

**interTwin**

# **D4.8 Final version of the DT capabilities for High Energy Physics, Radio astronomy and Gravitational-wave Astrophysics including validation reports**

**Status: Under EC Review**  
**Dissemination Level: public**



**Funded by the  
European Union**


**Disclaimer:** Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them

## Abstract

### Key Words

Digital Twin Engine, DT capabilities, High Energy Physics, GW Astrophysics, Radio Astronomy

This deliverable describes the release of Digital Twin (DT) applications that support the physics use cases. It details the capabilities, characteristics, and describes the functional specifications of the DT applications and the integration into the DTE architecture. Finally, it provides information about the final validation of the developed and integrated DTs.

Document Description			
D4.8 Final version of the DT capabilities for High Energy Physics, Radio astronomy and Gravitational-wave Astrophysics including validation reports			
Work Package 4			
Document type	Deliverable		
Document status	Under EC Review	Version	1
Dissemination Level	Public		
Copyright Status	 <p>This material by Parties of the interTwin Consortium is licensed under a <a href="https://creativecommons.org/licenses/by/4.0/">Creative Commons Attribution 4.0 International License</a>.</p>		
Lead Partner	CERN		
Document link	<a href="https://documents.egi.eu/document/3944">https://documents.egi.eu/document/3944</a>		
DOI	<a href="https://zenodo.org/records/17093659">https://zenodo.org/records/17093659</a>		
Author(s)	<ul style="list-style-type: none"> <li>• Yurii Pidopryhora (MPG)</li> <li>• Kalliopi Tsolaki (CERN)</li> <li>• Vera Maiboroda (CNRS)</li> <li>• Sofia Vallecorsa (CERN)</li> <li>• Javad Komijani (ETHZ)</li> <li>• Marina Marinkovic (ETHZ)</li> <li>• Gaurav Sinha Ray (CSIC)</li> <li>• Isabel Campos (CSIC)</li> <li>• Lorenzo Asprea (INFN)</li> <li>• Francesco Sarandrea (INFN)</li> <li>• Sara Vallero (INFN)</li> <li>• Federica Legger (INFN)</li> </ul>		
Reviewers	<ul style="list-style-type: none"> <li>• Andrea Cristofori (EGI Foundation)</li> <li>• Dijana Vrbanc (DESY)</li> </ul>		
Moderated by	<ul style="list-style-type: none"> <li>• Andrea Anzanello</li> </ul>		
Approved by	<ul style="list-style-type: none"> <li>• Andrea Manzi (EGI Foundation) on behalf of TCB</li> </ul>		

Revision History			
Version	Date	Description	Contributors
V0.1	09/07/2025	Template creation from the 1st version of the DTs capabilities	Andrea Cristofori (EGI Foundation)
V0.2	05/08/2025	All contributions added	All the authors
V0.3	15/08/2025	Internal review	Andrea Cristofori (EGI Foundation), Dijana Vrbaneć (DESY)
V0.4	28/08/2025	TCB review	Andrea Manzi (EGI Foundation)
<b>V1.0</b>		<b>Final</b>	

Terminology / Acronyms	
Term/Acronym	Definition
ANNALISA	Advanced Nonlinear Transient-Noise Analyser of Laser Interferometer Sensor Arrays
CNN	Convolutional Neural Network
CQT	Constant Q-Transform
DT	Digital Twin
DTE	Digital Twin Engine
ESS	Effective Sample Size
GAN	Generative Adversarial Networks
GAN	Generative Adversarial Network
GW	Gravitational Wave
HC-LHC	High Luminosity - Large Hadron Collider
HDF5	Hierarchical Data Format version 5
HEP	High Energy Physics
HPC	High Performance Compute
HPO	Hyper Parameter Optimization
ILDG	International Lattice Data Grid
ILDG-FC	International Lattice Data Grid File Catalogue
ILDG-MC	International Lattice Data Grid Metadata Catalogue
LQCD	Lattice QCD
MC	Monte Carlo
MCMC	Markov Chain Monte Carlo



ML	Machine Learning
ML-PPA	Machine Learning-based Pipeline for Pulsar Analysis
NF	Normalising Flow
ONNX	Open Neural Network Exchange
PoC	Proof of Concept
QCD	Quantum Chromodynamics
QED	Quantum Electrodynamics
RFI	Radio-Frequency Interference
RSE	Rucio Storage Element
SNR	Signal-to-Noise-Ratio
SQAaaS	Software Quality Assurance as a Service
TB	Terabyte
WP	Work Package

## Table of Contents

<b>1</b>	<b>Introduction.....</b>	<b>9</b>
1.1	Aim of this deliverable .....	9
1.2	Intended audience of this document .....	9
1.3	Structure of the document .....	9
<b>2</b>	<b>DTs Application.....</b>	<b>10</b>
2.1	DT Application: Lattice QCD simulation .....	10
2.2	DT Application: Detector simulation .....	11
2.3	DT Application: Noise simulation for radio astronomy .....	14
2.4	DT Application: VIRGO Noise detector .....	17
<b>3</b>	<b>DTs Application Final release and Validation .....</b>	<b>20</b>
3.1	DT Application: Lattice QCD simulation .....	20
3.1.1	DTs Application Integrations .....	21
3.1.2	Scope and limitations.....	23
3.1.3	Validation and Results .....	24
3.2	DT Application: Detector simulation .....	25
3.2.1	DTs Application Integrations .....	26
3.2.2	Scope and limitations.....	28
3.2.3	Validation and Results .....	29
3.3	DT Application: Noise simulation for radio astronomy .....	29
3.3.1	DTs Application Integrations .....	30
3.3.2	Scope and limitations.....	30
3.3.3	Validation and Results .....	32
3.4	DT Application: VIRGO Noise detector .....	32
3.4.1	DTs Application Integrations .....	33
3.4.2	Models and Data Requirements.....	34
3.4.3	Scope and limitations.....	35
3.4.4	Validation and Results .....	35
<b>4</b>	<b>Conclusions.....</b>	<b>38</b>
<b>5</b>	<b>References .....</b>	<b>39</b>

## List of Figures

Figure 1	Mind map of the interconnections between the Lattice Use case and the core services and tools within interTwin, and with the ILDG 2.0 initiative .....	10
Figure 2	Detailed graph representation of the training and inference workflows composition of the fast calorimeter detector simulation DT utilising 3DGAN or CaloINN approach .....	13
Figure 3	Operation Diagram of the Machine Learning-based Pipeline for Pulsar Analysis (ML-PPA) .....	14



Figure 4 Examples of four main types of data time-frequency frames in a pulsar-observation data set, visualised as 256x256 images.....	16
Figure 5 A sketch of the vetoing algorithm. ....	17
Figure 6 The example of how one individual glitch is processed and turned into two separate spectrograms. ....	18
Figure 7 Block diagram for the method of normalizing flows. ....	21
Figure 8 Module Integration Diagram for the Lattice QCD use case.....	22
Figure 9 Schema depicting the flow of control when accessing the Lattice Datalake. ....	23
Figure 10 The Effective Sample Size (ESS) of a ML model measured every 10 epochs during the training of an SU(3) gauge theory with $\beta=1$ on a 4x4x4x4 lattice [R22].....	25
Figure 11 High-level workflow composition of the fast particle detector simulation DT using ML techniques, and its connections with components from other work packages.....	26
Figure 12 General outline of the DT structure.....	30
Figure 13 Layered software architecture of the framework ML-PPA.....	31
Figure 14 Diagram of the final software product of ML-PPA (in the C4 model).....	32
Figure 15 System Context diagram of the DT for the veto/de-noising pipeline.....	34
Figure 16 The portions of the data which are correctly de-noised and cleaned, as a function of the threshold that we set to define what constitutes a glitch. ....	36
Figure 17 An example of a spectrogram containing a Scattered Light glitch that is correctly generated by the DT and then subtracted from the real data strain. ....	37

## Executive summary

This document is deliverable 4.8 of the interTwin project, part of work package 4. It is a report collectively written by the partners of tasks 4.1, 4.2, 4.3 and 4.4, who are directly involved in designing digital twins for the physics domain (High Energy Physics, Radio Astronomy, and Gravitational Waves Astrophysics). The objective of this report is to provide an overview of the functionalities offered by each of the Digital Twins (DTs) and their integration with the Digital Twin Engine (DTE).

It starts by outlining the capabilities of each DT application in general, demonstrating how they support their use cases and highlighting the specific features. Further description reviews the final release of each DT application, including the integration between each DT application and the main Core Components, as well as their integration with the Infrastructure Components. Validation and results are also discussed.

In general, every DT is uniquely crafted to target the particular task, offering new scientific tools that can be extremely helpful in the corresponding areas of physics.



# 1 Introduction

## 1.1 Aim of this deliverable

The overall objective of deliverable 4.8 is to provide information about the capabilities of DT Applications related to physical sciences (T4.1, T4.2, T4.3, T4.4). A Digital Twin application is a user-facing implementation of a DT. DT applications are the consumers of the capabilities offered by the interTwin DTE, thus introducing use case-specific requirements.

## 1.2 Intended audience of this document

The main audiences for deliverable 4.8 are the **developers** and **end users**.

For the DT Application and DTE **developers**: this deliverable tells them about implemented features of each of the DTs application, including interaction and integration with the underlying DTE modules. The DTE developers can then better understand the requirements of each use case, while for the DT Applications developers it serves as a reference. It gives insight into both categories on how the DTs fit into the overall interTwin architecture by using specific Core Components.

For **end users**: the specified deliverable provides information on what are the capabilities of each specific DT in the physics domain. The users can find out what each DT is intended for, including the configurations, limitations, and possible customisations to meet the individual specific needs. By establishing a common framework for communication, stakeholders will be able to exchange information, validate models, and collaboratively address their challenges.

## 1.3 Structure of the document

The structure of this deliverable is as follows. **Section 2** gives an overview of each DT application. **Section 3** describes each DT application development. Finally, **Section 4** provides the conclusions.

## 2 DTs Application

### 2.1 DT Application: Lattice QCD simulation

As a statistical tool, Lattice QCD (LQCD) has been successfully used by physicists for decades to investigate and evaluate the Standard Model of Particle Physics. It has been used to determine the quark masses and the strength of the Strong nuclear interaction with steadily improving accuracy. However, despite LQCD's many successes and the availability of ever more powerful computational resources, the limitations of current simulation algorithms have led to problems such as critical slowing down on lattices with physically interesting properties.

The purpose of Task 4.1 has been the development of a DT application that can more efficiently simulate quantum field theories on a lattice. It also seeks, in work done jointly with Task 7.1, to improve existing lattice DTs by integrating and tailoring services and tools being developed within interTwin into their existing workflows (see **Figure 1**). The two parallel (and complementary) paths being explored involve:

- Developing Machine Learning based lattice simulations via the use of Normalising Flows (NF).
- Improving data management and software workflows for conventional lattice simulations.

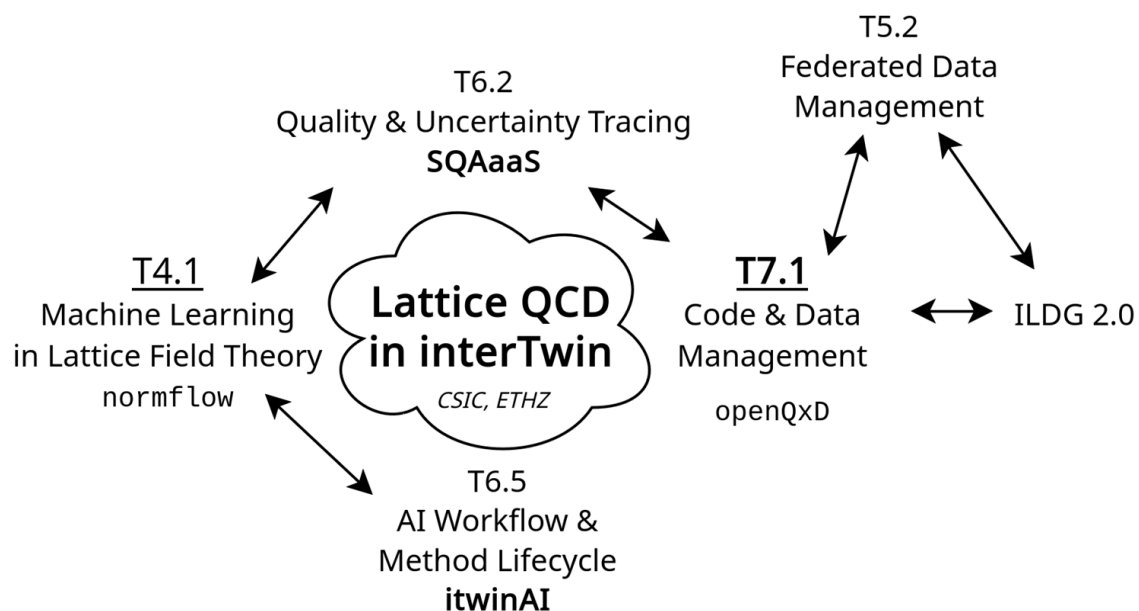


Figure 1 Mind map of the interconnections between the Lattice Use case and the core services and tools within interTwin, and with the ILDG 2.0 initiative

The **openQxD**<sup>1</sup> application is a good example of an already existing lattice DT used for the competitive simulations needed to study particle physics. It is a program written in C for simulating QCD+QED field theories on 4D spacetime lattices. It lets the user configure, according to their research goals, a large number of physics and solver parameters, and is optimised to run at scale on large HPC systems. It takes as both input and output large gauge field ensembles that have to be stored on an easily accessible storage.

Machine learning algorithms provide an alternative approach to lattice simulations and early results suggested that they could address some of the difficulties encountered with Markov Chain Monte Carlo (MCMC) studies [R1]. Through the development of the **normflow** software package we have been exploring deep generative models, in particular NF, in order to study alternatives to the standard algorithms used for generating lattice field configurations. We have been able to train ML models, up to a given accuracy, against scalar and gauge lattice field theories before using the models to generate lattice field configurations. These configurations are the basic research data used for most lattice research. The total cost in computing resources, primarily from training the model, is greater than that of the traditional highly optimised MCMC methods. As the models and methods used improve beyond the proof-of-concept stage the training cost will decrease and become comparable to or, ideally, cheaper than the cost of field generation using traditional methods.

Task 4.1 has explored the benefits to a typical lattice collaboration of the federated data capabilities, in particular the so-called “data lakes”, enabled by the tools developed within the interTwin project WP5 [R24]. The datalake framework allows one to use the International Lattice Data Grid (ILDG)’s Indigo IAM<sup>2</sup> as an Identity Provider (IdP). Instead of registering with multiple HPC centres, a lattice researcher only needs to be registered with the ILDG to access a “Lattice Datalake” that contains data uploaded by members of the lattice community to many different HPC centres. Using ILDG’s Oauth2 token-based authorisation also affords administrators more control over per-user permissions, this is vital given the size and international nature of lattice collaborations. Using the ILDG’s Indigo IAM as an IdP should also ease future integration with the ILDG-FC and ILDG-MDC, which are expected to grow to be the premier repositories for raw lattice data in the coming years.

## 2.2 DT Application: Detector simulation

Task 4.2 concerns the design and development of a DT application for particle detector simulation. The requirements and functionalities of such a DT application have been described in detail in D4.6 [R4] and D7.8 [R2]. In the present deliverable section, the DT’s goals and interface will be revised and updated.

Simulations in particle physics are needed to compare theoretical model predictions with experimental data. As the amount of experimental data increases, more simulated data needs to be produced. In experiments like those at the Large Hadron Collider, where

---

<sup>1</sup> <https://gitlab.com/rcstar/openQxD>

<sup>2</sup> <https://iam-ildg.cloud.cnaf.infn.it>



large amounts of data are collected, optimizing computational resources for simulations is crucial. The most computationally expensive step is modeling particle interactions with detector materials, especially in calorimeter detectors. In collider experiments, calorimeters are responsible for measuring the energy of the particles emerging after particle beams collisions. Those emerging particles travel through the detector and interact with the detector material through the fundamental forces. In particular, within electromagnetic or hadronic calorimeters, showers of secondary particles are created due to the interaction of each new particle with the dense calorimeter material [R5]. The secondary particle creation process is inherently a complex stochastic process, and it is typically modelled using Monte Carlo (MC) techniques. These simulations have a crucial role in High Energy Physics (HEP) experiments, and at the same time are very resource-intensive from a computing perspective.

The HEP community has long since started developing faster alternatives to Monte Carlo, including deep learning based techniques [R5, R6, R7]. In the calorimeter case, deep learning based fast simulation directly generates the detector output, without reproducing, step by step, each single particle that interacts with the detector material. More specifically, generative models have been used in related HEP applications, as they are able to combine deep learning with statistical inference and probabilistic modelling. A generative model's goal is to learn how to generate data based on a given unknown distribution describing a finite number of observations.

The detector simulation DT accelerates particle detector simulations, leveraging generative AI methods. Our methodology is leveraging Geant4<sup>3</sup>, a software toolkit for the simulation of the passage of particles through matter, to produce detector output data that then a Generative Adversarial Network (GAN), a class of machine learning frameworks for approaching generative AI, can be trained on. The technical requirements of such a DT have been identified, defined, and reported in detail in D7.8 [R2].

The DT includes two different generative models for fast calorimeter response simulations, 3DGAN and CaloINN.

Calorimeter detectors can be regarded as huge cameras taking pictures of particle interactions [R9], and a 3D convolutional GAN (3DGAN) model works with calorimeter response represented as 3D images. The voxels (3D calorimeter cells) are generated as monochromatic pixelated images with the pixel intensities representing the cell energy depositions. The GAN approach was shown to be able to generate highly realistic and sharp images [R8].

CaloINN model [R10] also represents calorimeter response as energy deposits in detector volume segmented into layers but does not use the spatial correlations of image representation. CaloINN is a NF model [R11] implementing a sequence of invertible layers to learn a transformation from a simple known distribution to a complex distribution of calorimeter output.

---

<sup>3</sup> <https://geant4.web.cern.ch/>



#### D4.8 Final version of the DT capabilities for High Energy Physics, Radio astronomy and Gravitational-wave Astrophysics including validation reports

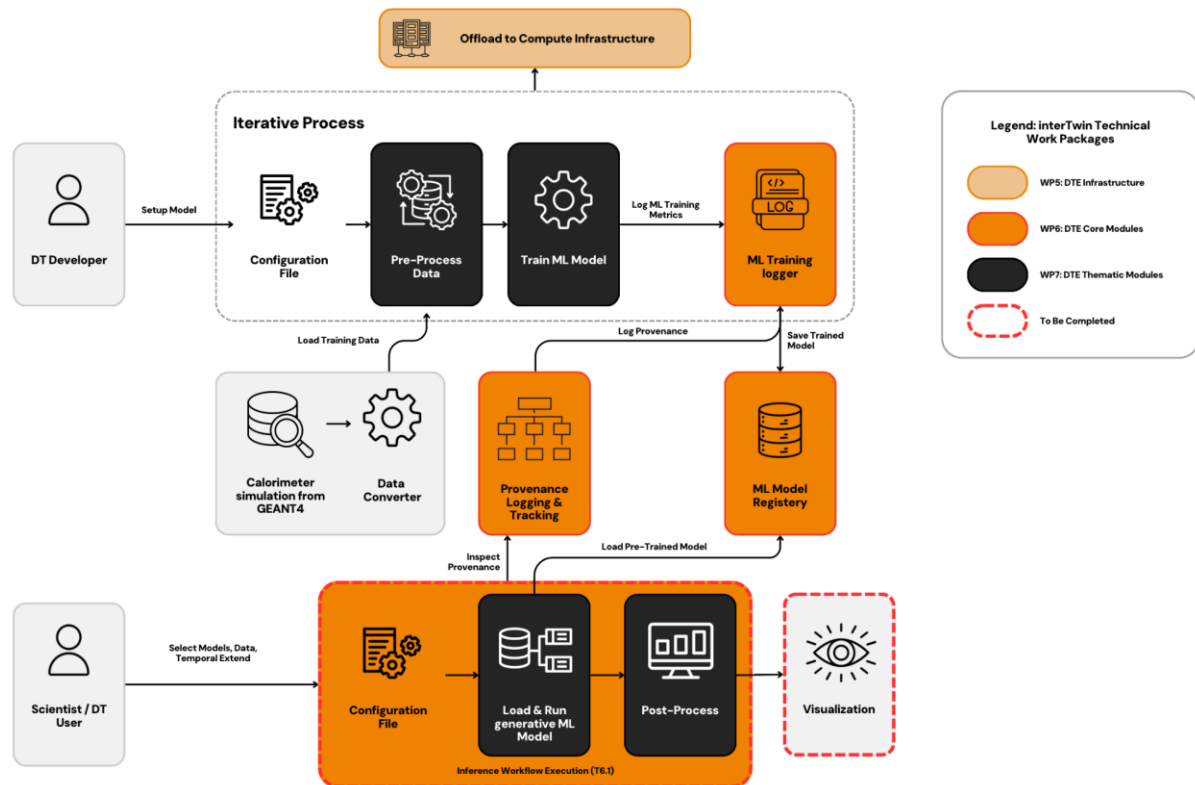


Figure 2 Detailed graph representation of the training and inference workflows composition of the fast calorimeter detector simulation DT utilising 3DGAN or CaloINN approach

Despite the difference in the approaches used by 3DGAN and CaloINN to simulate the calorimeter response, both models share the same workflows. A brief overview of the designed capabilities of the calorimeter detector simulation DT follows.

The DT operator is leveraging the Geant4 framework to produce data representing particles passing through a specific detector setup by employing MC-based simulations. The completion of this task is marked by the system's ability to successfully simulate particle passage through the specified detector setup and subsequently generate and store the simulation data for future use. Then, the data scientist's role involves preprocessing the simulated data to make it suitable for training the 3DGAN or CaloINN model. This process is crucial to ensure that the data can effectively train the model. The requirement is met when the data scientist has access to the raw simulation data, and the system supports all necessary preprocessing and preparation steps (changing the data format, extracting information of interest, making necessary selection on energy deposition in calorimeter cells), making the pre-processed data ready for model training.

Subsequently, the DT user is responsible for training the 3DGAN or CaloINN model on the pre-processed simulated data. Specific conditions such as the initial energy, and type need to be considered; additionally, 3DGAN requires the particle's entrance angle. The aim is to create a model that can reproduce data similar to the original simulated data. The process is considered successful if it can effectively use the pre-processed data, with the DT providing adequate tools for monitoring and tuning the training process, and the system validating the trained model through performance metrics.

Finally, the user can leverage the trained model during inference steps to facilitate fast AI-based simulation. The objective is to produce simulation data faster compared to

traditional Geant4 simulations. Success in this area is defined by the system's capability to generate accurate AI-based simulation data under specific initial conditions, and its comparison for consistency and speed against traditional data. A more comprehensive and detailed description of the aforementioned DT processes can be found in the deliverable D4.6 [R4] and seen in **Figure 2**.

## 2.3 DT Application: Noise simulation for radio astronomy

Within Task 4.3 a DT of an astronomical source-telescope system is developed, able to generate synthetic output signals identical to the data recorded by a real telescope, including both scientifically valuable data and various interference and noise signals. Together with Task 7.2, its goal is the creation of a larger framework called ML-PPA (Machine Learning-based Pipeline for Pulsar Analysis, see **Figure 3**) with a Convolutional Neural Network (CNN)-based ML classifier of the pulsar data, which can be trained using the DT-supplied data. The requirements and functionalities for these applications have already been described in D4.6 [R4] and D7.8 [R2]. A 50-page document [R14] has been written by our team, giving a detailed overview and background of the project. Here we only provide a brief overview.

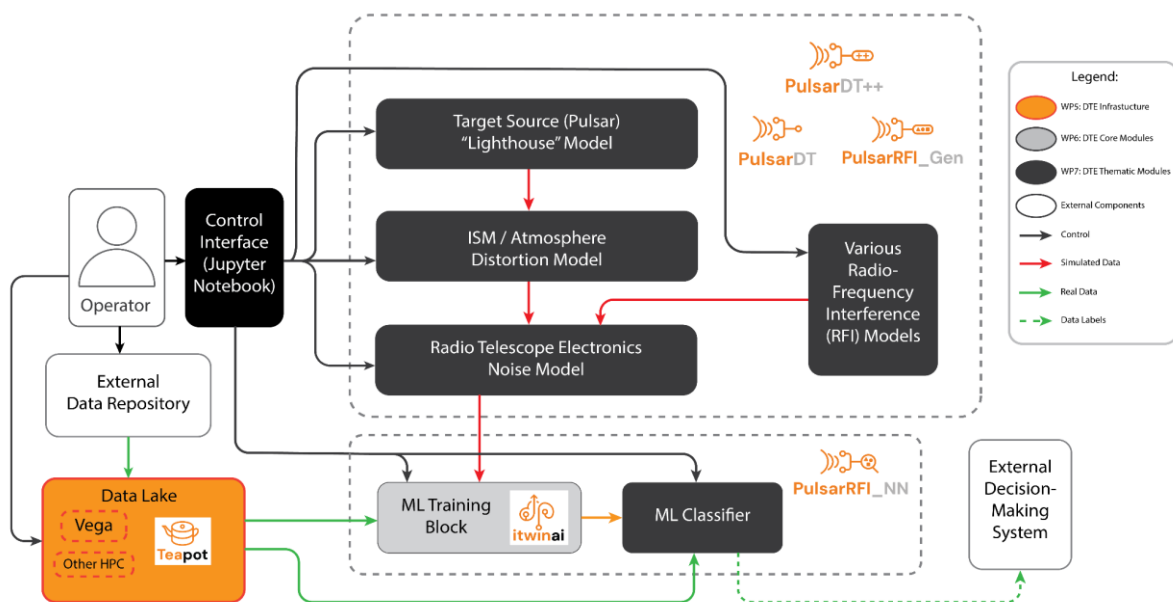


Figure 3 Operation Diagram of the Machine Learning-based Pipeline for Pulsar Analysis (ML-PPA)

The main motivation for this work comes from the need to solve the problem of data overflow, which is about to become one of the largest issues in modern observational astronomy in general and radio astronomy in particular. The traditional paradigm of data reduction is not sufficient anymore and with the new generation of telescopes generating larger amounts of data, storing everything long term to allow inspection and analysis by human experts is no longer feasible. The Square Kilometre Array (SKA)<sup>4</sup> "pathfinders",

<sup>4</sup> SKAO: <https://www.skao.int/en>



such as South African MeerKAT<sup>5</sup> or Australian ASKAP<sup>6</sup> can produce several petabytes of raw data per week, thus it strains the capacity of even the best scientific facilities to keep their datasets even for weeks. When the SKA itself will be operational in a few years, the data flood will increase even further.

One way to deal with this is a ML-based data classification pipeline, able to both sort-out all the scientifically insignificant data in real time and alert human operators immediately when something relevant is found. The ML-PPA developed within Tasks 4.3 and 7.2 is a pilot for just such a system.

An additional advantage of utilising an automated expert system like this is that it can react to unique events quickly enough for the new observational resources to be tasked with thoroughly studying them. This would allow it to systematically probe the transient radio sky, which currently is generally unknown. Transients, meaning spikes of radiation that appear unpredictably in random directions, may result from very far and enormously energetic exotic events (like black hole collisions) and thus are invaluable sources of information in the areas of physics that cannot be studied experimentally in any other way, e.g. quantum gravity. If a signature of a transient source is detected in the data flow, the automated expert system could trigger the “target of opportunity” alert for the anomaly, and notify the scientists on duty, who can decide on the best course for further action. In turn, this may lead to a concerted effort of observing the target by a number of instruments, e.g. combining Earth-based radio observations with space-based optical and X-ray ones. An incredibly rare event, which would otherwise be only noticed later while reviewing the data record, can therefore be studied in much more detail and with all the available tools.

To be able to detect special and important events in the data, one first has to understand the regular and common features of the data stream well. In radio astronomy this primarily means noise and Radio-Frequency Interference (RFI). Thus, a very important aspect of this project is understanding and modelling noise and RFI patterns in radio-astronomical data.

High data rates also require application of HPC in processing; however, current common radio astronomy software is not well suited to this as it handles parallelization poorly. Another goal of this project is to create a well-scalable pipeline and test it with supercomputers.

Pulsars are ideal test subjects for this task, since they reliably produce periodic bursts of scientifically significant data with certain variability in signal strength and other parameters. ML-PPA has been tested with real data collected by observing various pulsars with two telescopes: the Effelsberg 100m radio telescope<sup>7</sup> and the above-mentioned MeerKAT array.

---

<sup>5</sup> MeerKAT Radio Telescope: <https://www.sarao.ac.za/gallery/meerkat/>

<sup>6</sup> ASKAP-radio telescope: <https://www.csiro.au/en/about/facilities-collections/atnf/askap-radio-telescope>

<sup>7</sup> Radio Telescope Effelsberg: <https://www.mpifr-bonn.mpg.de/en/effelsberg>



Before the analysis, each set is broken into a sequence of square time-frequency frames, e.g. 256-time samples by 256 frequency samples (**Figure 4**). The ML data classification tool assigns labels to each data fragment, i.e. a time-frequency frame, based on the type of signal detected or not detected in it. The label describes the fragment on a basic level as “scientifically important data”, “no signal”, “interference of such and such type” etc., and in the future may also include more detailed properties.

Since the proportions of each data type in the real data flow are very different (e.g. scientifically important data might constitute much less than 1% of the data sample), efficient ML training requires synthetic data to be used. That is where the DT comes in, a physical model of the source, its signal transmission and registration.

Both the DT and ML-classifier were first developed in Python, but ultimately implemented in C++, ensuring high computational speed and scalability.

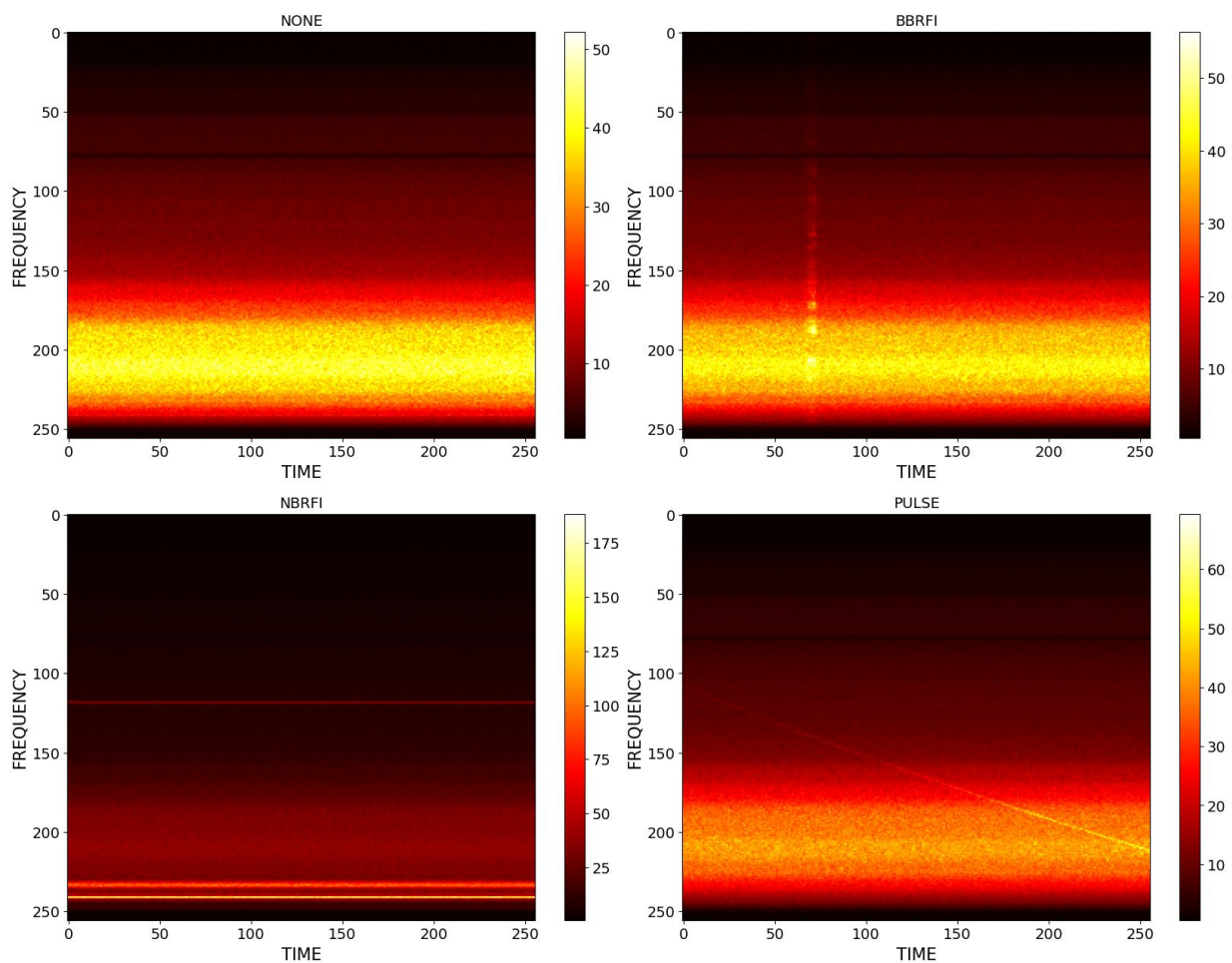


Figure 4 Examples of four main types of data time-frequency frames in a pulsar-observation data set, visualised as 256x256 images. The vertical axis corresponds to frequency, horizontal to time, the value of each point is signal intensity. The top left frame is the most common one, “none” or “empty”, it contains only noise, amplified differently because the telescope’s sensitivity is different for different frequencies, leading to apparent horizontal banding; the top right and bottom left frames contain broad-band (BB) and narrow-band (NB) radio-frequency interference (RFI) represented by brighter vertical or horizontal stripes, these types of data are undesirable but often recorded because the telescope is sensitive to various artificial or natural electro-magnetic phenomena (radio transmissions, emissions by various devices, electric storms etc.). Finally, the bottom-right frame contains a pulsar’s pulse, represented by the diagonal slightly curved line; this is the only desirable type of data frame that we would like to separate from all the other types.



## 2.4 DT Application: VIRGO Noise detector

The purpose of Task 4.4 was to develop a DT application that could simulate glitches in the observational channel of the Virgo Gravitational Wave (GW) interferometer<sup>8</sup> from its control channels. A glitch is a transient noise artefact observed in the observational, *strain*, channel. Glitches can occur due to environmental reasons, such as seismic activity, or as a result of resonances in the detector's subsystems. The monitoring of such effects is entrusted to a high number of control sensors, whose continuous data output is mapped onto so-called *auxiliary channels*. Glitches are divided into the classes identified by the GravitySpy project [R15].

The high number of observed glitches in the past observational runs and in the current O4<sup>9</sup> run makes it necessary to put into place effective vetoing and de-noising procedures (Figure 5) for the identification and removal of bad data from the analyses' pipelines. For this reason, the project aimed at identifying which auxiliary channels correlate strongly with the glitches observed in the strain and use generative methods to map the glitch from the former to the latter.

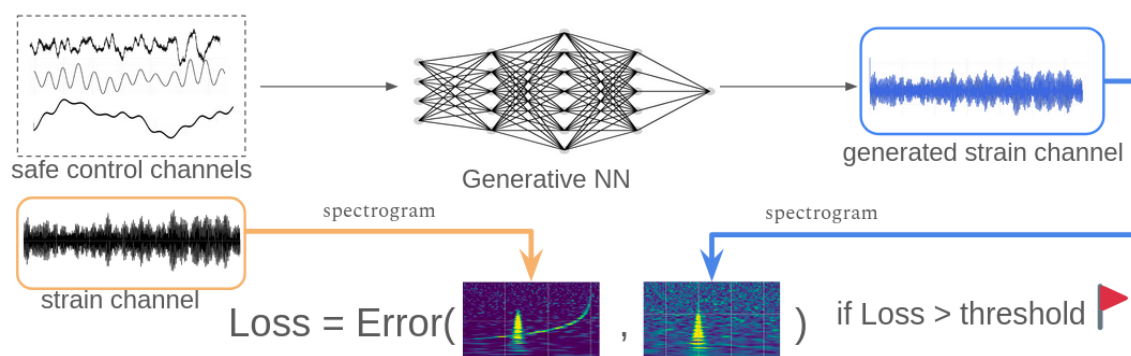


Figure 5 A sketch of the vetoing algorithm. The safe control channels are taken as input by the generative NN, which then outputs the generated glitch. The loss chosen for the training is a simple L1 distance.

The final implementation of the DT consists of a train and inference subsystems implemented as itwinai<sup>10</sup> plugins.

The main components of the training subsystem of the DT are **ANNALISA**, **Preprocess\_API**, and **GlitchFlow**. These modules are respectively used to identify the relevant auxiliary channels for input to the Neural Network (NN), to create a dataset of two-dimensional spectrograms from the input data, and to generate glitches in the strain channel based on these spectrograms.

The inference subsystem also uses **Preprocess\_API**, with the addition of **Generative\_API**, which is used for the glitch generation.

In this release we present the final versions of the modules mentioned above, some of which were already described in D4.6 [R4].

<sup>8</sup> <https://www.virgo-gw.eu/>

<sup>9</sup> Ligo, Virgo and Kagra observing runs: <https://observing.docs.ligo.org/plan/>

<sup>10</sup> itwinai: <https://www.intertwin.eu/article/core-dte-module-itwinai>

One significant addition in this final version was the introduction of spectrograms encoding the *phases* of the signals. Since the spectrograms consist of complex-valued matrices, each of them can be split into a spectrogram which represents the amplitude of the transformed signal and one that represents the phase, as shown in **Figure 6**.

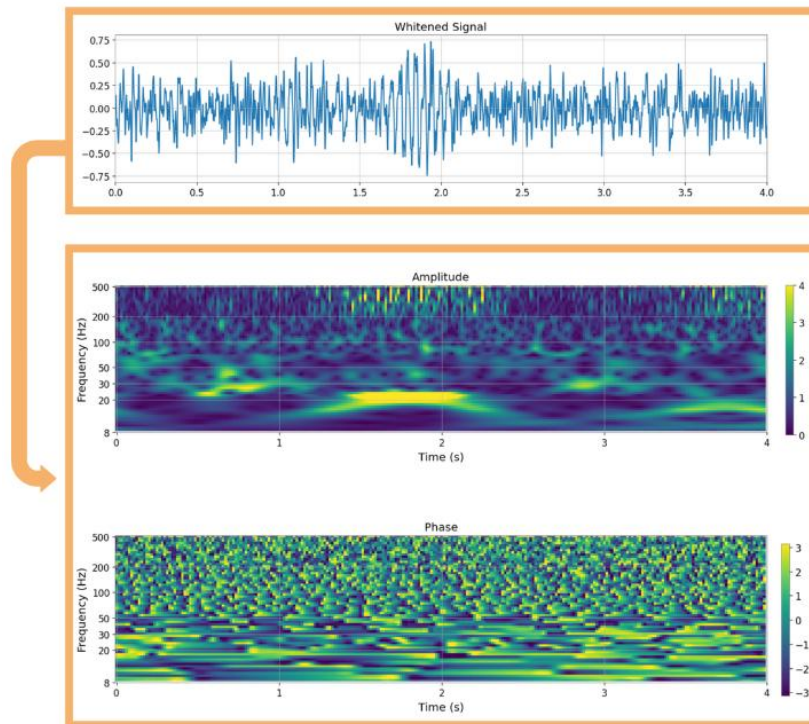


Figure 6 The example of how one individual glitch is processed and turned into two separate spectrograms. The phase spectrogram contains values that range from  $-\pi$  to  $\pi$ , and they represent the radians of the phase angles.

In this new approach, every time series is represented by two spectrograms; this doubles the total amount of data that needs to be stored and processed, but it also increases the information that we give as input to the Generative Neural Network (GNN). We found that this approach significantly improved the performance of the DT, as shown in **Section 3.4.4**.

As the aim of the project is to implement a vetoing and de-noising protocol to flag portions of the datastream as corrupted, it is of paramount importance to avoid the erroneous characterisation of GWs as glitches. The only way to achieve this is to provide as input to the NN only channels which do not detect any astrophysical signals; these channels are termed *safe channels*. A list of safe channels is available within the Virgo collaboration. This classification is constantly being updated for each new observational run, as channels are replaced, or their technical specifications are changed.

With tens of thousands safe channels in Virgo, feeding all of them into our analysis tool would be unfeasible. Instead, we leverage our understanding of specific glitch phenomena to identify relevant input channels. For example, scattered light glitches — currently the most frequently observed type — occur due to microseismic activity coupling with the detector, leading to photon scattering off moving reflective surfaces. Because these glitches originate from physical causes and are detected by sensors



measuring the optical bench's movement, velocity, or acceleration, we can select relevant input channels based on their corresponding units. We also consider channels monitoring the global status of the interferometer's optical components. This approach ensures a more precise and effective channel selection process and allows us to restrict the correlation scan to a few hundreds of auxiliary channels. This selection of channels is then fed as input to ANNALISA.

**ANNALISA** ("Advanced Nonlinear transient-Noise Analyser of Laser Interferometer Sensor Arrays") is a tool that makes use of time-frequency domain analysis of the data, namely the Constant Q-Transform (CQT), to evaluate correlations among the main and auxiliary channels as the ratio of temporally coincident spikes in the energetic content of the signals above a critical threshold over the total number of spikes in the main channel.

**Preprocess\_API** takes as input the data of the auxiliary channels selected by ANNALISA, in the form of one-dimensional time series. It then performs the operation of *whitening* of the data, consisting of a process that normalises the amplitude spectral density of the signal across all frequencies. It subsequently computes the CQT of the data, turning it into a series of two-dimensional spectrograms which are saved and used as input for **GlitchFlow**.

**GlitchFlow** is the module which contains the GNN for generating the glitches. The GNN model architecture in GlitchFlow is a U-Net inspired encoder-decoder that also incorporates attention gates and residual blocks.

GNNs, and generative models more generally, are a class of ML algorithms designed to learn and mimic the underlying data distribution of a given dataset. The main idea behind generative models is to generate new data points that are similar to those in the original dataset. A U-net is a CNN in which the input data goes through a series of pooling layers and subsequently a series of up-sampling ones, which restore its original resolution. The attention gates connect some pooling gates with their symmetric up-sampling ones, allowing the network to learn complex non-local features in the data.

The training process involves feeding the generative model in GlitchFlow with data from the auxiliary channels identified by ANNALISA and the corresponding glitches observed in the main channel. Through iterative training, the model learns to capture the complex relationships and dynamics present in the data, allowing it to generate realistic glitches that accurately reflect the characteristics of the observed glitches.

The channels selected by the ANNALISA module, i.e. those showing a correlation coefficient above a tenable threshold, are then passed to Preprocess\_API in order to create the dataset to train the GNNs in GlitchFlow for the task of glitch generation. Once trained the model is manually loaded as the generator in Generative\_API.

**Generative\_API** performs the actual inference on the data. It takes portions of data from the selected safe auxiliary channels and uses the trained GlitchFlow model to generate two-dimensional spectrograms. The generated spectrograms can then be subtracted from the real strain data to perform vetoing or de-noising.

## 3 DTs Application Final release and Validation

### 3.1 DT Application: Lattice QCD simulation

**Normflow** is a software package written in Python that uses the PyTorch library as its ML framework, with PyTorch functions modified and extended where required. For lattices of reasonable size, the training must be done on HPCs, either on CPUs or GPUs. The design of such a DT was described in Deliverable 4.6, section 3.1.2 [R4]. A technical specification about the resulting thematic module, **normflow**<sup>11</sup>, is given in D7.8, section 3.2 [R2].

Normflow is a proof of concept of a physics DT. It contains functions for the implementation of the NF method as a generative model for lattice field theories. The workflow is straightforward, the user simply:

- Chooses a field theory,
- selects the desired model and training parameters,
- trains the model on many CPUs or GPUs.

The key idea behind NF methods is to build and train a neural network that maps a theory of interest to one that is easier to simulate, ideally to a theory in which the degrees of freedom are decoupled. Complex probability distributions can be accurately modelled by transforming samples from a simple distribution via a series of invertible transformations. Once such a map is found, one can then efficiently draw many samples from the theory of interest. We have been working with a method that combines a NF method with aspects of the MCMC method in order to improve efficiency and reduce autocorrelations. The basic algorithmic structure we are employing is shown in **Figure 7**.

Normflow currently supports the modelling of scalar theories in any number of dimensions. We have extended the package to include gauge theories with the goal of the efficient modelling of SU(2) and SU(3) gauge theories in 4D spacetime. In order to achieve this, one should ideally use a network that respects the gauge symmetry. As previously reported, we are investigating networks that encode the necessary gauge-equivariant transformations. In addition, normflow now supports the saving and loading of models, thus improving the robustness and flexibility of the training procedure and allowing the end user to debug the model at an intermediate stage if they wish.

---

<sup>11</sup> <https://github.com/jkomijani/normflow>



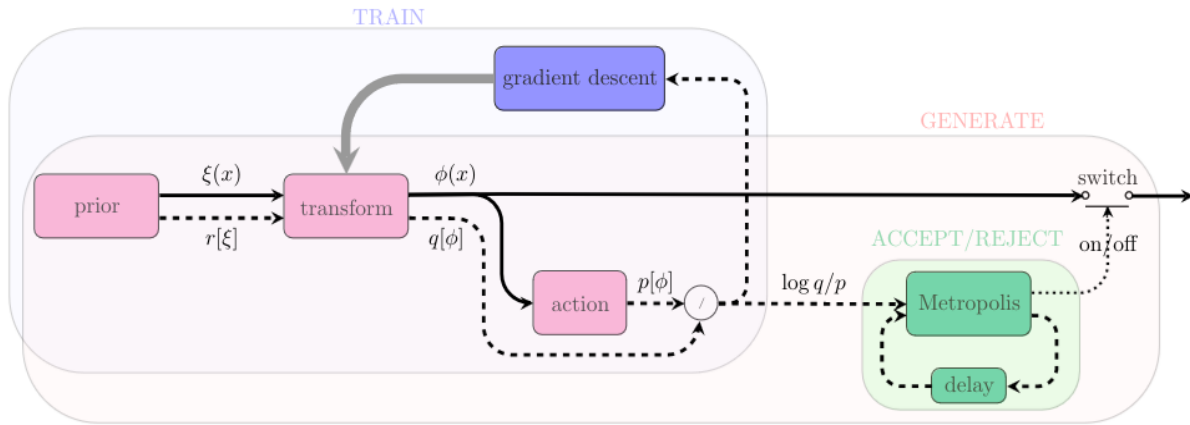


Figure 7 Block diagram for the method of normalizing flows.  $\xi$  and  $\phi$  are the prior and transformed fields at position  $x$ .  $r$  and  $q$  are the corresponding probability densities. The GENERATE block illustrates the integration of NF and MCMC by including an accept/reject step.

### 3.1.1 DTs Application Integrations

**Figure 8** summarises the typical workflow for DT Users and Developers. It also highlights the integrations between the Lattice Use Case and the services and tools provided by interTwin.

Improvements to the software development workflow were made with the integration of normflow and openQxD in the Software Quality Assurance as a Service (SQaaS) developed in WP6. Software quality assurance in this context involves making sure releases are tagged properly, documented well, and adhere to reasonable code style standards. SQaaS has been integrated into the normflow development workflow via GitHub Actions, obtaining a Silver badge.

The itwinai platform, developed by T6.5, is a framework for advanced ML workflows that is intended for use by AI based DTs. itwinai has been integrated with normflow<sup>12</sup> and this integration has been used to benchmark the performance and energy efficiency of normflow across different distributed training strategies [R16]. A more flexible normflow-itwinai plugin<sup>13</sup> has been created as well, which we have used to generate data on the CESGA HPC.

<sup>12</sup> <https://github.com/interTwin-eu/itwinai/tree/main/use-cases/lattice-qcd>

<sup>13</sup> <https://github.com/interTwin-eu/normflow-plugin>

D4.8 Final version of the DT capabilities for High Energy Physics, Radio astronomy and Gravitational-wave Astrophysics including validation reports

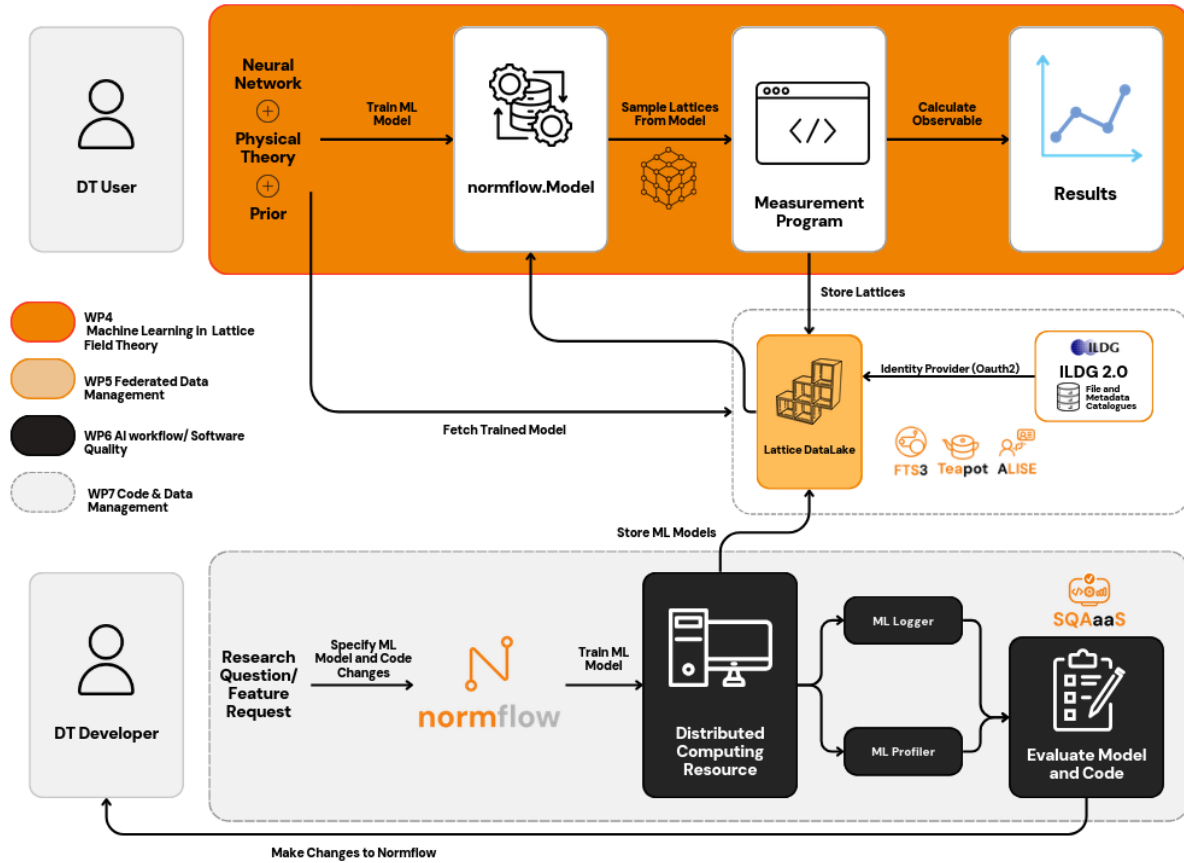


Figure 8 Module Integration Diagram for the Lattice QCD use case

Lattice simulations are executed at large scale on HPC systems controlled by a batch system. In order to facilitate data analysis, lattice data should be readily available to the members of a collaboration in a controlled way. Thanks to the efforts of WP5 and the ILDG, there is now a prototype “Lattice Datalake” with four storage endpoints (SEs): DESY-Hamburg, DESY-Zeuthen, JSC, and CESGA [R24]. **Figure 9** illustrates the flow of control when accessing the datalake, and highlights how a user, once authenticated, can also access the ILDG’s FC and MDC.

To gain authorisation to access the Data Lake, a user must be a member of the ILDG and have an `oidc-agent`<sup>14</sup> configured to interact with an ILDG client — currently, an interTwin public client — to obtain the appropriate access tokens. User permissions are controlled by the Identity Access Management (IAM) and the Access Control Services (ACS) and are encoded in the scopes of the user’s access token. ILDG’s IAM supports a fine-grained authorization model, with permissions controlled through the following scopes: `storage.create`, `storage.modify`, `storage.read`, and `storage.stage`. This model aligns with the structure of a typical lattice collaboration, where tasks are divided among members such that only those directly responsible for lattice generation runs are granted create and modify permissions.

<sup>14</sup> <https://docs.egi.eu/users/aai/check-in/obtaining-tokens/oidc-agent/>





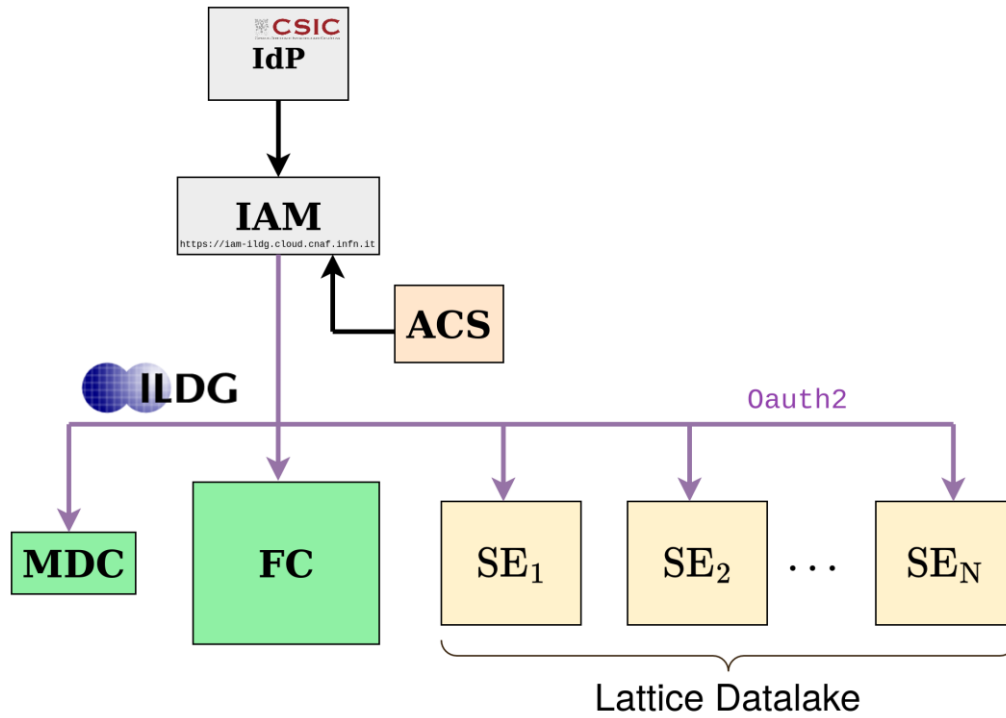


Figure 9 Schema depicting the flow of control when accessing the Lattice Datalake. In this example the institutional IdP is CSIC, but it could be any other IdP trusted by the ILDG.

Once a user has a valid access token, they can interact with the SEs of the Data Lake and, given appropriate authorisation, initiate the transfer of data between them. Users can use `rc1one`<sup>15</sup> to interact with a specific SE, and FTS<sup>16</sup> (the File Transfer Service) for direct data transfers between SEs. The FTS can efficiently handle large numbers of file transfers and supports multiple transfer protocols.

### 3.1.2 Scope and limitations

The `normflow` application's scope includes simulating  $SU(N)$  gauge theories on reasonably sized lattices and comparing its performance with conventional lattice methods. The aim of this work is to guide the research community on whether integrating NF into MCMC algorithms is desirable and, if so, in what manner. While our results and those in the literature are promising and provide ample reason for continued research, it is currently unknown whether ML-based Lattice gauge field theory simulations will be more efficient than MCMC simulations. There are also limitations imposed by using a ML-based method. A whole new set of models needed to be developed for gauge theories, which are based on a particular class of complex matrices that collectively respect a huge class of the so-called gauge symmetries. Training these models is a significant time and resource commitment, and it is not a priori clear if this saves time and computational resources in the regions of parameter space targeted by modern Lattice collaborations.

<sup>15</sup> <https://rclone.org/>

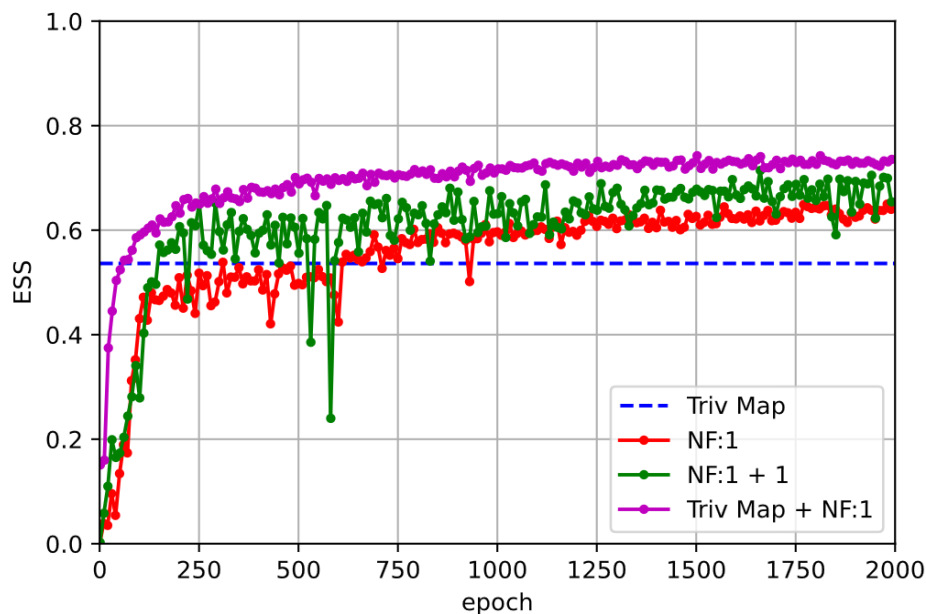
<sup>16</sup> <https://fts.web.cern.ch/fts/>

The Lattice Data Lake, unlike the other Data Lakes developed within interTwin, does not use Rucio to manage transfers because it does not support ILDG's fine-grained authorisation capabilities. Much of the same functionality (querying, token management, transferring data) is achieved by the use of alternative components, in particular `rc1one` and the FTS REST CLI<sup>17</sup>[R20].

### 3.1.3 Validation and Results

The purpose of the `normflow` package is to use generative models to study physical theories. Therefore, to validate its results, we compare them against those obtained using alternative methods of calculation. In most cases, this can be done by simulating the theory using traditional, reliable simulation algorithms, Hybrid Monte Carlo (HMC) algorithms in particular, which also lets us compare the speed and efficiency of the two approaches. For simple, exactly solvable theories, we check whether simulation results agree with the exact answers within uncertainties. We have confirmed that the magnetic susceptibility of a real scalar field using an ML approach and a traditional MCMC approach agrees within uncertainties.

One of our preferred metrics for judging the quality of a trained model is the Effective Sample Size (ESS), where an ESS of 1 indicates a perfectly trained model. **Figure 10** shows the evolution of the ESS for three SU(3) models on a  $4^4$  lattice with  $\beta = 1$ . For comparison, the leading-order trivializing map (with no trainable parameters) is shown with a blue dashed line. The training progress for a model with 1288 parameters is shown in red. The model's ESS matches the reference trivializing map after about 500 epochs and surpasses it as the training progresses. **Figure 10** highlights the impact on model quality due to network and prior selection.



<sup>17</sup> <https://fts3-docs.web.cern.ch/fts3-docs/fts-rest/docs/cli/>



Figure 10 The Effective Sample Size (ESS) of a ML model measured every 10 epochs during the training of an SU(3) gauge theory with  $\beta=1$  on a 4x4x4x4 lattice [R22]

## 3.2 DT Application: Detector simulation

This section provides an overview of CERN's DT application of a calorimeter detector simulation, the current status of the DT and its integration with other DTE components. It describes the key steps, from particle simulations to event generation, and subsequent data comparison with real data. The process is explained in detail, highlighting the functionalities at each stage. Furthermore, it illustrates the flexibility in tuning the system to accurately represent various detectors' responses. This explanation is designed to give readers an understanding of the entire workflow design, shedding light on current practices and potential areas of future improvement. It also opens the way for a deeper discussion on the challenges faced, decisions made, and future strategies in the ongoing development of this innovative simulation application.

The calorimeter detector simulation DT is described by two main workflows, the training workflow and the inference workflow. The training workflow design includes the following functionalities, which run on HPC systems (in the case of 3DGAN this process is managed by containerized Kubeflow components). Geant4 simulates particle interactions, producing data based on a detector-specific configuration. The produced data consists of the energy per calorimeter cell measured by the detector sensors, the properties of the initial particle, such as its type, energy, trajectory angle with respect to the detector volume and other metadata. The produced data, in ROOT format<sup>18</sup>, is stored at different data centres, with CERN currently serving as the primary storage site. The data produced from the traditional Geant4 simulation in ROOT format requires conversion into the HDF5<sup>19</sup> format for further preprocessing before being input into the 3DGAN or CaloINN model. The converted data will then be stored at data centres. Following the ROOT to HDF5 format conversion, the HDF5 data is further pre-processed and transformed into numpy arrays, a process currently incorporated within the model training scripts. The generative models are trained on the pre-processed data, conditioned on specific input describing the properties of the particles. The data is retrieved from the storage space where they reside. During validation and hyperparameter optimization, the model generated data and the Geant4 simulated data distributions will both be visualised. At the end, the training workflow stores the optimised models, selected based on validation results.

The inference workflow design includes the following functionalities. The 3DGAN model performs inference on specified initial conditions of a particle's energy and angle, and the CaloINN model takes only the particle's energy as initial condition. The 3DGAN or CaloINN model completes the process of generating events, simulating the passage of particles through the calorimeter of a detector. Data distribution comparisons are drawn between the AI-generated data and Geant4 data. These comparisons are essential for validating the efficacy and accuracy of the AI-generated data.

---

<sup>18</sup> <https://root.cern/>

<sup>19</sup> <https://www.hdfgroup.org/solutions/hdf5/>



Finally, based on the visualised results, two possible workflows are proposed for simulation tuning. The model can either be re-inferred with different model input parameter values, provided these parameter values have been accounted for during model training. Alternatively, if a different value range of the conditional parameters or another detector geometry is needed, the training workflow must be re-run from the beginning and new simulated data would need to be produced using the Geant4 application. These two possible workflows allow for greater flexibility and adaptability in tuning the detector's responses to various particle interactions.

### 3.2.1 DTs Application Integrations

The workflows of the particle detector simulation DT (**Figure 11** for 3DGAN model) described above will exploit several components from the project's DTE. Related integration activities with these components have been performed.

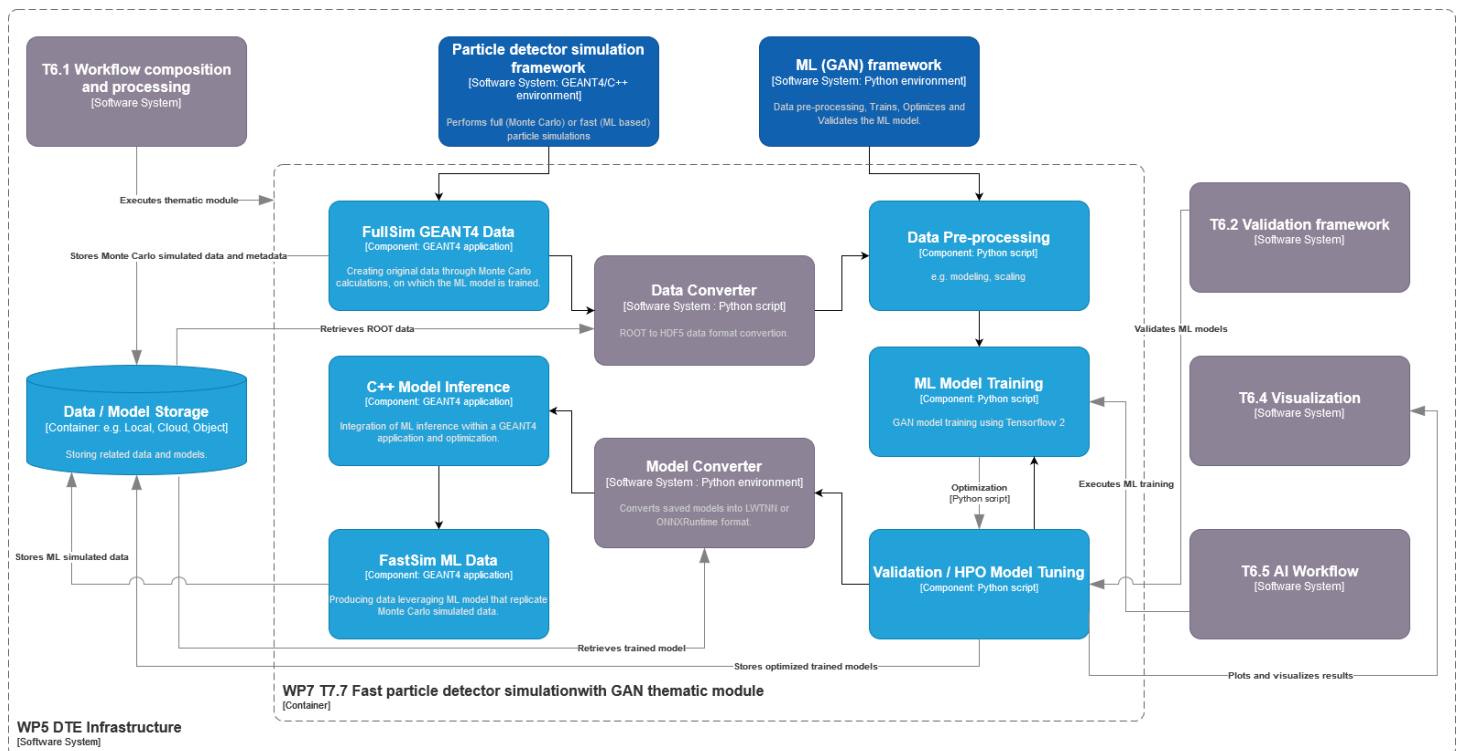


Figure 11 High-level workflow composition of the fast particle detector simulation DT using ML techniques, and its connections with components from other work packages

The workflows depend on the WP7 3DGAN<sup>20</sup> and the CaloINN components reported in D7.9. The 3DGAN model is implemented based on the PyTorch Lightning ML framework, and the CaloINN model uses PyTorch. Both models have been trained to produce data similar to the one produced by Monte Carlo simulations with Geant4. 3DGAN models the calorimeter detector's layers of cells as monochromatic, pixelated images, where the cell energy depositions correspond to pixel intensities. 3DGAN consists of 2 networks, a generator and a discriminator, the two networks compete with each other trying to optimise a loss function until the convergence point, where the discriminator won't be able to distinguish the images generated by the generator from

<sup>20</sup> <https://github.com/interTwin-eu/DetectorSim-3DGAN/tree/main>

the real images. Each network is trained using 3-dimensional convolution layers to represent the 3 spatial dimensions of the calorimeter images. CaloINN represents the calorimeter response as energy deposits in a detector volume segmented into layers aligned with the direction of the incoming particle, with each layer further divided into radial and angular bins in polar coordinates (voxels). This model consists of a sequence of invertible layers that learn a transformation from a simple, known distribution to a complex distribution of calorimeter outputs.

The particle detector simulation DT includes activities towards the implementation of the validation framework of the model's performance. This involves Python analysis scripts that compare the AI-generated calorimeter responses to Geant4 data events.

Integration activities of the SQAaaS service<sup>21</sup> developed by T6.2 have been performed for 3DGAN models. Software quality assurance in this context involves making sure releases are tagged properly, documented well, and adhere to reasonable code style standards. SQAaaS has been integrated successfully into the detector simulation ML development workflow via GitHub Actions.

itwinai developed under T6.5 is a Python library that streamlines AI workflows, while reducing coding complexity. It seamlessly integrates with HPC resources, making workflows highly scalable and promoting code reuse. With built-in tools for hyperparameter optimization, distributed machine learning, and pre-trained ML models, itwinai empowers AI researchers. It also integrates smoothly with Jupyter-like GUIs, enhancing accessibility and usability. With itwinai exploitation we have managed to represent our AI workflows in a modular and reproducible way using configuration files, which break down the training and inference workflows into steps. The steps included are data loading, pre-processing, ML training and testing. This provides a unified entry point from which the entire AI workflow can be configured and modified in successive executions. The integration with itwinai framework has been successfully completed for both models (the CaloINN inference step implementation is being finalised at the time of writing this report). With itwinai, it is possible to inspect and analyse the training results from MLFlow<sup>22</sup> dashboards, as well as store and manage models in the provided model registry.

Moreover, for 3DGAN model, T5.2 engineers have successfully uploaded the simulated data produced by Geant4 and transformed into HDF5 format, into the interTwin data lake.

Through the integration of the particle detector simulation DT with itwinai framework, the DT has been successfully deployed accessing the project infrastructure resources. Exploiting a TOSCA template and a deployed Kubeflow instance (T6.4), developers have managed to access Jupyter notebooks and deployed the 3DGAN model from detector simulation use-case. The 3DGAN training configuration files were used again and were translated into relevant Kubeflow pipelines<sup>23</sup>.

---

<sup>21</sup> <https://confluence.egi.eu/display/interTwin/Use+cases+tracking>

<sup>22</sup> <https://mlflow.org/>

<sup>23</sup> <https://www.kubeflow.org/docs/components/pipelines/v1/introduction/>



Furthermore, the 3DGAN model from the detector simulation use case has been deployed through the itwinai package inside a Docker container using interLink<sup>24</sup>. The training was executed at the Jülich Supercomputing Centre, utilising this deployment. By defining a Kubernetes pod, developers managed to allocate the desired hardware (GPUs) for the model training process. The training was then executed by using an itwinai use-case image that was staged beforehand and by stating the training command as defined in the itwinai configuration files.

Finally, since the training of the generative models is completed on HPC (both 3DGAN and CaloINN) and cloud (3DGAN only) resources, developers have used the models in inference mode to make predictions and produce detector output. This was achieved by exploiting OSCAR<sup>25</sup> for 3DGAN. OSCAR is a serverless computing platform that is accessing HPC resources and offloading computations using interLink for federated computing infrastructure. Using an itwinai inference pipeline for the 3DGAN detector simulation, along with the provided use-case image and a shell script that performs inference on the input data, the resulting data flow could be inspected and the storage events monitored. Data produced during inference were uploaded to dCache, which triggers an OSCAR service to process the data by offloading a job to the specified computing resources. In this context, the particle detector simulation DT was successfully integrated with the aforementioned DTE components and offloaded for inference to a VEGA node. The above processes can also be observed in a interTwin engine demo<sup>26</sup>, where the 3DGAN detector simulation use case is used for testing and experimentation.

### 3.2.2 Scope and limitations

By developing the calorimeter detector simulation use-case and integrating it with the DTE components, we aim to provide a seamless Proof of Concept (PoC) of a physics DT leveraging ML capabilities.

The scope includes the development and implementation of both training and inference workflows, using high-performance computing systems. These workflows involve the simulated data conversion and preprocessing, and the training of the 3DGAN or/and CaloINN model to generate calorimeter energy depositions that mimic real detector outputs. The DT leverages ML frameworks such as PyTorch Lightning<sup>27</sup> and the itwinai package. This integration facilitates scalable and efficient training, model validation, and inference processes, thereby enhancing the overall effectiveness and accessibility of the AI workflows.

The workflows exploit various other components, such as the interTwin data lake, interLink, OSCAR and SQAaaS, available for 3DGAN model. These integrations help in

---

<sup>24</sup> <https://interlink-project.dev/>

<sup>25</sup> [https://docs.google.com/document/d/1AxD8kXddzSgePBc3X7381YP1T\\_EbDWwu3Xb38ox1a-k/edit](https://docs.google.com/document/d/1AxD8kXddzSgePBc3X7381YP1T_EbDWwu3Xb38ox1a-k/edit)

<sup>26</sup> [https://www.youtube.com/watch?v=NoVCfSxwtX0&ab\\_channel=interTwin](https://www.youtube.com/watch?v=NoVCfSxwtX0&ab_channel=interTwin)

<sup>27</sup> <https://lightning.ai>



managing the data effectively and maintaining high standards in software development and deployment.

The DT's efficacy depends on the 3DGAN and CaloINN models' ability to generalise from the trained scenarios. Changes in the detector configurations or operating conditions that were not included in the training data might require retraining of the model, which can be resource intensive. Continuous validation is critical to ensure the selected model's outputs remain reliable over time. There is also a risk of bias in the model outputs if not adequately addressed during the model training and validation phases, which could lead to inaccurate simulations. Moreover, the integration of multiple components and services introduces complexity. This could lead to challenges in maintaining system stability and troubleshooting, especially when deploying updates or new features.

By understanding these scope and limitations, the involved stakeholders can better manage their expectations and plan for future enhancements to ensure the DT remains a robust and valuable PoC for CERN's particle detector simulations.

### 3.2.3 Validation and Results

The performance of generative models for calorimeter response simulation, 3DGAN and CaloINN, was validated using different approaches. First, a set of metrics (such as AUC<sup>28</sup> or JSD<sup>29</sup>) is used to measure the difference between Geant4 and model-generated calorimeter responses. Second, visualisations are available to compare the original and simulated data. Visualisation for 3DGAN model includes plotting averaged calorimeter responses, modelled in 3D, into 3 projections. For CaloINN visualisation includes plotting high-level features describing the calorimeter response in physical terms.

The results show that both 3DGAN and CaloINN models are able to provide high quality calorimeter response simulation faster than Geant4 simulations (when running on GPUs). Users can decide whether considerable speed up is worth the inevitable loss of quality. Currently, the study of possible improvement of CaloINN performance is still ongoing and will continue after the end of the project, targeting the issue of distributions tail mismodelling, common for generative models.

## 3.3 DT Application: Noise simulation for radio astronomy

This DT application addresses the challenge of identifying radio signals from intermittent astrophysical sources from large-volume data streams during the data acquisition phase. One of the main tasks is to identify noise and interference signals coming from different sources. The DT simulates the propagation of pulsar signals from the source to radio astronomical antennas and their subsequent processing by radio telescope electronics, aiming to generate synthetic output signals identical to those recorded by real telescopes. It is a part of a larger framework ML-PPA (**Figure 3**). In addition to the DT, it includes a

---

<sup>28</sup> [https://en.wikipedia.org/wiki/Receiver\\_operating\\_characteristic](https://en.wikipedia.org/wiki/Receiver_operating_characteristic)

<sup>29</sup> [https://en.wikipedia.org/wiki/Jensen%E2%80%93Shannon\\_divergence](https://en.wikipedia.org/wiki/Jensen%E2%80%93Shannon_divergence)

CNN-based ML classifier of the pulsar data, which can be trained using the DT-supplied data.

### 3.3.1 DTs Application Integrations

ML training is implemented using itwinai from WP6, and the modules are integrated with Data Lake. Teapot<sup>30</sup> is currently being implemented at DZA<sup>31</sup> to provide access to input data stored in DZA's repositories. Integration of these components with itwinai has been tested on resources provided by Julich Supercomputing Centre.

### 3.3.2 Scope and limitations

Currently, four software modules have been released (as described in D7.8 [R2]), all under the umbrella designation of ML-PPA:

- **PulsarDT**
- **PulsarRFI\_Gen**
- **PulsarRFI\_NN**
- **PulsarDT++**

**PulsarDT:** A physics-based DT that simulates the propagation of pulsar signals from the source to the telescope (**Figure 12**) and generates synthetic data. It is written in Python to test algorithmic strategies for physical models of pulsars, interstellar medium, telescopes, interference and noise, all later implemented in PulsarDT++.

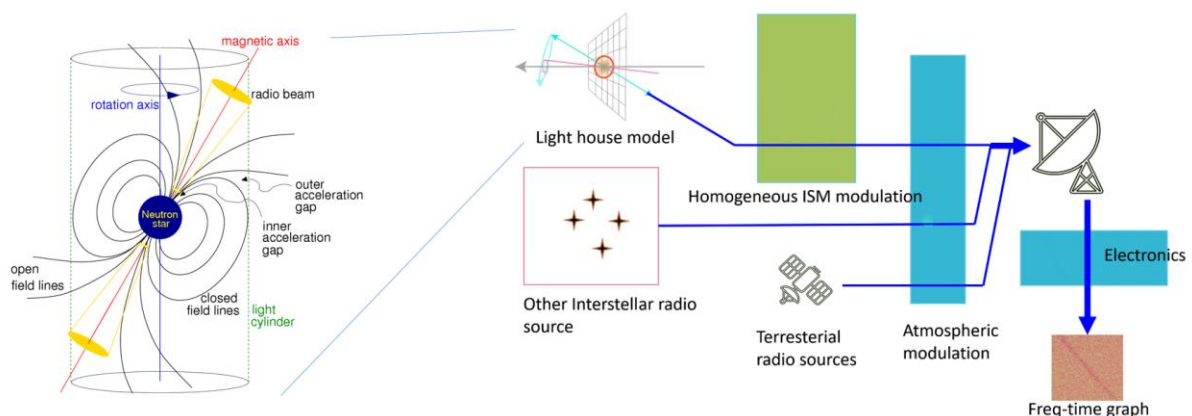


Figure 12 General outline of the DT structure: modelling the astrophysical source (pulsar), transmission of the signal through the interstellar matter, receiving and processing by a radio telescope, adding sources of both natural and artificial interference and noise

**PulsarRFI\_Gen:** An empirical DT that generates 2D time-frequency images (**Figure 4**) of all possible types of telescope output from a pulsar observation: pulses (scientifically relevant data), two different types ("narrow" and "broad") of RFI signals, and "empty" frames, containing only noise. It creates these data frames by mimicking available real data (based on the geometry of images, noise characteristics etc.) rather than generating

<sup>30</sup> <https://intertwin-eu.github.io/teapot>

<sup>31</sup> <https://www.deutscheszentrumastrophysik.de/>



them from the physical first principles as done by PulsarDT. The purpose of this tool is to provide an alternative to training data produced by PulsarDT for testing and QA purposes.

**PulsarRFI\_NN:** The ML classifier, which is a CNN-based tool for identification of various types of pulsar and RFI signals. Like PulsarDT, it is written in Python, with its components later contributing to PulsarDT++.

**PulsarDT++:** This is a C++ implementation of PulsarDT and PulsarRFI\_NN with a general test of architecture of the whole ML-PPA package (see [Figure 13](#) and [Figure 14](#)). The released version (0.2) includes:

- a layered architecture: a Python-based user interface on top of a layer containing various modules for time-critical analyses (implemented in C++),
- a simplified simulation model of pulsar signals,
- real and synthetic (generated by PulsarRFI\_Gen) data of pulsar and RFI signals,
- a container with all necessary components, parallelization using the python-based ray framework.

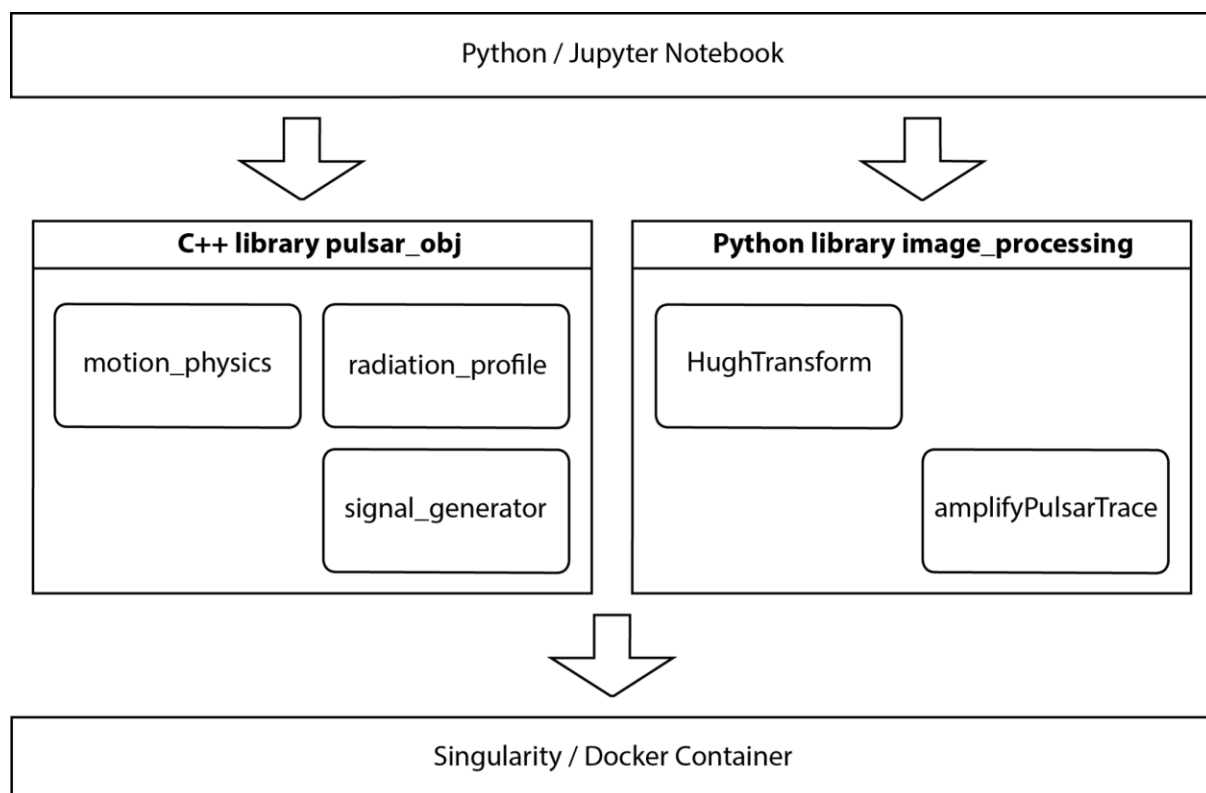


Figure 13 Layered software architecture of the framework ML-PPA

## D4.8 Final version of the DT capabilities for High Energy Physics, Radio astronomy and Gravitational-wave Astrophysics including validation reports

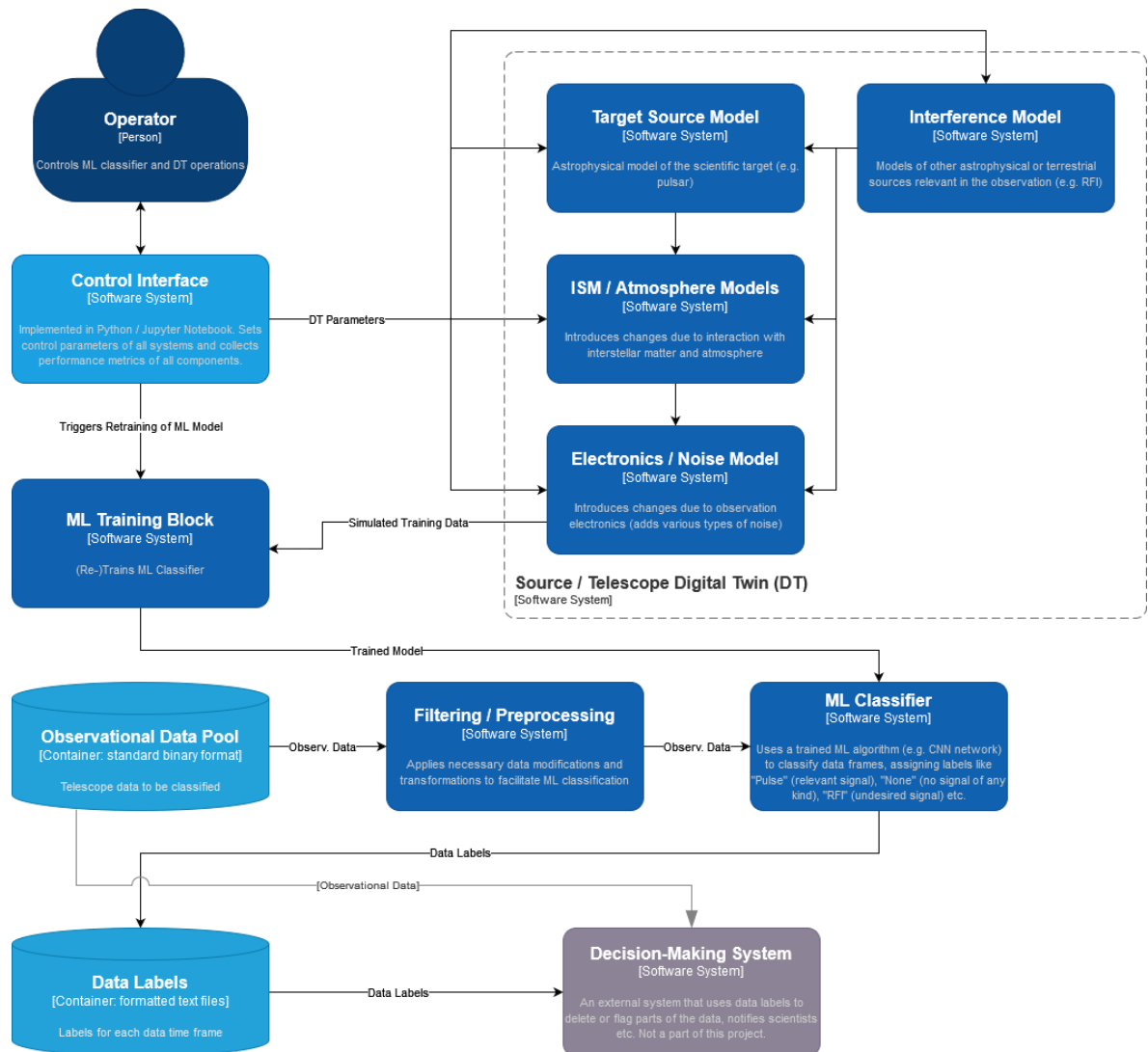


Figure 14 Diagram of the final software product of ML-PPA (in the C4 model)

### 3.3.3 Validation and Results

ML-PPA has been extensively validated using both synthetic data generated with the DT and real datasets obtained by observing several pulsars with Effelsberg and MeerKAT telescopes. It consistently shows detection accuracy above 99% for signal-to-noise ratios 7 and above, and data frames 128 by 128 and above, which is more than enough for real-life radio astronomical applications. Its deployment and operation have been successfully tested at four different HPC centers: Vega (Slovenia), Julich, DESY, and DZA (Germany).

## 3.4 DT Application: VIRGO Noise detector

The core of the VIRGO Noise detector DT is a GNN used to find the transfer function of the system producing non-linear noise in the detector output. The information produced by the GNN is used to identify the presence of glitches in incoming data, which could mimic a signal of astrophysical origin (the gravitational wave).





### 3.4.1 DTs Application Integrations

The DT, as described in **Section 2.4**, is organised as a pipeline in which all the different modules are itwinai plugins developed in collaboration with colleagues from WP 6. During the training phase, accuracy and performance metrics are logged in MLflow directly via itwinai. The updated model is also stored in a dedicated model registry after each training run. The inference module logs the spectrograms of the real and generated data to TensorBoard<sup>32</sup>, where they can be visualised as two-dimensional images.

Another significant integration involves the development of the Virgo Data Lake, in collaboration with the developers from Work Package 5. This is where the data used for training the GNN is stored. The Data Lake is managed by the Rucio<sup>33</sup> software, which ensures scalable and efficient data transfer and storage; it consists of two Rucio Storage Elements (RSEs): one at INFN, where the data is originally stored on tape, and one at the Vega EuroHPC<sup>34</sup>. Once the files are transferred via Rucio to the Vega RSE, the modules can be run in a JupyterHub environment set up with interLink, a service developed for efficient exploitation of federated computing resources. By following this procedure, we were able to fully utilise the computational resources provided by the collaboration.

The high-level architecture of the DT, together with services provided by other working groups, is shown in **Figure 15**.

---

<sup>32</sup> TensorBoard: <https://www.tensorflow.org/tensorboard>

<sup>33</sup> Rucio: <https://rucio.cern.ch/>

<sup>34</sup> Vega: <https://en-vegadocs.vega.izum.si/introduction/>



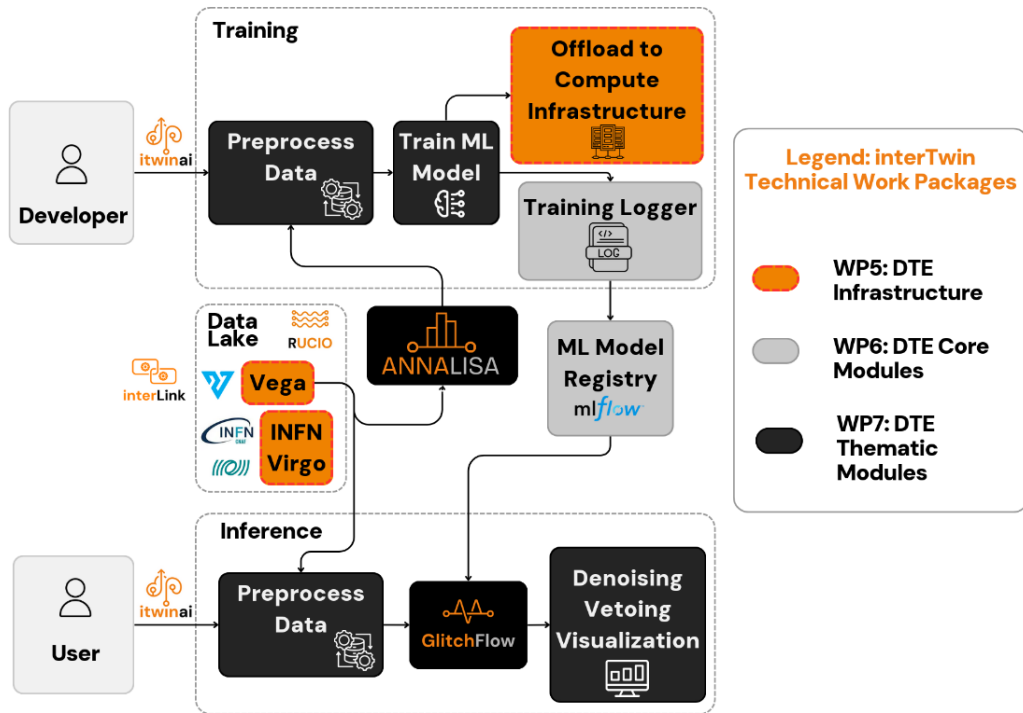


Figure 15 System Context diagram of the DT for the veto/de-noising pipeline

### 3.4.2 Models and Data Requirements

All the modules of the DT described in [section 2.4](#) rely on classes and functionalities in the `ml4gw`<sup>35</sup> github repository, which contains a large number of utilities that we have developed to replicate the functions present inside the `GWpy`<sup>36</sup> package, but making them compatible with PyTorch tensor. This strategy makes it possible to heavily parallelise the computations and drastically reduces the computational costs.

The data used for the off-line analysis and benchmarking was downloaded from the RSE at INFN as described in [section 3.4.1](#).

A pip-installable version of the ANNALISA package, the Jupyter notebook containing GlitchFlow, and the preprocessing module can be found in the GitHub repository<sup>37</sup>.

Following this section, we provide details on the data inputs and corresponding outputs for all the modules described in [section 2.4](#).

#### ANNALISA

The input data consists of time series contained in the strain and the physically selected auxiliary channels, stored in files in HDF5 binary data format. The data is extracted from the files and read as one-dimensional PyTorch tensors.

<sup>35</sup> `ml4gw`: <https://github.com/ML4GW/ml4gw>

<sup>36</sup> `GWpy`: <https://github.com/gwpy/gwpy>

<sup>37</sup> [https://github.com/interTwin-eu/DT-Virgo-dags/tree/main/Final\\_Release](https://github.com/interTwin-eu/DT-Virgo-dags/tree/main/Final_Release)

The time series need to have a minimum length of six seconds and they must include a time in which a glitch was observed in the strain channel, so that a meaningful correlation analysis can be performed.

### **Preprocess\_API**

The input must be a dataset in the form of a new custom data class named TFrame, which we developed. Built upon PyTorch's Tensor, TFrame allows for the inclusion of metadata, such as the ID of each glitch used for the training. The output of the module is another TFrame which contains the spectrograms in the form of two-dimensional CQTs.

### **GlitchFlow**

The input data consists of the TFrame containing spectrograms of the events within the selected time intervals and frequency range. In order to have balanced classes during training, the input data includes not only the spectrograms containing glitches but also those without glitches, allowing the GNN to learn the detector background. The module's output is a trained model stored in a MLflow catalogue as a serialized PyTorch state dictionary, along with additional metrics on the training process.

### **Generate\_API**

The input data consist of a TFrame of PyTorch spectrograms to be used as test data, along with the trained model stored by GlithFlow in the catalogue. The module outputs a TFrame containing the generated spectrograms for testing, along with accuracy metrics for vetoing and de-noising; all are uploaded to TensorBoard.

## **3.4.3 Scope and limitations**

The DT for the Virgo noise detector allows the user to:

- Identify how sensitive each auxiliary channel of the Virgo detector is to a specific glitch,
- Use the data of said auxiliary channels to reliably simulate a glitch in the strain channel,
- Implement an off-line vetoing and de-noising protocol for GW detector events.

The DT presents the following limitations:

- Vetoing and de-noising can only be performed on scattered-light glitches,
- The performance has only been evaluated on O3a data, and not on the most recent O4 data.

## **3.4.4 Validation and Results**

The vetoing accuracy of the model is computed by summing the number of true positives (glitches are generated when a real glitch is present in the strain channel) and true negatives (no glitch is generated when no glitch is present in the strain channel) and dividing that by the total number of spectrograms generated for a given test set. The de-noising accuracy is evaluated by first subtracting the generated spectrogram from the real one in the strain channel; if the resulting spectrogram has no glitches, it is counted



as successfully de-noised. We then divide the number of successes by the total number of generated spectrograms.

The data was split into a training, validation and test dataset, respectively of 9600, 1200 and 1200 elements each. After the training we tested the vetoing and de-noising accuracy as a function of the threshold sensitivity. By threshold sensitivity we refer to the minimum Signal-to-Noise-Ratio (SNR) which needs to be recorded in a spectrogram for it to be considered as glitch-containing.

For a low SNR=3, we measure a vetoing accuracy of 85% and a de-noising accuracy of 95%. For SNR=6, which is the standard lower bound for the SNR for many on-line analysis tools used by the Virgo collaboration [R23], the vetoing accuracy is 95% and the de-noising accuracy is 99%. These measures are shown in **Figure 16**; in **Figure 17** we show the example of a glitch which has been correctly generated and subtracted from its real spectrogram.

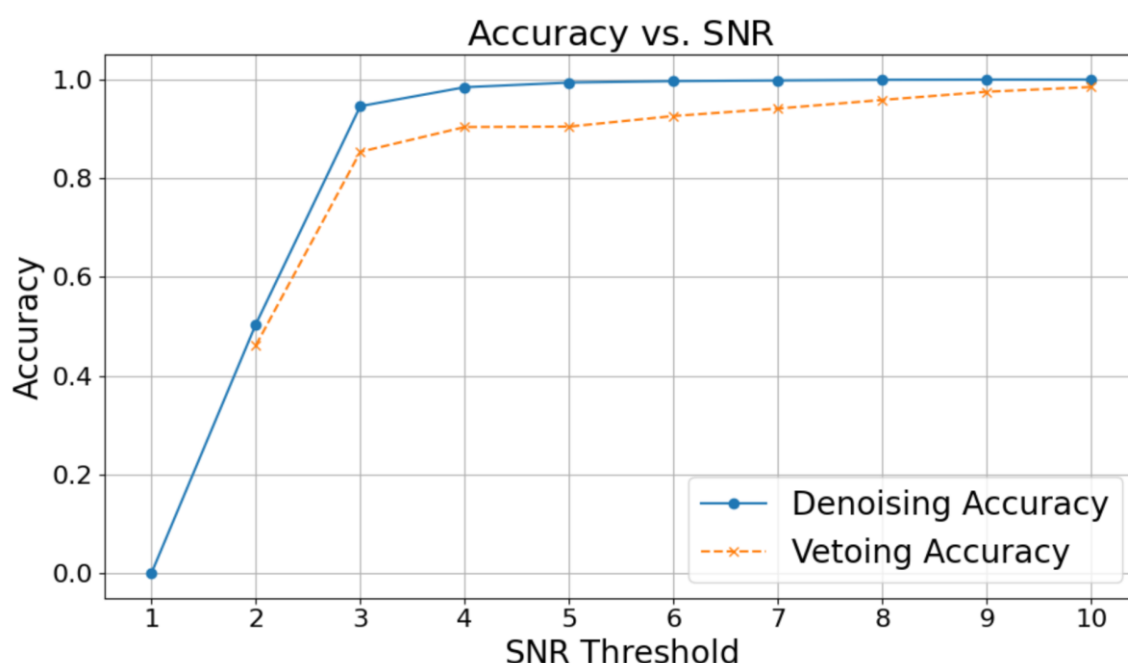
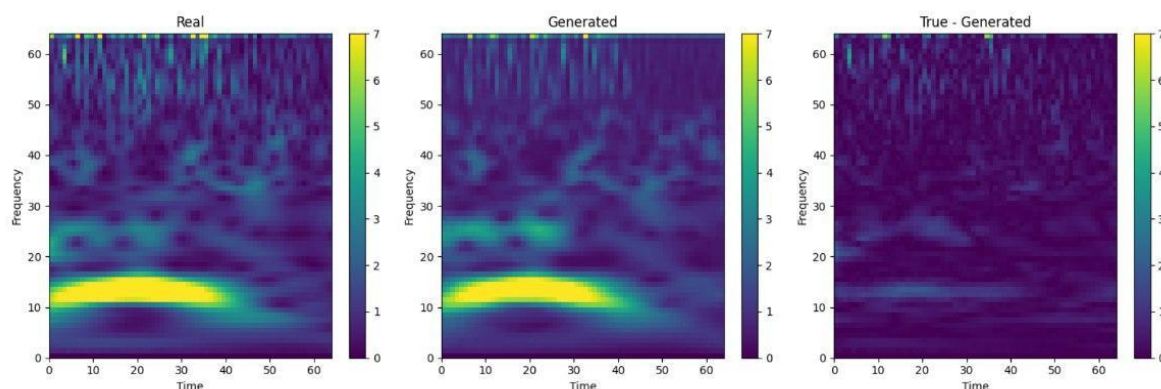


Figure 16 The portions of the data which are correctly de-noised and cleaned, as a function of the threshold that we set to define what constitutes a glitch. Spectrograms which contain at least one pixel above the threshold are counted as “glitch-containing” and considered as not properly cleaned. The SNR=6 value is a standard choice for the lowest sensitivity in low-latency glitch-detection pipelines.



*Figure 17 An example of a spectrogram containing a Scattered Light glitch that is correctly generated by the DT and then subtracted from the real data strain. The resulting spectrogram only contains background noise.*

These results show that the DT can successfully reproduce and remove the majority of Scattered Light glitches, making it a useful tool for increasing the sensitivity of the Virgo Detector.

## 4 Conclusions

The final release of the interTwin DTs applications for WP4 focused on the physics domain was developed in the last months of the project. In the current deliverable D4.8, the main focus is on specifying (in **Section 2**) the capabilities of each specific DT (T4.1, T4.2, T4.3, and T4.4) and describing the final release (in **Section 3**), its integration with the Infrastructure components provided by WP5 and the Core components provided by WP6, along with schematic high-level workflows, scope and limitations of the final release are identified for each individual task.

ML-PPA, the framework, combining four applications: PulsarDT, PulsarRFI\_Gen, PulsarRFI\_NN, and PulsarDT++, allows to generate synthetic datasets identical to real ones as observed by telescopes and to process real datasets, identifying their characteristics. It has been extensively validated using both synthetic data generated with the DT and real datasets obtained by observing several pulsars with Effelsberg and MeerKAT telescopes. It consistently shows detection accuracy above 99% for signal-to-noise ratios 7 and above, and data frames 128 by 128 and above, which is more than enough for real-life radio astronomical applications. Its deployment and operation has been successfully tested at four different HPC centers: Vega (Slovenia), Julich, DESY, and DZA (Germany).

The final version of the Lattice DT, normflow, empowers researchers to explore the application of generative ML techniques to lattice simulations, while being augmented and extended by the integration of multiple components from the DTE (SQAaaS, itwinai, DataLake).

The Detector Simulation DT provides researchers with two generative model approaches for fast calorimeter response simulation. Either model can be used depending on the technical requirements of the produced simulation. The itwinai implementation ensures scalability and monitoring.

The final version of GlitchFlow makes it possible to faithfully simulate scattered-light glitches as they are recorded in the main observation channel of the Virgo detector. This makes it possible to subtract the 2D spectrogram of the noise from the strain for realistic SNR values. The entire pipeline is implemented as itwinai modules, ensuring easy deployability, scalability and easy access to the output metrics of the model.

## 5 References

Reference	
No	Description/Link
<b>R1</b>	<b>Flow-based generative models for Markov chain Monte Carlo in lattice field theory.</b> M. S. Albergo, G. Kanwar, and P. E. Shanahan <i>Phys. Rev. D</i> 100, 034515 (2019) DOI: <a href="https://link.aps.org/doi/10.1103/PhysRevD.100.034515">https://link.aps.org/doi/10.1103/PhysRevD.100.034515</a>
<b>R2</b>	<b>interTwin D7.8 Final version of the thematic module for the physics domain</b> (V1 Under EC review). Sinha Ray G. et al., (2025). DOI: <a href="https://doi.org/10.5281/zenodo.14931995">10.5281/zenodo.14931995</a>
<b>R3</b>	<b>interTwin D7.4 First version of the thematic modules for the physics domain</b> (Submitted to EC). G. S. Ray <i>et al.</i> (2023). DOI: <a href="https://doi.org/10.5281/zenodo.10224277">10.5281/zenodo.10224277</a>
<b>R4</b>	<b>interTwin D4.6 Final Architecture design of the DTs capabilities for High Energy Physics, Radio astronomy and Gravitational-wave Astrophysics</b> (Submitted to EC). K. Tsolaki <i>et al.</i> (2025). DOI: <a href="https://zenodo.org/records/15120028">https://zenodo.org/records/15120028</a>
<b>R5</b>	<b>Fast Simulation for ATLAS: Atlfast-II and ISF.</b> W. Lukas. Technical Report ATL-SOFT-PROC-2012-065, CERN, Geneva, Jun (2012) DOI: <a href="https://doi.org/10.1088/1742-6596/396/2/022031">https://doi.org/10.1088/1742-6596/396/2/022031</a>
<b>R6</b>	<b>Fast simulation of the CMS detector.</b> D. Orbaker. J. Phys. Conf. Ser., (219):32–53, 2010. Part of Proceedings, 17th International Conference on Computing in High Energy and Nuclear Physics (CHEP 2009): Prague, Czech Republic (2009). DOI: <a href="https://doi.org/10.1088/1742-6596/219/3/032053">https://doi.org/10.1088/1742-6596/219/3/032053</a>
<b>R7</b>	<b>Fast simulation of electromagnetic showers in the ATLAS calorimeter:</b> Frozen showers. E. Barberio et al. J. Phys. Conf. Ser.,160, 012082, (2009). DOI: <a href="https://doi.org/10.1088/1742-6596/160/1/012082">https://doi.org/10.1088/1742-6596/160/1/012082</a>
<b>R8</b>	<b>Generative Adversarial Networks.</b> Ian J. Goodfellow et al. DOI: <a href="https://doi.org/10.48550/arXiv.1406.2661">https://doi.org/10.48550/arXiv.1406.2661</a>
<b>R9</b>	<b>Fast Simulation of a High Granularity Calorimeter by Generative Adversarial Networks.</b> Gul Rukh Khattak et al. <a href="https://arxiv.org/abs/2109.07388">https://arxiv.org/abs/2109.07388</a> DOI: <a href="https://doi.org/10.48550/arXiv.2109.07388">https://doi.org/10.48550/arXiv.2109.07388</a>
<b>R10</b>	<b>Normalizing Flows for High-Dimensional Detector Simulations;</b> F. Ernst et al. (2025) DOI: <a href="https://doi.org/10.48550/arXiv.2312.09290">10.48550/arXiv.2312.09290</a>
<b>R11</b>	<b>Normalizing Flows: An Introduction and Review of Current Methods.</b>





	<b>Ivan Kobzyev;</b> Simon J.D. Prince; Marcus A. Brubaker. IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 43, Issue: 11, 01 November 2021) DOI: <a href="https://doi.org/10.1103/PhysRevD.100.034515">10.1103/PhysRevD.100.034515</a>
<b>R14</b>	<b>ML-based Pipeline for Pulsar Analysis (ML-PPA).</b> Andrei Kazantsev, Tim Oelkers, Yurii Pidopryhora, Tanumoy Saha, Marcel Trattner, and Hermann Heßling (2023). DOI: <a href="https://gitlab.com/ml-ppa/gitlab-profile/-/blob/main/PUNCH_interTwin_project.pdf">https://gitlab.com/ml-ppa/gitlab-profile/-/blob/main/PUNCH_interTwin_project.pdf</a>
<b>R15</b>	<b>Gravity Spy: integrating advanced LIGO detector characterization, machine learning, and citizen science.</b> M. Zevin et al. DOI: <a href="https://arxiv.org/abs/1611.04596">https://arxiv.org/abs/1611.04596</a>
<b>R16</b>	<b>Enabling Lattice QCD Normalizing Flows in HPC Infrastructures</b> Bunino, M. et al. Poster, <a href="#">PASC25</a> (2025)
<b>R17</b>	<b>How OpenID Connection Works.</b> Accessed on 07/03/2025: <a href="https://openid.net/developers/how-connect-works/">https://openid.net/developers/how-connect-works/</a>
<b>R18</b>	Komijani, J. and Ray, G. normflow_ (Version v1.1.0-beta) [Computer software]. (2024). <a href="https://github.com/interTwin-eu/Use_Case_T4.1_normflow">https://github.com/interTwin-eu/Use_Case_T4.1_normflow</a>
<b>R20</b>	<b>FST client- installation and configuration manual.</b> Accessed on 07/03/2025: <a href="https://confluence.egi.eu/display/interTwinDTE/FTS+client+-+installation+and+configuration+manual">https://confluence.egi.eu/display/interTwinDTE/FTS+client+-+installation+and+configuration+manual</a>
<b>R21</b>	<b>interTwin D7.6 Updated report on requirements and thematic modules functionalities for the physics domain,</b> Vallero, S. et al. <i>Zenodo</i> . (1 Approved by the EC). (2024). DOI: <a href="https://zenodo.org/records/14975072">https://zenodo.org/records/14975072</a>
<b>R22</b>	Javad Komijani, J. and Marinkovic, M.K. "Normalizing flows for SU(N) gauge theories employing singular value decomposition". <i>The 41th International Symposium on Lattice Field Theory</i> . (2024). DOI: <a href="https://arxiv.org/abs/2501.18288">https://arxiv.org/abs/2501.18288</a>
<b>R23</b>	<b>Virgo detector characterization and data quality: tools,</b> Acernese, F., et al., <i>Classical and Quantum Gravity</i> 40.18 (2023).
<b>R24</b>	<b>interTwin D5.5 DTE infrastructure development and integration report,</b> Ciangottini, D. et al. <i>Zenodo</i> (Version 1 Under EC Review) (2025) DOI: <a href="https://doi.org/10.5281/zenodo.16567076">https://doi.org/10.5281/zenodo.16567076</a>

