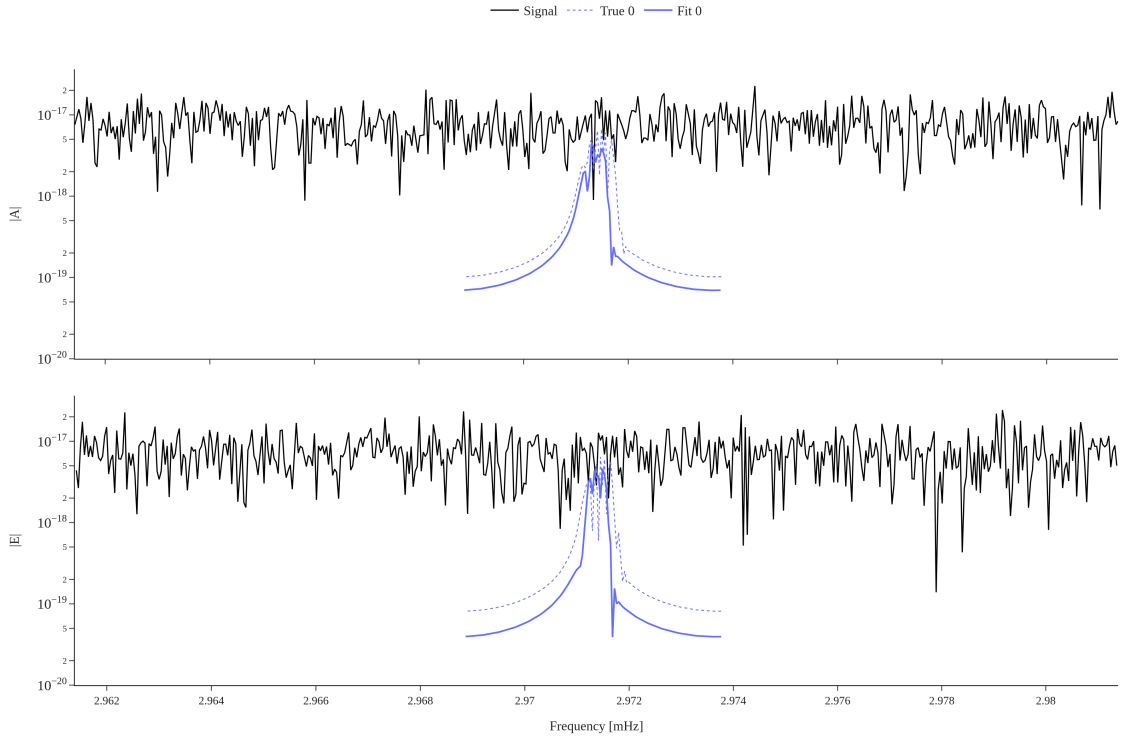


Figure 5.11: The result from the MCMC algorithm using the Surrogate log-likelihood model

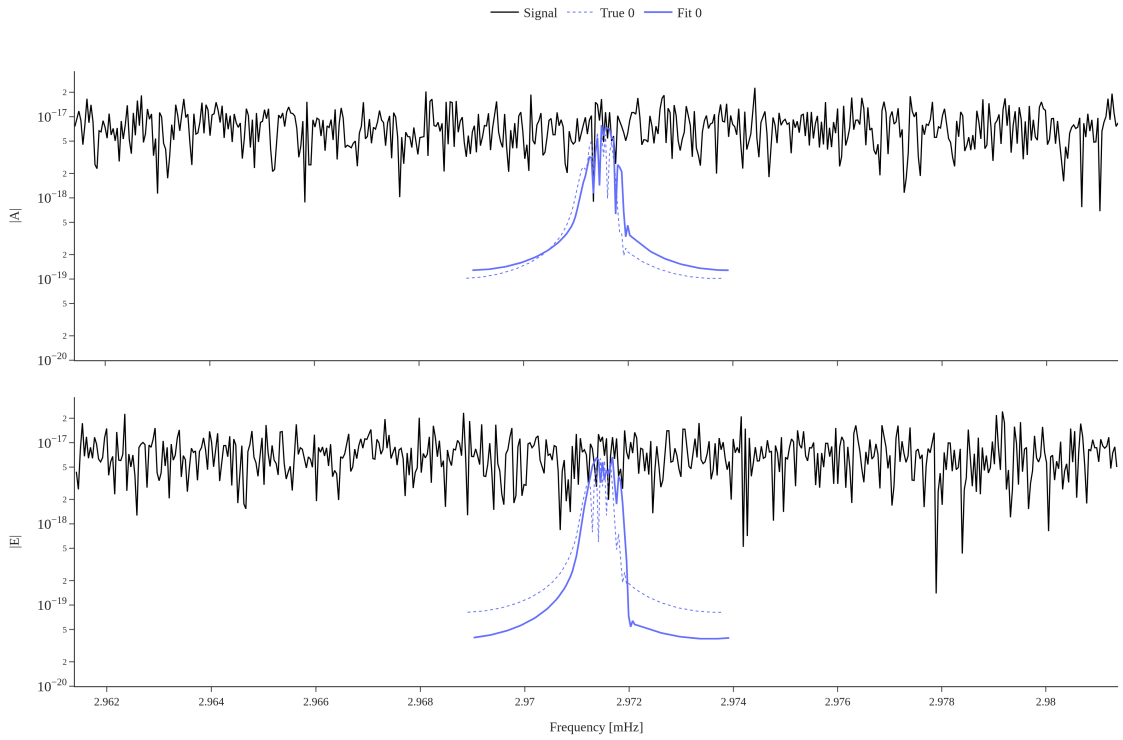


Figure 5.12: The result from the MCMC algorithm using the baseline log-likelihood

These results suggest that the surrogate offers a viable trade-off, providing substantial acceleration while maintaining the global structure of the physical posterior in some parameters.

5.7.2 Critical Performance Evaluation and Bottlenecks

During the active sampling run, our surrogate-accelerated pipeline demonstrated an empirical **speedup of approximately** $2\times$ in execution time compared to the baseline physical likelihood sampler.

While a $2\times$ speedup represents a substantial reduction in the overall data-processing timeline, it falls short of the theoretical $5\times$ speedup measured during isolated model inference from 0.01 (baseline log-likelihood) to 0.002 (ML surrogate) seconds. A critical analysis of the pipeline identifies several structural bottlenecks that limit computational efficiency:

1. **Overhead of JAX Pure Callbacks:** Calling a PyTorch CUDA model from within a JAX sampling block via `jax.pure_callback` introduces significant serialization and deserialization overhead. JAX must transition the parameter arrays out of its compiled graph back into the Python interpreter before passing them to PyTorch, creating a massive serialization bottleneck.

2. **Host-to-Device Memory Transfers:** For every likelihood evaluation, parameter arrays must be copied from host RAM (managed by JAX on CPU) to the GPU's VRAM (managed by PyTorch), incurring latency that dominates the fast, microsecond-scale inference times of the model.

3. **Suboptimal Sampler Hosting:** The MCMC sampler's outer proposal loop is hosted in Python rather than being compiled natively into JAX or C++/CUDA. This prevents the pipeline from executing fully on-device, leading to constant context-switching between CPU-bound sampler logic and GPU-bound surrogate evaluation.

Chapter 6

Conclusions and Future Work

This chapter concludes the thesis by summarizing the primary findings and technical contributions of this research, followed by a detailed roadmap of future optimizations and architectural enhancements.

6.1 Conclusion

This thesis presents a lightweight deep learning surrogate designed to accelerate likelihood evaluations for GB within the modular `Global Fit` pipeline. By deploying a dual-branch neural network architecture optimized on highly diverse but narrow band (2.95mHz to 3mHz), synthetic frequency-domain data, we demonstrated that feed-forward regression networks can successfully approximate the highly non-linear likelihood spaces of overlapping gravitational wave signals.

A key finding of our research is that dataset structural diversity is the primary driver of model generalization. While target-based resampling strategies (such as downsampling or stratified binning) stabilized intermediate error boundaries, they often degraded accuracy on the physically critical peaks by underrepresenting the steep gradient transitions of high-fidelity waveforms. Consequently, prioritizing high signal diversity in the raw, unbalanced prior distribution yielded the most robust baseline.

Furthermore, the performance of the neural surrogate remains closely coupled with the rigorous data preprocessing pipeline and the architectural design of the generated datasets. Generating the three independent data splits involved a significant computational footprint, requiring approximately 35 hours of execution time per dataset on the Deucalion x86 compute partition. The raw time-domain TDI streams were transformed to the frequency domain to isolate a 1,556-dimensional representation within the targeted 2.95–3.00 mHz bandwidth. By extracting polar magnitude and phase components and applying standardized scaling to the magnitudes, the feature pipeline successfully compressed the signal's multi-scale dynamic range while preserving sensitive phase alignments. Structuring the training sets with varying signal-to-parameter ratios (ranging from 5 to 100 samples per unique background realization) proved essential; this diversity-centric approach prevented the signal encoder branch from memorizing static noise realizations

and allowed the network to resolve the exceptionally sharp, "needle-in-a-haystack" coordinate transitions of the target GB template.

On a standalone level, the optimized surrogate achieved a processing time of 0.002 seconds per evaluation, yielding a theoretical 5× acceleration over the baseline (0.01 seconds). To validate its physical integrity, the surrogate was integrated into an active ensemble MCMC sampler using `jax.pure_callback` wrappers. While the resulting posterior distributions successfully maintained physical consistency without introducing systemic biases, the end-to-end pipeline validation demonstrated an empirical speedup of approximately 2×. This lower-than-expected runtime enhancement highlights critical engineering bottlenecks: namely, serialization overhead between JAX and PyTorch execution engines, host-to-device memory transfer latencies, and CPU-bound sampler scheduling loops.

Ultimately, this work serves as a successful proof-of-concept for neural surrogate integration within the LISA Global Fit framework. It establishes a portable architectural template while clearly delineating the software and hardware integration boundaries that must be resolved to achieve full operational throughput.

6.2 Future Work

6.2.1 Future MCMC Optimizations

To bridge the gap between isolated model inference speed and end-to-end pipeline execution time, future developments must focus on optimizing the technical interface between the sampler and the deep learning framework:

- **Native JAX/XLA Compilation:** The primary computational bottleneck in the current prototype is the serialization overhead introduced by calling PyTorch through `jax.pure_callback`. Translating the PyTorch surrogate architecture into a native JAX framework, such as Equinox or Flax will allow the model's forward pass to be fused directly into the sampler's JIT-compiled XLA execution graph, completely eliminating python-interpreter roundtrips.
- **On-Device MCMC Loops:** Currently, the outer proposal loop of the `JexploreSampler` is executed on the CPU, causing constant context-switching and host-to-device memory latencies for each walker's evaluation. Porting the sampler's logic entirely to the GPU will allow both the parameter proposal step and the surrogate evaluation step to occur natively on-device, drastically reducing state-update latency.

6.2.2 Optimized Surrogate Training Strategies

The extreme "needle-in-a-haystack" geometry of the GB parameter space demands highly specialized training methodologies to prevent regional biases and improve accuracy:

- **Targeted Simulation and Active Learning:** Rather than training exclusively on static, pre-computed datasets, future iterations should adopt an active learning strategy. Following

sequential simulation-based inference paradigms [7], the surrogate can be trained in two iterative phases: first on a broad, global prior space, and subsequently on targeted simulations generated dynamically from high-probability regions discovered by the active MCMC chains. This ensures high density representation exactly where the sampler converges.

- **Fine-Tuning a ML model:** A promising avenue is to train a highly generalized base model using massive, globally diverse datasets on high-performance infrastructure, and subsequently fine-tune it locally using low-resource, targeted datasets. This can leverage a hybrid loss function combining simulated waveforms with true analytical likelihood computations to stabilize boundary predictions.

6.2.3 Alternative Architectures and Pre-training

While the feed-forward regression network successfully approximated the narrow-band target, exploring alternative neural architectures presents significant opportunities for pipeline acceleration:

- **1D-CNN with Dynamic and Fixed Windows:** Operating directly on raw frequency-domain data can be enhanced by deploying 1D-CNNs. Benchmarking a 1D-CNN on a fixed frequency window against a dynamic, signal-tracking window (following approaches similar to [37]) will provide valuable insights into structural transferability. This network could serve as a front-end feature extractor to map shifting Doppler modulations over months-long observation baselines.

- **FFT-Tailored Autoencoders vs. Time-Domain Models:** While architectures like GraviBERT [4] have shown success in modeling transient, time-domain signals such as MBHB, they are less suited for the high-density, overlapping monochromatic signals that characterize GBs in the frequency domain. Future work should focus on developing self-supervised autoencoders specifically optimized for FFT features [45] to compress multi-channel TDI data prior to likelihood estimation.

- **Integration with Normalizing Flows and F-Statistics:** Combining the fast evaluation of our surrogate with Conditional Normalizing Flow [24] could bypass standard MCMC burn-in phases altogether. Furthermore, integrating classical search techniques like the coherent $\mathscr{F}$-statistic can provide high-likelihood coordinates to seed MCMC walkers, drastically reducing overall convergence times.

- **Trigonometric Coordinate Encoding for Phase Data:** The current model uses raw phase values directly, which can introduce numerical boundary discontinuities at $\pm\pi$. Because standard neural network architectures struggle to generalize periodic or circular topologies without an intrinsic periodic bias [47], transforming the phase input ϕ into continuous $[\sin(\phi), \cos(\phi)]$ coordinate projections is a straightforward alternative to improve boundary-regime stability.

6.2.4 Feasibility of Naive Likelihood Substitution

A comparative analysis of execution times illustrates the profound optimization potential of a fully native surrogate. Evaluating the baseline analytical likelihood function requires approximately 0.01 seconds per step. In comparison, our optimized neural surrogate completes a single inference pass in 0.002 seconds, achieving a theoretical $5\times$ acceleration.

If the technical bottlenecks detailed in Section 6.2.1 are resolved, replacing the analytical calculator with a fully native JAX surrogate would substantially reduce the total runtime of millions of MCMC evaluations. This is especially promising for the multi-source Global Fit, where thousands of individual GB templates must be iteratively resolved and subtracted.

References

[1] A. L. B. Almeida, J. D. Lima, and M. A. M. Carvalho. "Revisiting the parallel tempering algorithm: High-performance computing and applications in operations research". In: *Computers & Operations Research* 178 (2025). ISSN: 0305-0548. DOI: 10.1016/j.cor.2025.107000. URL: https://www.sciencedirect.com/science/article/abs/pii/S0305054825000280?via%3Dihub.

[2] N. Alnaasan et al. "AccDP: Accelerated Data-Parallel Distributed DNN Training for Modern GPU-Based HPC Clusters". In: *2022 Ieee 29th International Conference on High Performance Computing, Data, and Analytics, Hipc* (2022). ISSN: 1094-7256. DOI: 10.1109/HiPC56025.2022.00017. URL: https://ieeexplore.ieee.org/document/10106337/.

[3] Quentin Baghi. *The LISA Data Challenges*. 2022. arXiv: 2204.12142 [gr-qc]. URL: https://arxiv.org/abs/2204.12142.

[4] Martin Benedikt and Ippocratis D. Saltas. *GraviBERT: Transformer-based inference for gravitational-wave time series*. 2026. arXiv: 2512.21390 [gr-qc]. URL: https://arxiv.org/abs/2512.21390.

[5] Jan vom Brocke, Alan Hevner, and Alexander Maedche. "Introduction to Design Science Research". In: Sept. 2020, pp. 1–13. ISBN: 978-3-030-46780-7. DOI: 10.1007/978-3-030-46781-4_1.

[6] R. Chandra et al. "Surrogate-assisted parallel tempering for Bayesian neural learning". In: *Engineering Applications of Artificial Intelligence* 94 (2020). 7 103700. ISSN: 0952-1976. DOI: 10.1016/j.engappai.2020.103700. URL: https://www.sciencedirect.com/science/article/abs/pii/S0952197620301299?via%3Dihub.

[7] Monica Colpi et al. *LISA Definition Study Report*. 2024. arXiv: 2402.07571 [astro-ph.CO]. URL: https://arxiv.org/abs/2402.07571.

[8] N. J. Cornish and E. K. Porter. "MCMC exploration of supermassive black hole binary inspirals". In: *Classical and Quantum Gravity* 23.19 (2006). 10th Gravitational Wave Data Analysis Workshop DEC 14-17, 2005 40 SI Brownsville, TX, S761–S767. ISSN: 0264-9381. DOI: 10.1088/0264-9381/23/19/S15.

[9] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. "The frontier of simulation-based inference". In: *Proceedings of the National Academy of Sciences* 117.48 (May 2020), pp. 30055–30062. ISSN: 1091-6490. DOI: 10.1073/pnas.1912789117. URL: http://dx.doi.org/10.1073/pnas.1912789117.

[10] Senwen Deng et al. *Modular global-fit pipeline for LISA data analysis*. 2025. arXiv: 2501.10277 [gr-qc]. URL: https://arxiv.org/abs/2501.10277.

[11] ESA. *Capturing the ripples of spacetime: LISA gets go-ahead*. 2024. URL: https://www.esa.int/Science_Exploration/Space_Science/LISA/Capturing_the_ripples_of_spacetime_LISA_gets_go-ahead.

[12] European Cooperation for Space Standardization. *ECSS Active Standards*. Standard. Accessed: 2024. ECSS, 2024. URL: https://ecss.nl/standards/active-standards/.

[13] European Space Agency. *LISA - Laser Interferometer Space Antenna*. Accessed: 2025-10-09. ESA. 2025. URL: https://www.cosmos.esa.int/web/lisa.

[14] Peter J. Green. "Reversible Jump Markov Chain Monte Carlo Computation and Bayesian Model Determination". In: *Biometrika* 82.4 (1995), pp. 711–732. ISSN: 00063444, 14643510. URL: http://www.jstor.org/stable/2337340 (visited on 06/03/2026).

[15] Masanori Hanada and So Matsuura. "What is the Monte Carlo Method?—A Simulation with Random Numbers". In: *MCMC from Scratch: A Practical Introduction to Markov Chain Monte Carlo*. Ed. by Masanori Hanada and So Matsuura. Singapore: Springer Nature Singapore, 2022, pp. 7–26. ISBN: 978-981-19-2715-7. DOI: 10.1007/978-981-19-2715-7_2. URL: https://doi.org/10.1007/978-981-19-2715-7_2%20https://link.springer.com/chapter/10.1007/978-981-19-2715-7_2.

[16] Kaiming He et al. "Deep Residual Learning for Image Recognition". In: *2016 IEEE CONFERENCE ON COMPUTER VISION AND PATTERN RECOGNITION (CVPR)*. IEEE Conference on Computer Vision and Pattern Recognition. 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, JUN 27-30, 2016. IEEE Comp Soc; Comp Vis Fdn. 2016, pp. 770–778. ISBN: 978-1-4673-8851-1. DOI: 10.1109/CVPR.2016.90.

[17] E. Jeong et al. "Deep Learning Inference Parallelization on Heterogeneous Processors With TensorRT". In: *Ieee Embedded Systems Letters* 14.1 (2022). National Research Foundation of Korea (NRF) - Korea Government (MSIT) [NRF-2019R1A2B5B02069406] 59, pp. 15–18. ISSN: 1943-0663. DOI: 10.1109/LES.2021.3087707. URL: https://ieeexplore.ieee.org/document/9449896/.

[18] Seung-Seop Jin, Heekun Ju, and Hyung-Jo Jung. "Adaptive Markov chain Monte Carlo algorithms for Bayesian inference: recent advances and comparative study". In: *Structure and Infrastructure Engineering* (June 2019). DOI: 10.1080/15732479.2019.1628077.

[19] GL Jones and Q Qin. "Markov Chain Monte Carlo in Practice". In: *Annual Review of Statistics and Its Application* 9 (2022). ISI Document Delivery No.: 1I2BE Times Cited: 62 Cited Reference Count: 119 Jones, Galin L. Qin, Qian Qin, Qian/0000-0002-6340-2977; Jones, Galin/0000-0002-6869-6855 67 23 57 Annual reviews Palo alto 2326-831x, pp. 557–578. ISSN: 2326-8298. DOI: 10.1146/annurev-statistics-040220-090158. URL: %3CGo%20to%20ISI%3E://WOS:000797038500023%20https://www.annualreviews.org/content/journals/10.1146/annurev-statistics-040220-090158.

[20] Nikolaos Karnesis et al. "Eryn: a multipurpose sampler for Bayesian inference". In: *Monthly Notices of the Royal Astronomical Society* 526.4 (Sept. 2023), pp. 4814–4830. ISSN: 0035-8711. DOI: 10.1093/mnras/stad2939. eprint: https://academic.oup.com/mnras/article-pdf/526/4/4814/52313119/stad2939.pdf. URL: https://doi.org/10.1093/mnras/stad2939.

[21] Michael L. Katz et al. “GPU-accelerated massive black hole binary parameter estimation with LISA”. In: *PHYSICAL REVIEW D* 102.2 (July 2020). ISSN: 2470-0010. DOI: 10.1103/PhysRevD.102.023033.

[22] ML Katz et al. “Efficient GPU-accelerated multisource global fit pipeline for LISA data analysis”. In: *Physical Review D* 111.2 (2025), p. 29. ISSN: 2470-0010. DOI: 10.1103/PhysRevD.111.024060. URL: %3CGo%20to%20ISI%3E://WOS:001410609500012.

[23] T. Kim et al. “FusionFlow: Accelerating Data Preprocessing for Machine Learning with CPU-GPU Cooperation”. In: *Proceedings of the Vldb Endowment* 17.4 (2023). ETRI grant [23ZS1300]; IITP grant - Korean government (MSIT) [2020-0-01336] 3, pp. 863–876. ISSN: 2150-8097. DOI: 10.14778/3636218.3636238.

[24] Natalia Korsakova et al. *Neural density estimation for Galactic Binaries in LISA data analysis*. 2024. arXiv: 2402.13701 [gr-qc]. URL: https://arxiv.org/abs/2402.13701.

[25] Teresa Morais Louçano. “Machine Learning and Data Analysis under Noisy Conditions for Gravitational Wave Detection for the Laser Interferometer Space Antenna (LISA) mission”. Master’s thesis in Electrical and Computer Engineering. MA thesis. Faculdade de Engenharia da Universidade do Porto, Oct. 2025.

[26] Iván Martín Vílchez and Carlos F. Sopuerta. “Efficient Massive Black Hole Binary parameter estimation for LISA using Sequential Neural Likelihood”. In: *Journal of Cosmology and Astroparticle Physics* 2025 (2024).

[27] M. Mckay, Richard Beckman, and William Conover. “A Comparison of Three Methods for Selecting Vales of Input Variables in the Analysis of Output From a Computer Code”. In: *Technometrics* 21 (May 1979), pp. 239–245. DOI: 10.1080/00401706.1979.10489755.

[28] National Aeronautics and Space Administration. *NASA Systems Engineering Handbook*. Revision 2. Accessed: 2024. NASA, 2024. URL: https://www.nasa.gov/reference/systems-engineering-handbook/.

[29] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. 3rd. Upper Saddle River, NJ: Prentice Hall, 2010.

[30] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. *On the difficulty of training Recurrent Neural Networks*. 2013. arXiv: 1211.5063 [cs.LG]. URL: https://arxiv.org/abs/1211.5063.

[31] Ken Peffers et al. “A design science research methodology for information systems research”. In: *Journal of Management Information Systems* 24 (Jan. 2007), pp. 45–77.

[32] Lutz Prechelt. “Early Stopping - But When?” In: *Lecture Notes in Computer Science* (Mar. 2000). DOI: 10.1007/3-540-49430-8_3.

[33] R. A. Schoonhoven, B. van Werkhoven, and K. J. Batenburg. “Benchmarking Optimization Algorithms for Auto-Tuning GPU Kernels”. In: *Ieee Transactions on Evolutionary Computation* 27.3 (2023). Netherlands Organization for Scientific Research (NWO) [639.073.506]; Dutch Research Council (NWO) [NWA.1160.18.316] 7, pp. 550–564. ISSN: 1089-778X. DOI: 10.1109/TEVC.2022.3210654. URL: https://ieeexplore.ieee.org/document/9905732/.

[34] Y. Sim, W. Shin, and S. Lee. "Automated code transformation for distributed training of TensorFlow deep learning models". In: *Science of Computer Programming* 242 (2025). ISSN: 0167-6423. DOI: 10.1016/j.scico.2024.103260. URL: https://www.sciencedirect.com/science/article/abs/pii/S0167642324001837?via%3Dihub.

[35] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

[36] Samuel L. Smith et al. *Don't Decay the Learning Rate, Increase the Batch Size*. 2018. arXiv: 1711.00489 [cs.LG]. URL: https://arxiv.org/abs/1711.00489.

[37] Rahul Srinivasan et al. "Simulation-based population inference of LISA's Galactic binaries: Bypassing the global fit". In: *Physical Review D* 112.10 (Nov. 2025). ISSN: 2470-0029. DOI: 10.1103/shym-w46f. URL: http://dx.doi.org/10.1103/shym-w46f.

[38] P. Stpiczynski. "Vectorized algorithm for multidimensional Monte Carlo integration on modern GPU, CPU and MIC architectures". In: *Journal of Supercomputing* 74.2 (2018), pp. 936–952. ISSN: 0920-8542. DOI: 10.1007/s11227-017-2172-x. URL: https://link.springer.com/content/pdf/10.1007/s11227-017-2172-x.pdf.

[39] S. Syed et al. "Non-reversible parallel tempering: A scalable highly parallel MCMC scheme". In: *Journal of the Royal Statistical Society Series B-Statistical Methodology* 84.2 (2022). National Science and Engineering Research Council; Michael Smith Foundation RRF Grant; Engineering and Physical Sciences Research Council [CoSInES EP/R034710/1]; EPSRC [EP/R034710/1, EP/R018561/1] Funding Source: UKRI; Engineering and Physical Sciences Research Council [EP/R018561/1, EP/R034710/1] Funding Source: researchfish 48, pp. 321–350. ISSN: 1369-7412. DOI: 10.1111/rssb.12464.

[40] N. P. Tran, M. Lee, and J. Choi. "Parameter based tuning model for optimizing performance on GPU". In: *Cluster Computing-the Journal of Networks Software Tools and Applications* 20.3 (2017), pp. 2133–2142. ISSN: 1386-7857. DOI: 10.1007/s10586-017-1003-4. URL: https://link.springer.com/article/10.1007/s10586-017-1003-4.

[41] M. Vallero, P. Rech, and F. Vella. "State of practice: Evaluating GPU performance of state vector and tensor network methods". In: *Future Generation Computer Systems-the International Journal of Escience* 174 (2026). ISSN: 0167-739X. DOI: 10.1016/j.future.2025.107927. URL: https://www.sciencedirect.com/science/article/pii/S0167739X25002225?via%3Dihub.

[42] He Wang and Liang Zeng. *Automated Algorithmic Discovery for Gravitational-Wave Detection Guided by LLM-Informed Evolutionary Monte Carlo Tree Search*. Electronic Article. Aug. 2025. DOI: 10.48550/arXiv.2508.03661. URL: https://ui.adsabs.harvard.edu/abs/2025arXiv250803661W%20https://arxiv.org/pdf/2508.03661.

[43] L. Wang et al. "LADDER: Enabling Efficient Low-Precision Deep Learning Computing through Hardware-aware Tensor Transformation". In: *Proceedings of the 18th Usenix Symposium on Operating Systems Design and Implementation, Osdi 2024* (2024), pp. 307–323.

[44] Z. Y. Wang et al. "Stitching Weight-Shared Deep Neural Networks for Efficient Multi-task Inference on GPU". In: *2022 19th Annual Ieee International Conference on Sensing, Communication, and Networking (Secon)* (2022), pp. 145–153. ISSN: 2473-0440. DOI: 10.1109/SECON55815.2022.9918563. URL: https://ieeexplore.ieee.org/document/9918563/.

[45] Kyrylo Yemets, Ivan Izonin, and Ivanna Dronyuk. “Enhancing the FFT-LSTM Time-Series Forecasting Model via a Novel FFT-Based Feature Extraction–Extension Scheme”. In: *Big Data and Cognitive Computing* 9 (Feb. 2025), p. 35. DOI: 10.3390/bdcc9020035.

[46] D. L. Zhuang et al. “MonoNN: Enabling a New Monolithic Optimization Space for Neural Network Inference Tasks on Modern GPU-Centric Architectures”. In: *Proceedings of the 18th Usenix Symposium on Operating Systems Design and Implementation, Osdi 2024* (2024), pp. 989–1005.

[47] Liu Ziyin, Tilman Hartwig, and Masahito Ueda. *Neural Networks Fail to Learn Periodic Functions and How to Fix It*. 2020. arXiv: 2006.08195 [cs.LG]. URL: https://arxiv.org/abs/2006.08195.

Appendix A

Figures

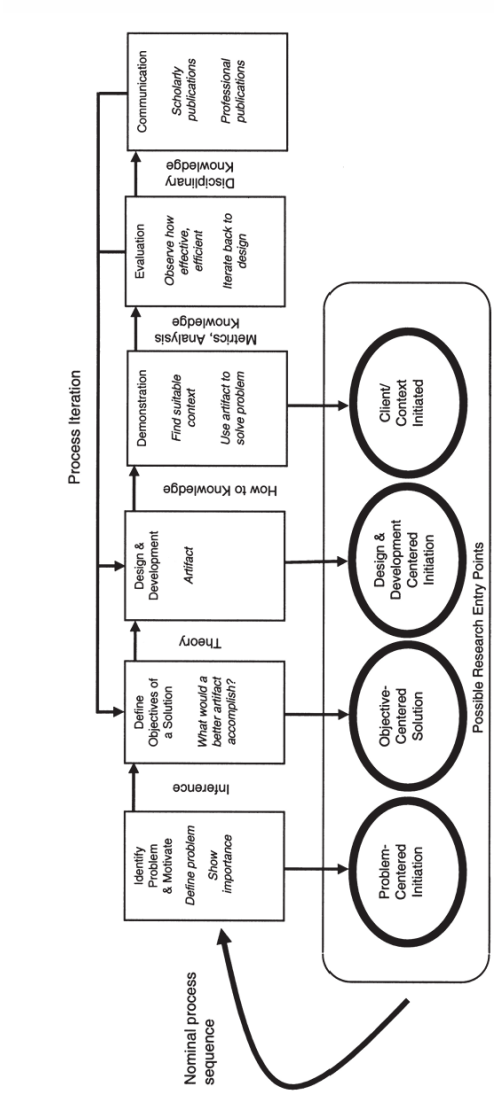


Figure A.1: DSRM Process Model [5]