

interTwin

D7.8 Final version of the thematic modules for the physics domain

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

DT Thematic Modules, Radio Astronomy, Gravitational Waves, High Energy Physics, Machine Learning

interTwin co-designs and implements the prototype of an interdisciplinary Digital Twin Engine (DTE). The DTE will be an open-source platform that includes software components for modelling and simulation to integrate application-specific Digital Twins (DTs). InterTwin's WP7 will provide the aforementioned sets of software components, called thematic modules, for the use cases defined in WP4. This report describes the status of the development of the final version of thematic modules in the physics domain.



| Document Description | | | |
|---|---|----------------|-----|
| D7.8 Final version of the thematic module for the physics domain | | | |
| Work Package number 7 | | | |
| Document type | Deliverable | | |
| Document status | Under EC Review | Version | 1.0 |
| Dissemination Level | Public | | |
| Copyright Status |  <p>This material by Parties of the interTwin Consortium is licensed under a Creative Commons Attribution 4.0 International License.</p> | | |
| Lead Partner | MPG | | |
| Document link | https://documents.egi.eu/document/4178 | | |
| DOI | https://zenodo.org/records/14931996 | | |
| Author(s) | <ul style="list-style-type: none"> • Gaurav Sinha Ray (CSIC) • Javad Komijani (ETHZ) • Isabel Campos (CSIC) • Yurii Pidopryhora (MPG) • Sara Vallero (INFN) • Francesco Sarandrea (INFN) • Lorenzo Asprea (INFN) • Sofia Vallecorsa (CERN) • Kalliopi Tsolaki (CERN) | | |
| Reviewers | <ul style="list-style-type: none"> • Michal Orzechowski (ACK) • Liam Atherton (UKRI) | | |
| Moderated by: | <ul style="list-style-type: none"> • Yurii Pidopryhora (MPG) • Andrea Anzanello (EGI) | | |
| Approved by | AMB | | |

| Revision History | | | |
|-------------------------|-------------|---|--|
| Version | Date | Description | Contributors |
| V0.1 | 04/12/2024 | Created template | Charis Chatzikyriakou (EODC) |
| V0.2 | 03/02/2025 | A preliminary draft | All contributors |
| V0.3 | 07/02/2025 | All sections updated, ready for internal review | All contributors / Yurii Pidopryhora (MPG) |
| V1.0 | | Final | |

| Terminology / Acronyms | |
|-------------------------------|---|
| Term/Acronym | Definition |
| GAN | Generative Adversarial Network |
| GNN | Generative Neural Networks |
| CNN | Convolutional Neural Network |
| DAG | Directed Acyclic Graph |
| HPC | High-Performance Computing |
| PoC | Proof of Concept |
| DT | Digital Twin |
| DTE | Digital Twin Engine |
| ML | Machine Learning |
| GW | Gravitational Wave |
| QCD | Quantum Chromodynamics |
| HEP | High Energy Physics |
| HL-LHC | High Luminosity - Large Hadron Collider |
| MC | Monte Carlo |
| ML-PPA | Machine Learning-based Pipeline for Pulsar Analysis |
| RFI | Radio Frequency Interference |

Terminology / Acronyms: <https://confluence.egi.eu/display/EGIG>



Table of Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 8 |
| 1.1 | Scope | 8 |
| 1.2 | Document Structure | 8 |
| 2 | Overview of Thematic Modules for the Physics Domain | 9 |
| 2.1 | T7.1 Lattice QCD simulations | 9 |
| 2.1.1 | Advanced Data management for Lattice QCD | 9 |
| 2.1.2 | Generative models using Machine Learning | 10 |
| 2.2 | T7.2 Noise simulation for radio astronomy | 12 |
| 2.2.1 | Use-case description | 12 |
| 2.2.2 | Machine Learning-based Pipeline for Pulsar Analysis (ML-PPA) | 14 |
| 2.3 | T7.3 GAN-based thematic modules to manage noise simulation, low-latency de-noising, and veto generation for gravitational waves | 16 |
| 2.3.1 | Use-case description | 16 |
| 2.3.2 | High-level architecture of the DT implementation | 17 |
| 2.3.3 | The Training DT subsystem | 18 |
| 2.3.4 | The Inference DT subsystem | 21 |
| 2.3.5 | The Virgo Data Lake | 22 |
| 2.4 | T7.7 Fast particle detector simulation with GAN | 23 |
| 3 | Thematic Modules / Components | 26 |
| 3.1 | T7.1 openQxD | 26 |
| 3.1.1 | Release notes | 27 |
| 3.1.2 | Future plans | 27 |
| 3.2 | T7.1 normflow | 27 |
| 3.2.1 | Release notes | 28 |
| 3.2.2 | Future plans | 28 |
| 3.3 | T7.2 PulsarDT | 28 |
| 3.3.1 | Release notes | 29 |
| 3.3.2 | Future plans | 29 |
| 3.4 | T7.2 PulsarDT++ | 29 |
| 3.4.1 | Release notes | 30 |
| 3.4.2 | Future plans | 30 |
| 3.5 | T7.2 PulsarRFI_Gen | 30 |
| 3.5.1 | Release notes | 31 |
| 3.5.2 | Future plans | 32 |
| 3.6 | T7.2 PulsarRFI_NN | 32 |
| 3.6.1 | Release notes | 32 |
| 3.6.2 | Future plans | 33 |
| 3.7 | T7.3 Virgo DT Datastore Services | 33 |
| 3.7.1 | Release notes | 34 |
| 3.7.2 | Future plans | 34 |



| | | |
|------------|--------------------------|-----------|
| 3.8 | T7.7 3DGAN | 34 |
| 3.8.1 | Release notes | 36 |
| 3.8.2 | Future plans | 40 |
| 4 | Conclusions | 40 |
| 5 | References | 42 |

List of Figures

| | |
|---|----|
| Figure 1 - Graphical representation of the classical generation of configurations using Monte Carlo algorithms | 11 |
| Figure 2 - Graphical representation of the Normalising Flows method | 11 |
| Figure 3 - Module Integration Diagram for the Lattice QCD use case | 12 |
| Figure 4 - General outline of the DT structure. | 15 |
| Figure 5 - Diagram of the ML-PPA (in the C4 model) | 16 |
| Figure 6 - High-level architecture of the DT | 17 |
| Figure 7 - System Context diagram (in the C4 model) of the DT for the veto pipeline. | 18 |
| Figure 8 - Services layout..... | 19 |
| Figure 9 - Airflow DAG for training sub-system, as shown in the Airflow Dashboard | 20 |
| Figure 10 - Container diagram (in the C4 model) of the Inference subsystem. | 22 |
| Figure 11 - Graph representation of the training and inference workflows composition of the fast particle detector simulation DT utilising 3DGAN approach | 25 |
| Figure 12 - 2D Projections of Electromagnetic Showers Generated Using PyTorch-Trained 3DGAN Weights | 39 |
| Figure 13 - 2D Projections of Electromagnetic Showers Generated Using TensorFlow-Trained 3DGAN Weights. | 39 |

List of Tables

| | |
|--|----|
| Table 1 - openQxD..... | 26 |
| Table 2 - normflow | 27 |
| Table 3 - PulsarDT | 28 |
| Table 4 - PulsarDT++ | 29 |
| Table 5 - PulsarRFI_Gen | 30 |
| Table 6 - PulsarRFI_NN..... | 32 |
| Table 7 - Virgo DT Datastore Services..... | 33 |
| Table 8 - 3DGAN | 35 |



Executive summary

This report describes the status of development for the final version of the thematic modules that have been identified within the interTwin project for the Digital Twin (DT) applications in the physics domain. It is a collective document written by the scientists developing these modules. The use cases served by these modules cover high energy physics (Tasks 7.1 and 7.7), radio astronomy (Task 7.2), and gravitational wave astronomy (Task 7.3). Each module's functionality is described and contextualised by reference to the relevant DT and its use cases. Also included is a technical summary of each software module providing basic information such as its software licence and release notes. The report concludes with a short summary of the integration status of the different thematic components with the other interTwin work packages and future work.

It is important to mention that, although this document aims to overview the final version of all the thematic components from the physics domain, more developments and extensions are planned for most of the thematic modules in the remainder of the project. To (i) further improve the modules and (ii) adapt the solutions as the integration with the Digital Twins and other components of the interTwin DTE progresses.



1 Introduction

1.1 Scope

The document is an updated version of the previous report **D7.4 First version of the thematic modules for the physics domain** [R2]. Other previous reports have also identified the technical requirements that are important for the development of the thematic modules in WP7 [R1, R3]. This deliverable summarises the status of development of the final versions of the physics thematic modules, which are needed to implement the physics domain DTs and realise the use cases of WP4. In WP7 the physics domain covers:

- T7.1 Lattice QCD simulations and data management
- T7.2 Noise simulation for radio astronomy
- T7.3 GAN-based thematic modules for gravitational waves
- T7.7 Fast particle detector simulation with GAN.

1.2 Document Structure

[Section 2](#) provides an overview of the thematic modules developed as well as the links with the DT applications in the context of their corresponding use case. [Section 3](#) summarises the specifications for each thematic module / software component, including their functionalities, licence, documentation, release notes and a short description of the corresponding future development plans. [Section 4](#) provides a summary of the integration of the thematic modules with the other Work Packages in interTwin, as well as the main conclusions.

2 Overview of Thematic Modules for the Physics Domain

2.1 T7.1 Lattice QCD simulations

Lattice QCD involves the study of the properties of Quantum Chromodynamics in the low energy/strong coupling limit, where perturbation theory breaks down and numerical approaches are required. Within interTwin two parallel and complementary tracks are being explored that address the practical and theoretical challenges of Lattice QCD simulations. These are the practical challenge of storing and moving the ever-increasing amounts of data associated with traditional large scale HPC simulations and the theoretical challenge of exploring, at the proof-of-concept level, the extent to which contemporary Machine-Learning techniques can make lattice simulations more efficient.

2.1.1 Advanced Data management for Lattice QCD

Lattice QCD simulations are executed at a large scale on HPC systems that are controlled by a batch system (such as SLURM¹). A typical workflow involves the generation of lattice field configurations, the measurement of an observable of interest over those configurations, and the statistical analysis of those measurements. All of these steps, but especially the generation of configurations, can be highly computationally intensive.

The openQxD simulation software is a C code designed to simulate QCD and QCD+QED theories on a lattice. It is available on Gitlab² and is described at length in the literature [R4]. A concise description of the software is given in section 3.1 of D7.2 [R1] along with links to further technical documentation. It is under active development though this work is not being done as part of the interTwin project.

In previous deliverables we described some of the issues encountered by lattice researchers when trying to store and access their data [R2]. We argued that lattice configurations should be made more easily available to the members of a collaboration. It was realised early on that the use of federated identities and group-based access control would be crucial to achieving this goal of easier access in a controlled way. The DataLake framework proposed and developed by WP5 followed naturally. In this framework the members of a collaboration would have group-access enabled read permission for their data while a subset of the collaboration, those in charge of generating configurations, would also have write permission. In D7.4 we described our efforts relating to testing and benchmarking the DataLake prototype with real and toy lattice data [R2]. After providing feedback to the DataLake developers it was decided that the Lattice group should get its own Lattice Data Lake in order to satisfy its particular read/write permission specifications.

¹ <https://slurm.schedmd.com>

² <https://gitlab.com/rcstar/openQxD>



D7.8 Final version of the thematic module for the physics domain

Since the last update there have been continued discussions between ourselves, WP5, the ILDG³, and the administrators of the CESGA HPC facility⁴. Our goal is to extend the ILDG catalogue to support a Lattice Data Lake as a possible source of data. It was decided that a phased approach to the Lattice Data Lake rollout would be preferable. The first phase involved opening an FTS connection and transferring data between the DESY-Zeuthen storage endpoint and a new endpoint at CESGA. The required FTS server has been updated and is now able to accept ILDG tokens. This is important as the Lattice Data Lake will use the ILDG as its identity and access manager. Once we are satisfied this FTS connection is robust WP5 will introduce and configure the Rucio service which will act as an intermediary between the users and the Lattice Data Lake. The ILDG can support multiple locations, recorded as URLs, for each piece of data. We outlined a possible schema for the Data Lake URLs that would be recorded in the ILDG in D7.6 [R3].

As simply as possible data access will look like this:

- A client queries the ILDG for the location of some files it wishes to access.
- ILDG authorises the request, verifying the user is allowed to read the files.
- ILDG returns the location of the files along with the token(s) needed for their access.
- For each file the client contacts the storage to request access and supplies the corresponding token.
- The storage verifies the token and provides the requested file.

2.1.2 Generative models using Machine Learning

The efficiency of general purpose Monte Carlo algorithms decreases dramatically when the simulations need to take place near critical points due to critical slowing down. This is a general phenomenon in simulations in Physics related to phase transitions, which happens as well in Lattice QCD, for example with simulations at very fine distances that are needed for extrapolation to the continuum limit. Simulations need to take place in areas of the parameter space where topology freezing (among other factors) induces very large autocorrelations.

There is a developing literature that argues Normalising Flows (a class of deep generative models) may help to improve this situation (a review is available for instance at [R6] and a block diagram illustrating the method is shown in Figure 1). The underlying idea is to use Machine Learning techniques to map the theory of interest to a “simpler” theory, easier to simulate. Several papers, such as [R5], have demonstrated the proof of concept for simple models. However, further studies indicate that the training cost in CPU time can be, in general, prohibitively high for large lattices, and the acceptance rates in the accept/reject step (Figure 2) drop fast as the lattice size increases unless better architectures and methods are achieved (see for example [R7]). The question under

³ <https://hpc.desy.de/ildg/>

⁴ <https://www.cesga.es/en/home-2/>



D7.8 Final version of the thematic module for the physics domain

investigation is therefore how expensive it is to train a model compared with making a classical Monte Carlo simulation.

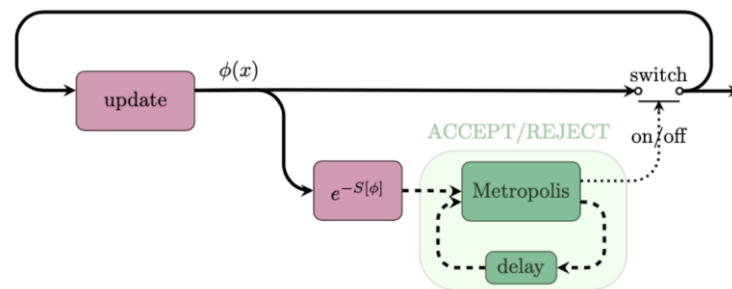


Figure 1 - Graphical representation of the classical generation of configurations using Monte Carlo algorithms

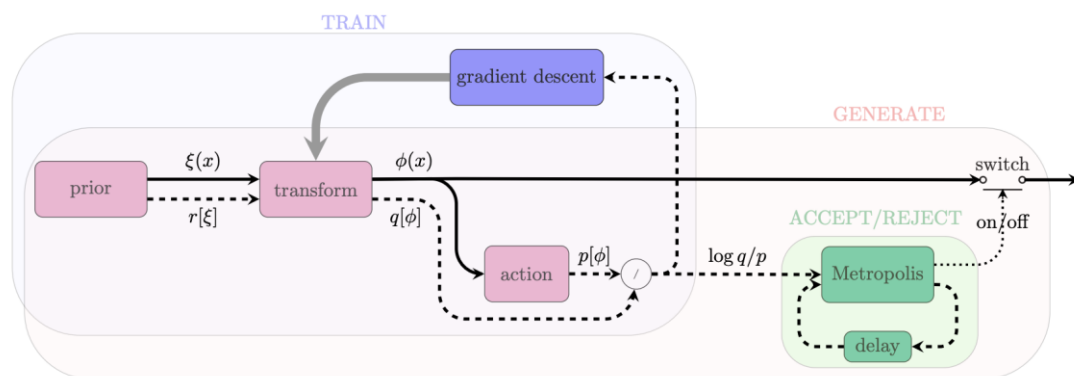


Figure 2 - Graphical representation of the Normalising Flows method including a correcting accept/reject step to account for the fact that the model cannot be perfectly trained

The purpose of this work is to design better architectures for Machine Learning models so that the acceptance rates become reasonable (~50%) as the volume of the lattice increases.

Through the development of normflow⁵ we have shown that Machine Learning can be used for field configuration generation with scalar theories on low dimensional lattices [R8]. Moreover, it is being developed to handle the more complicated family of SU(3) gauge theories, this being an important step towards a ML lattice simulation of QCD [R9]. **Figure 3** outlines schematically the typical workflows of a developer and a user of normflow.

Improvements to the normflow software development workflow have been made by the integration of the SQAaaS module developed by WP6.2, with the progress of this integration being tracked in the corresponding WP4/7 deliverables. Software quality assurance in this context means making sure software packages developed for scientific research, like normflow, adhere to research software best practices, such as being licensed with an Open Source Initiative approved licence. Currently the public version of normflow is credited with the Silver SQAaaS badge and this is displayed prominently on

⁵ <https://github.com/jkomijani/normflow>



D7.8 Final version of the thematic module for the physics domain

normflow's public repository webpage. Since the previous deliverable we have implemented the first automated tests of normflow, using the pytest package and the bash testing framework. Automated testing was demoed by WP6.2 at the 2024 IBERGRID conference. We continue to add tests and are looking at an alternative, more flexible, way of automating the testing with SQAaaS. The main remaining criteria required to obtain the Gold SQAaaS badge are the code style checks. We are in discussions with WP6.2 as to how best to satisfy these criteria.

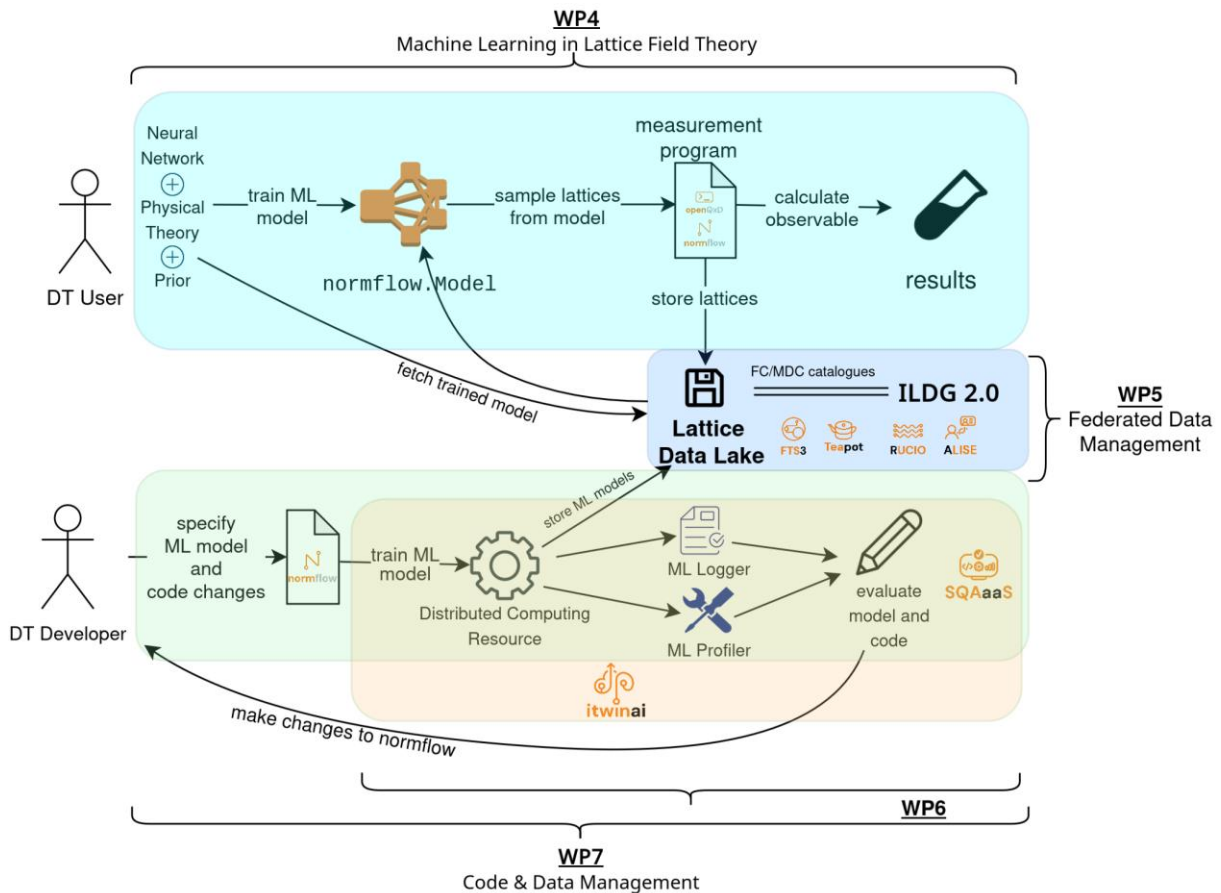


Figure 3 - Module Integration Diagram for the Lattice QCD use case

2.2 T7.2 Noise simulation for radio astronomy

2.2.1 Use-case description

As outlined in the previous reports [R1, R3], this task is designed to be instrumental in solving a big problem that is about to arise in modern observational astronomy in general and radio astronomy in particular, and to become one of the largest issues in the whole field: the problem of data overflow. Previous generations of telescopes typically produced no more than a few petabytes of data per year, thus the raw data was generally kept either indefinitely or long enough for the science team to reduce and analyse it, and then approve the deletion, which meant several months or even years. With the arrival of the



D7.8 Final version of the thematic module for the physics domain

new so-called Square Kilometre Array⁶ "pathfinders", such as South African MeerKAT⁷ or Australian ASKAP⁸, the data acquisition rate increases enormously, these tools can easily produce several petabytes of raw data per week⁹. No current astronomical institution can handle keeping such volumes of data even for a month or employ a team of experts large enough to quickly process it or sort through it manually. Thus, it is crucial to develop automated decision-making systems that can sort through the raw data in real or near-real time (since telescopes usually have downtime due to maintenance or source availability, the data can be pooled for short periods of time of order of days) and separate the data flow into the scientifically important data that must be kept while the rest that can be safely deleted.

Another reason to be able to automatically sort through the incoming data is that modern radio astronomy is increasingly interested in transient sources. Previously sources had to be observed for long periods of time to be able to achieve the necessary signal to noise ratio, thus it was possible to observe reliably or even discover at all only permanent or fast periodic¹⁰ sources like pulsars. Since the new telescopes are much more sensitive, they can systematically probe the transient radio sky, which currently is generally unknown. Such studies are very important, since it is believed that the transients¹¹ result from very far and enormously energetic exotic events (like a collapsing supermassive star) that may provide essential clues for the areas of physics that cannot be studied experimentally in any other way, e.g., quantum gravity. An automated expert system can help with this: if something like a transient source (or unusual in general) signature is found in the data flow, it can immediately trigger the "target of opportunity" mode of observation for the detected anomaly, and alert the scientists on duty, who would decide the best course of further action. This will also allow us to easily organise concerted efforts of observing rare important sources by a number of instruments, covering a range of wavelengths, e.g., combining Earth-based radio observations with space-based optical and X-ray observations — it is already done today, but with typical response times very far from ideal¹².

⁶ SKAO: <https://www.skao.int/en>

⁷ MeerKAT Radio Telescope: <https://www.sarao.ac.za/gallery/meerkat/>

⁸ ASKAP-radio telescope: <https://www.csiro.au/en/about/facilities-collections/atnf/askap-radio-telescope>

⁹ Predicted data rate for an SKA pathfinder like MeerKAT is of order 10 Gbytes/s or up to 1 Pbytes/day. The SKA itself is expected to produce up to 200 Pbytes/day, which is ~70 exabytes per year. To put it into perspective, the latter is about the same as the expected data rate of CERN's LHC after the High-Luminosity upgrade (60 exabytes per year) and at about the same time (SKA's first light is expected in 2027 and the High-Luminosity LHC should go online in 2029).

¹⁰ Known pulsars have periods from a few milliseconds to 8 seconds, thus over a typical observational session of several hours one can observe many pulses, which makes pulsars relatively easy to detect and observe. However, if we imagine a transient phenomenon similar to a pulse of a pulsar, but either non-periodic or with periods of order of hours or days, discovering it is close to impossible except by sheer luck.

¹¹ Examples of such transients that attract a lot of attention in the radio astronomical community are "fast radio bursts" (FRBs), see e.g., [arXiv: 2107.10113](https://arxiv.org/abs/2107.10113) and references thereof.

¹² Even in the best case scenario when a special "target of opportunity" (ToO) event is expected, and a change of scheduling is proposed in advance for all the observatories involved, the actual triggering of such an event is



D7.8 Final version of the thematic module for the physics domain

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. However, because of their nature “silent” most of the time, a telescope observing a pulsar mostly records either an “empty” data stream, i.e., only the noise, or some sort of RFI due to artificial or natural electro-magnetic phenomena unrelated to space.

The third reason for this task is that current common radio astronomy software tools are inadequate, they are computationally slow and handle parallelization poorly. For the tasks at hand, we are building tools that can be efficiently run on modern HPC clusters, with scalability to at least hundreds of cores. It is connected to the main task of the ML data classification system in a way that, although the classification system itself will be run on ordinary observatory computers embedded in a telescope’s data acquisition system, the training of new models before each new type of observation, which is the most computationally intensive task, will have to be performed on supercomputers.

To be able to detect special and important events in the data, one has first to well understand the regular and mundane features of the data stream that in radio astronomy translates into noise and radio-frequency interference (RFI).

2.2.2 Machine Learning-based Pipeline for Pulsar Analysis (ML-PPA)

As previously reported [R2], motivated by these points we are developing a framework for extracting pulsar signals from radio-astronomical observatory data streams, under the designation of ML-PPA (Machine Learning-based Pipeline for Pulsar Analysis): a ML-based data-labelling system that reads the data flow coming from a real telescope observing a pulsar. An important separate component is a DT of an astronomical source-telescope system, able to generate synthetic output signals identical to the data recorded by a real telescope. The resulting DT-generated data is to be used to train the ML data-classification tool. The DT is physics-based: a set of control parameters will allow adjustment of the output to different sources, detection instruments, and observing conditions.

Four modules are being developed under the umbrella designation of ML-PPA:

- [PulsarDT](#)
- [PulsarDT++](#)
- [PulsarRFI Gen](#)
- [PulsarRFI NN](#)

PulsarDT: physics-based DT, simulation of the propagation of pulsar signals from the source to antennas ([Figure 4](#)) and generation of synthetic data – written in Python, to test algorithmic strategies for physical models of pulsars, interstellar medium, telescopes, interference, and noise.

a complicated and disruptive procedure involving many exchanges between various personnel of many institutions, thus the response time is rarely shorter than a day. Using an automated decision making system with pre-approved criteria can change this to minutes, most of the time taken to actually reposition the telescopes.



D7.8 Final version of the thematic module for the physics domain

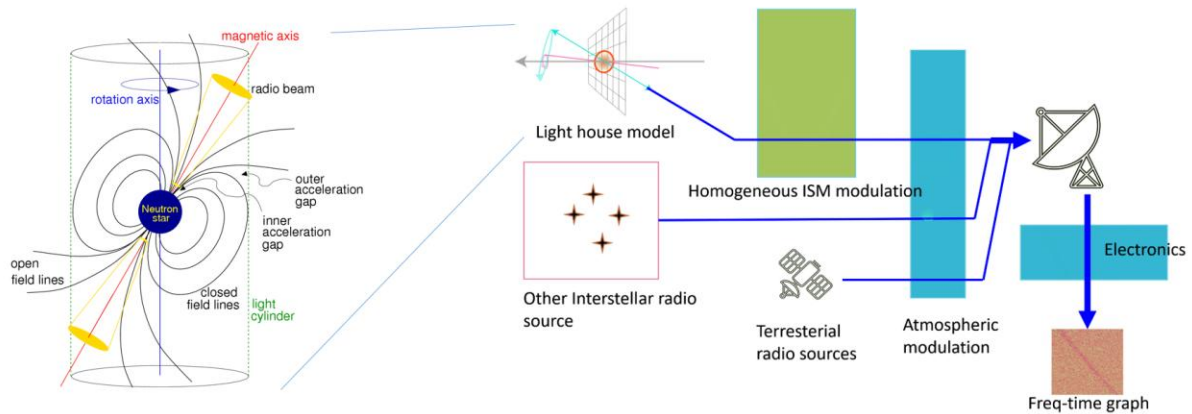


Figure 4 - General outline of the DT structure: modelling the astrophysical source (pulsar), transmission of the signal through the interstellar matter, receiving and processing by a radio telescope, adding sources of both natural and artificial interference and noise.

PulsarDT++: PulsarDT is implemented in C++ in order to improve its speed and allow for parallelization, easily deployable in a singularity container.

PulsarRFI_Gen: empirical DT, generating “timeframes”, 2D images (time-frequency) of all possible types of telescope output observing a pulsar: pulses (scientifically relevant data), two different types (“narrow” and “broad”) of RFI signals, and “empty” frames, containing only noise. It creates these timeframes by mimicking available real data (based on the geometry of images, noise characteristics etc.) rather than generating them from the physical first principles as PulsarDT does. By using this alternative method it provides comparison for PulsarDT/DT++ and substitute training data for the ML classifier.

PulsarRFI_NN: the ML classifier. It is a CNN-based tool for the identification of various types of pulsar and RFI signals in the “timeframes”, 2D images (time-frequency).

The general diagram of the intended operation of the ML-PPA is shown in [Figure 5](#). The final version of the pipeline is intended for use with the real data flow of the MeerKAT telescope, and, later, possibly with other telescopes as well. However, these goals are already beyond the scope of the current project.

ML-PPA is currently being tested with real data collected by observing various pulsars with two telescopes: the Effelsberg 100m radio telescope¹³ and the above-mentioned MeerKAT array.

A more detailed overview of the current state of the project and its full theoretical background can be found in [\[R10\]](#).

¹³ Radio Telescope Effelsberg: <https://www.mpifr-bonn.mpg.de/en/effelsberg>

D7.8 Final version of the thematic module for the physics domain

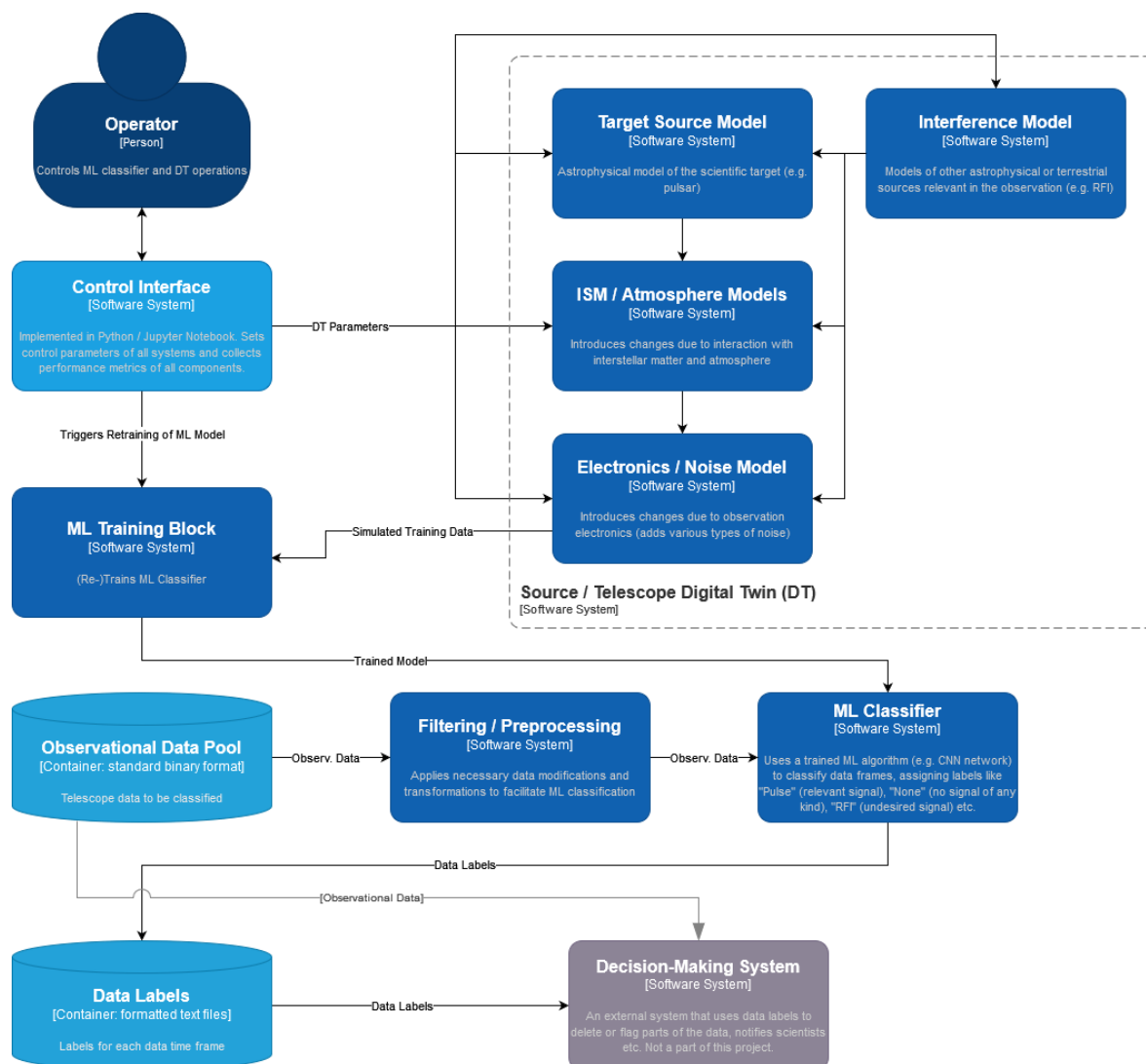


Figure 5 - Diagram of the ML-PPA (in the C4 model)

2.3 T7.3 GAN-based thematic modules to manage noise simulation, low-latency de-noising, and veto generation for gravitational waves

2.3.1 Use-case description

The sensitivity of Gravitational Wave (GW) interferometers is limited by noise. We have been using Generative Neural Networks (GNNs) to produce a Digital Twin (DT) of the Virgo interferometer to realistically simulate transient noise in the detector. We have used the GNN-based DT to generate synthetic strain data (a channel that measures the deformation induced by the passage of a gravitational wave). Furthermore, the detector is equipped with sensors that monitor the status of the detector's subsystems as well as the environmental conditions (wind, temperature, seismic motions) and whose output is saved in the so-called auxiliary channels. Therefore, in a second phase, also from the



D7.8 Final version of the thematic module for the physics domain

perspective of the Einstein Telescope, we will use the trained model to characterise the noise and optimise the use of auxiliary channels in vetoing and denoising the signal in low-latency searches, i.e., those data analysis pipelines that search for transient astrophysical signals in almost real time. This will allow the low-latency searches (not part of the DT) to send out more reliable triggers to observatories for multi-messenger astronomy.

Figure 6 shows the high-level architecture of the DT. Data streams from auxiliary channels are used to find the transfer function of the system producing non-linear noise in the detector output. The output function compares the simulated and the real signals in order to issue a veto decision (to further process incoming data in low-latency searches) or to remove the noise contribution from the real signal (denoising).

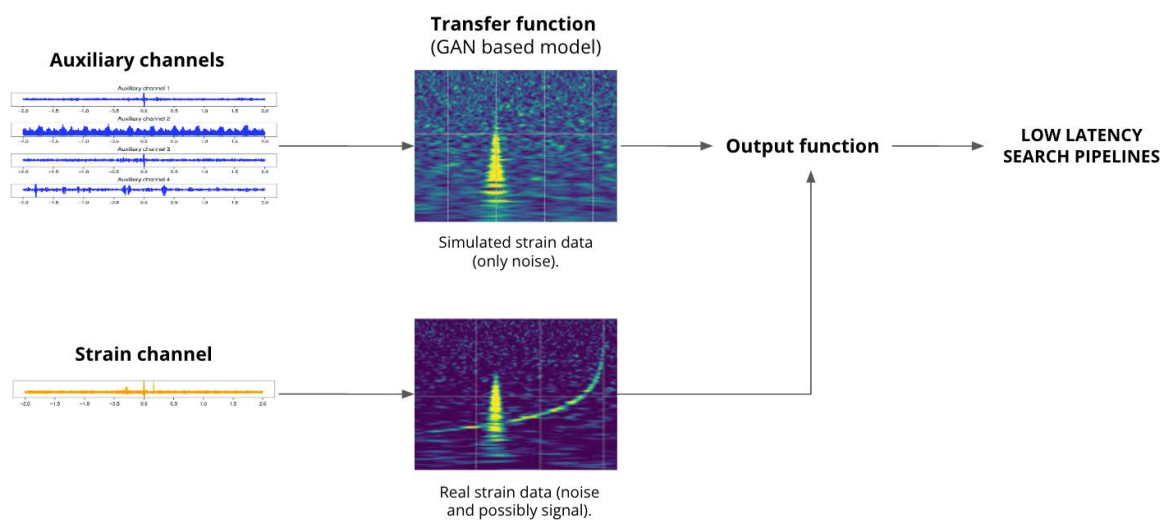


Figure 6 - High-level architecture of the DT

2.3.2 High-level architecture of the DT implementation

Figure 7 shows the System Context diagram (in the C4 model) of the DT for the veto pipeline. In the rest of the document, we will focus on the veto pipeline only, but similar diagrams also apply to the denoising pipeline.

Two main subsystems characterise the DT architecture: the *Training DT subsystem* and the *Inference DT subsystem*. The Training DT subsystem is responsible for the periodical retraining of the DT model on a buffered subsample of the most recent Virgo data. The DT model needs to be updated to reflect the current status of the interferometer, so continuous retraining of the GAN needs to be carried out.

D7.8 Final version of the thematic module for the physics domain

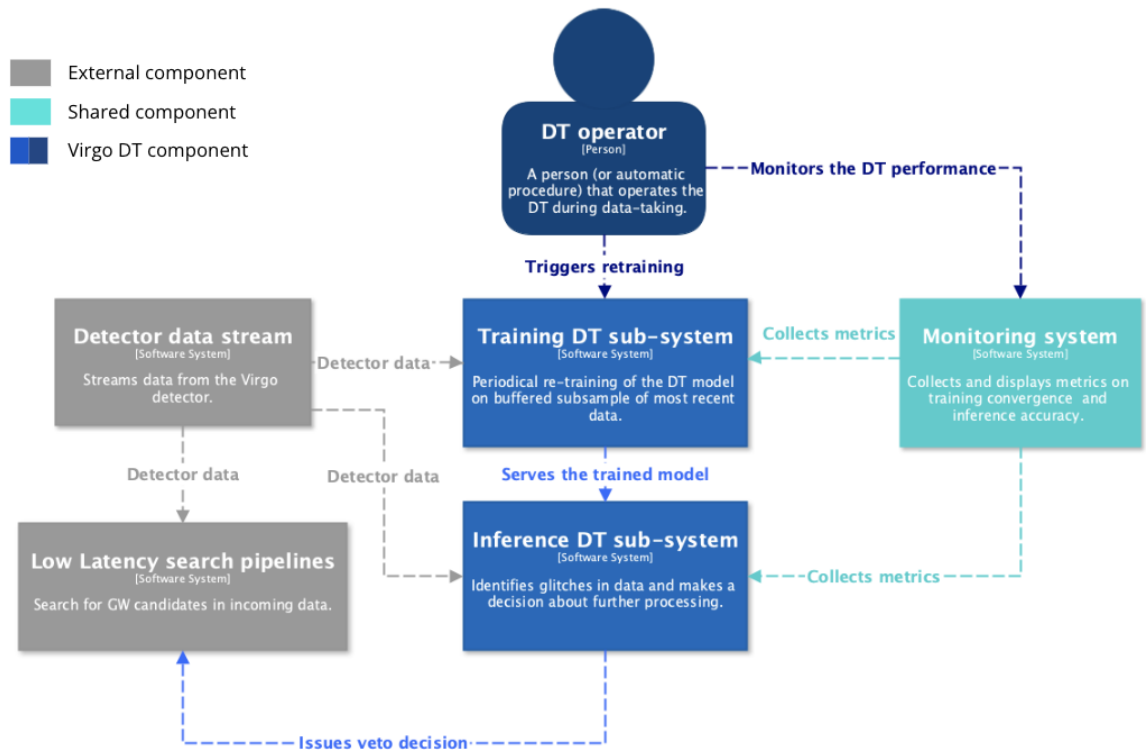


Figure 7 - System Context diagram (in the C4 model) of the DT for the veto pipeline.

2.3.3 The Training DT subsystem

The main component of the Training DT subsystem is the Data Store. The Data Store is used to store data in the form of time-series originating from the Virgo strain channel and relevant auxiliary channels. The length of the buffer is currently under study, but we foresee using about a one-month equivalent of data. The normal operating conditions of the Data Store is to act as a FIFO buffer, receiving an incoming stream of data from the interferometer. The DT Operator triggers the training of the GAN using data from the Data Store periodically or under certain conditions.

The monitoring and coordination of the sub-systems are achieved by implementing the modules as Kubernetes Pods orchestrated by Airflow¹⁴. This is accomplished via several interconnected services designed for managing and processing data, as shown in [Figure 8](#).

¹⁴ <https://airflow.apache.org>



D7.8 Final version of the thematic module for the physics domain

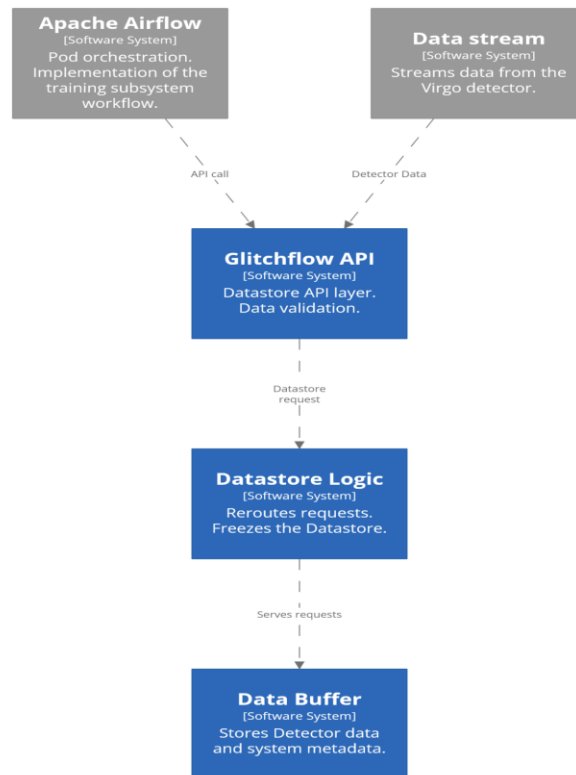


Figure 8 - Services layout

- **The Data Buffer**, implemented in Go¹⁵ using the Gin framework¹⁶, stores GW data and state information, for example the number of files on disk and their total size, utilising Go's concurrency for thread-safe operations. Kubernetes storage functionalities are used to store collected data in the local cluster. Several service's endpoints are dedicated to the retrieval of state information.
- **The Datastore Logic service**, built with Python's Flask¹⁷ and running on Gunicorn¹⁸, handles datastore functionalities including freezing the system when a size limit is reached. The Flask-based Datastore Logic service communicates with the Go-based Data Buffer service via HTTP requests, where Flask handles high-level datastore management and forwards data and status queries to the Go service for storage and state handling. Another service function is the identification of the DT operator. Moreover it offers a webpage to monitor the subsystem.
- **The GlitchFlowApi**, developed with FastAPI and hosted on Unicorn, provides an async API layer for external interactions and relies on Datastore Logic for request handling. Data sent to the system are validated with the Pydantic framework integrated with FastAPI.

¹⁵ Go: <https://go.dev>

¹⁶ Gin: <https://gin-gonic.com>

¹⁷ Flask: <https://flask.palletsprojects.com/en/3.0.x/>

¹⁸ Gunicorn: <https://gunicorn.org>



D7.8 Final version of the thematic module for the physics domain

The entire system operates within a Kubernetes setup, with the Data Buffer as a stateful set, due to its dependence on storage, and other services as deployments which permit pod replication. Other fine-tuning configurations have been made at the web server level. Apache Airflow orchestrates these components, managing the workflow through tasks like monitoring buffer status and freezing the datastore, utilising various Airflow operators to enforce flow control and handle conditions. The Airflow DAG implementing the workflow can be seen in [Figure 9](#).

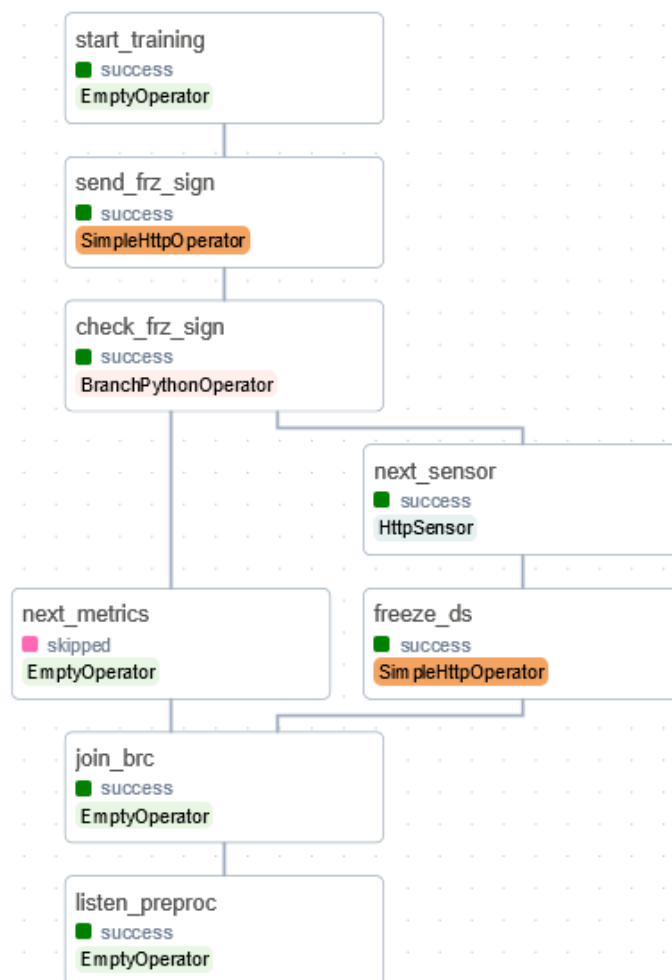


Figure 9 - Airflow DAG for training sub-system, as shown in the Airflow Dashboard

The branches in the DAG represent a different handling of the datastore state. An operator leveraging Kubernetes API is used to launch a pod processing data on runtime. The subsystem has been tested with data taken from public GW catalogues using the GWpy framework. During tests it has been observed that the time necessary for Apache Airflow to deploy the processing pod is independent of the size of the processed data.

A similar Airflow DAG for the inference sub-system is under development.



2.3.4 The Inference DT subsystem

Figure 10 shows the Container diagram (in the C4 model) of the Inference sub-system.

The Inference sub-system is still under development. The main components of the sub-system are:

- **Preprocessing_API**, common to the *Training sub-system*
- **Generative_API**, the module which employs the pre-trained GNN-model to map the transient noise in the auxiliary channels to the one observed in the strain channel (under development).
- **Veto_API**, based on Python and proprietary IGWN libraries, is an interface to the Virgo Low Latency framework (under development) .

The flow of detector data at various processing steps (grey arrows) is also shown in the figure, as well as the flow of DT artefacts (blue arrows). In this subsystem, the DT artefacts are the trained model from the *Training subsystem*, the simulated strain data, a veto decision in a data representation internal to the DT system and a veto decision in a data representation compatible with the Virgo Low Latency framework.



D7.8 Final version of the thematic module for the physics domain

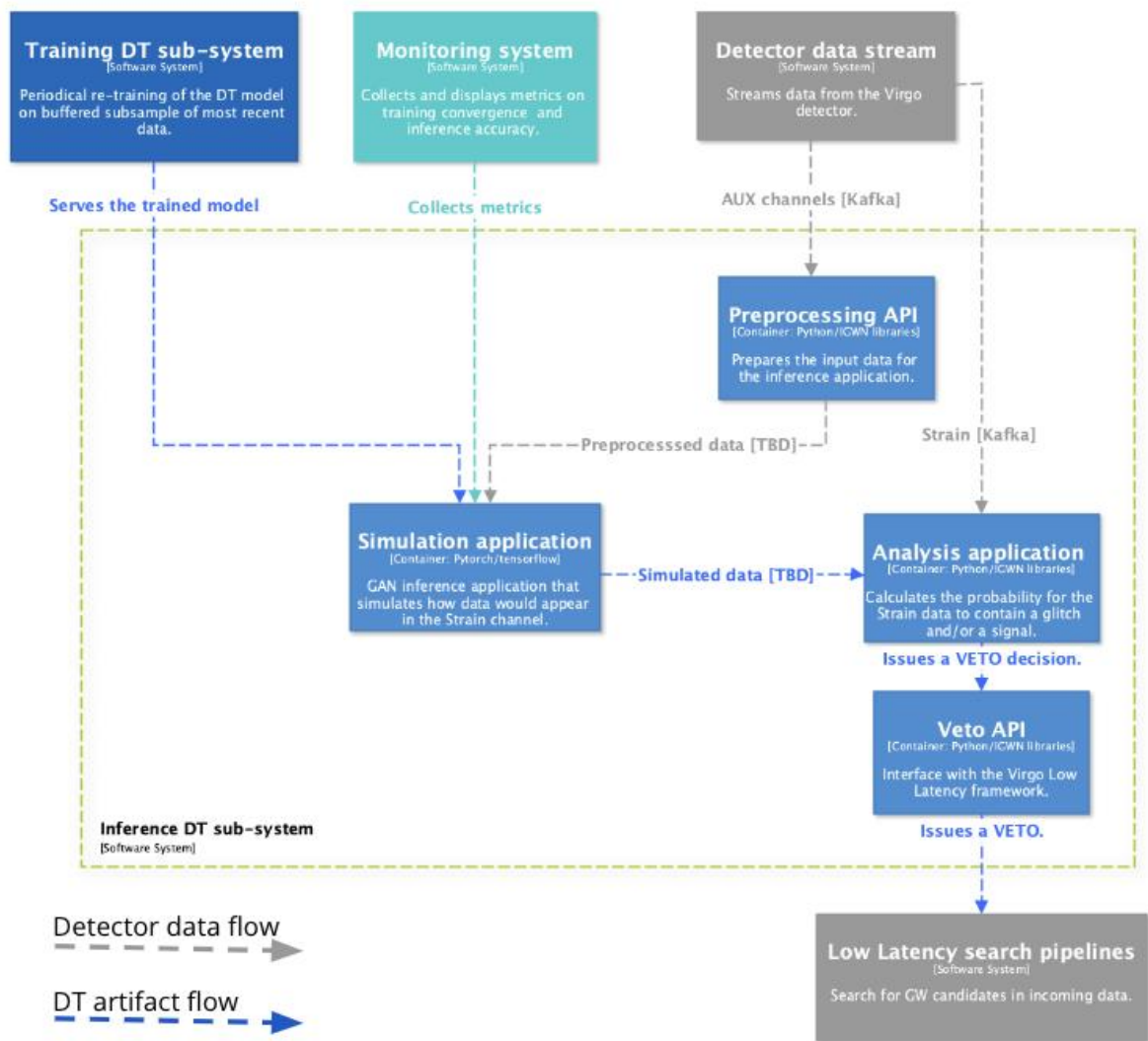


Figure 10 - Container diagram (in the C4 model) of the Inference subsystem.

2.3.5 The Virgo Data Lake

The transient noise data is being stored in the InterTwin Data Lake, which we are managing in synergy with the developers of task 5.1. The Data Lake is managed by the Rucio software, which ensures scalable and efficient data transfer and storage. Specifically, we registered two Rucio Storage Elements (RSEs): one at INFN, where the data is originally stored on tape, and one at the Vega EuroHPC¹⁹. The RSEs are part of a private Virgo Virtual Organisation (virgo.intertwin.eu), created to restrict data access to only authorised people who are part of the Virgo community. The data is transferred from the former RSE to the latter via a Transfer File System mediated by Rucio. It is then possible to use the data to develop and deploy the different modules directly on Vega, making full use of the computational resources made available by the collaboration.

¹⁹ Vega: <https://en-vegadocs.vega.izum.si/introduction/>



2.4 T7.7 Fast particle detector simulation with GAN

Task 7.7's goal is to develop the thematic module for the fast detector simulation using Generative Adversarial Networks (GANs) [R12]. This thematic module consists of two components. The simulation component that incorporates the Monte Carlo based simulation framework, Geant4²⁰, is utilized to produce the training data. The deep learning (3D Generative Adversarial Network - 3DGAN [R11]) component, which is the deep learning model developed for a specified particle detector set-up.

The two components are linked together to support the DT development in the context of WP4. More specifically, T7.7 is developing capabilities for T4.2 defined DT application that will enable the specific DT operator to:

- pre-process the data, that a Geant4 application produced and simulate particles passing through a specific detector setup
- train a GAN model on the pre-processed simulated data, with specified model input conditions (e.g. particle's entrance angle, initial energy, and type)
- use the trained GAN model during the inference step to replicate the detector's response (fast/ GAN-based simulation).

A methodology that accelerates particle detector simulations, leveraging generative deep learning methods, has already been described and is available in deliverable 7.2 (D7.2) [R1]. Our methodology uses Geant4, a software toolkit for the simulation of the passage of particles through matter, and GAN, a class of machine learning frameworks for approaching generative AI. The technical requirements have been identified, defined, and reported in detail in D7.2. Moreover, the underlying challenges of detector simulation for CERN and the High Energy Physics (HEP) community, as well as the importance of developing a DT digital twin system that integrates simulation methods with machine learning, were analysed and described.

This section provides a brief overview of CERN's digital twin application of a detector simulation, as it has already been described in detail in deliverables 7.2 and 4.2 (D4.2 [R13]). It describes the key steps, from particle simulations to event generation, and subsequent data comparison with real data. The process is explained, 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 architecture, 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 simulation application.

This application consists of two components, the component that incorporates the Geant4-based simulation framework and the deep learning component, which uses a

²⁰ GEANT4: <https://geant4.web.cern.ch>



D7.8 Final version of the thematic module for the physics domain

deep generative model developed for a specified particle detector setup. The two components are encapsulated into two main workflows, the training workflow, and the inference workflow, as illustrated in [Figure 11](#). Below, the application functionalities and their specifications included in each workflow are described.

The Geant4 simulation toolkit performs particle physics simulations based on Monte Carlo (MC) techniques. The training workflow design includes the following functionalities, which will run on HPC systems. Geant4 simulates particle interactions, producing data based on a detector-specific configuration. The produced data consists of the energy measured by the detector sensors, the properties of the initial particle, such as its type, energy, and trajectory angle with respect to the detector volume, and other metadata. The produced data, in ROOT format, are stored at data centres provided by project partners, 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 format for further preprocessing before being input into the GAN model. This conversion is currently performed using a Python script. The converted data will then be stored at the data centres. Following the ROOT to HDF5 format conversion, the HDF5 data is further pre-processed and transformed into numpy arrays, a process incorporated within the model training scripts.

A GAN is trained [\[R11\]](#) 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. Hyperparameter optimization (HPO) is also employed to improve model performance. During validation and HPO, the model-generated data and the Geant4 simulated data distributions will both be visualised. Additional validation techniques are exploited.

At the end, the training workflow stores the optimised models, selected based on validation results. The model registry where the GAN models are stored is managed by Task 6.5.

During the inference workflow process, 3D images replicating the specified detector's response are produced. Those images consist of the secondary particles' positions (x, y, z coordinates) in the detector and their corresponding energy measurements.

Data distribution comparisons are drawn between the GAN-generated data and real data (either derived from a traditional Geant4 simulation or data derived from accelerator test beams). These comparisons are essential for validating the efficacy and accuracy of the GAN-generated data.

Finally, based on the results visualised, 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 is needed, the training workflow must be re-run from the beginning. These two possible



D7.8 Final version of the thematic module for the physics domain

workflows allow for greater flexibility and adaptability in tuning the detector's responses to various particle interactions.

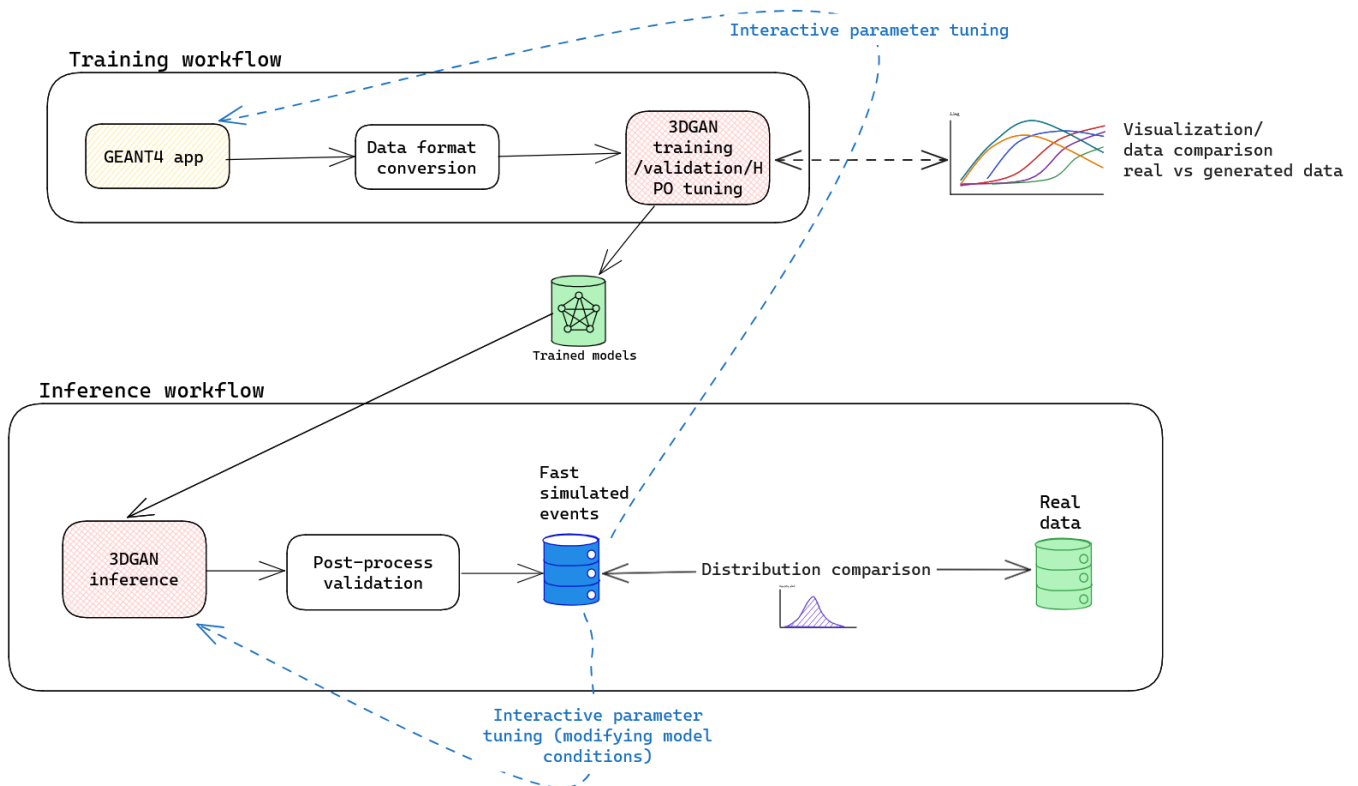


Figure 11 - Graph representation of the training and inference workflows composition (as described above) of the fast particle detector simulation DT utilising 3DGAN approach

In addition to the 3DGAN, a new component for fast calorimeter simulation is being introduced. As demonstrated in the CaloChallenge [R14] — a Fast Calorimeter Simulation Challenge aimed at promoting the development and benchmarking of fast, high-fidelity calorimeter shower generation using deep learning methods — Normalizing Flows (NF) models [R15] tend to simulate calorimeter showers more accurately than GAN-based models while maintaining competitive simulation speeds. However, this improved expressivity comes at the cost of slightly slower generation times. Notably, the CaloINN model [R16], an invertible NF neural network, strikes a strong balance between simulation quality and speed. The CaloINN model is planned to be integrated as an itwinai component.

CaloINN uses an input structure similar to 3DGAN. The detector volume is segmented into layers aligned with the direction of the incoming particle, with each layer further divided into radial and angular bins in polar coordinates. The shower is characterized by the incident energy of the incoming particle and the energy depositions in each voxel. In CaloINN, coupling layer-based normalizing flows are used to create bijective mappings between a latent space and the physical phase space. After learning this invertible

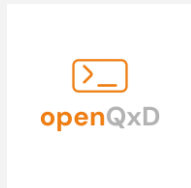
mapping by training the network, a physical phase space distribution can be sampled from the known latent space.

3 Thematic Modules / Components

In this section basic information on the software components for each thematic module is summarised, and release notes and future plans are provided.

3.1 T7.1 openQxD

Table 1 - openQxD

| | |
|---------------------------|---|
| Component name and logo |  openQxD |
| Page on interTwin website | https://www.intertwin.eu/article/thematic-module-openqxd |
| Description | Flexible code that implements advanced lattice simulation techniques on HPC systems. |
| Value proposition | The base software component necessary to simulate quantum field theories with C* boundary conditions. |
| Users of the Component | Expert users and Developers |
| User Documentation | https://gitlab.com/rcstar/openQxD/-/tree/master/doc |
| Technical Documentation | https://gitlab.com/rcstar/openQxD/-/tree/master/doc?ref_type=heads |
| Responsible | RC* Collaboration |
| Licence | GPLv2 |
| Source code | https://gitlab.com/rcstar/openQxD |

3.1.1 Release notes

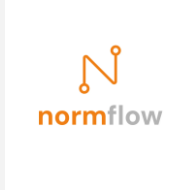
Notes are available in the repository²¹.

3.1.2 Future plans

The code will be evolved in the direction of embedding a proper CI/CD workflow for the software development process, and a FAIR data evaluation automated procedure in cooperation with WP6.

3.2 T7.1 normflow

Table 2 - normflow

| | |
|---------------------------|---|
| Component name and logo |  normflow |
| Page on interTwin website | https://www.intertwin.eu/article/thematic-module-normflow |
| Description | For applying the method of normalising flows as a generative model for lattice simulations. |
| Value proposition | This package contains utilities for the implementation of normalising flows as a generative model using Pytorch. |
| Users of the Component | Expert users and Developers |
| User Documentation | https://github.com/jkomijani/normflow |
| Technical Documentation | https://github.com/jkomijani/normflow |
| Responsible | Javad Komijani |
| Licence | MIT |
| Source code | https://github.com/jkomijani/normflow |

²¹ <https://gitlab.com/rcstar/openQxD/-/blob/master/CHANGELOG>



3.2.1 Release notes

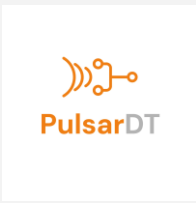
Normflow contains utilities for the implementation of the method of normalising flows as a generative model for lattice field theory. It currently supports scalar theories in any number of dimensions [R8].

3.2.2 Future plans

We are studying how the singular value decomposition can be used to construct gauge-invariant quantities which can serve as the building blocks for designing gauge-equivariant transformations of SU(N) gauge links [R9].

3.3 T7.2 PulsarDT

Table 3 - PulsarDT

| | |
|---------------------------|---|
| Component name and logo |  <p>PulsarDT</p> |
| Page on interTwin website | https://www.intertwin.eu/article/thematic-module-pulsardt |
| Description | Physics-based DT, simulation of the propagation of pulsar signals from the source to antennas and generation of synthetic data – written in Python. |
| Value proposition | The physics-based DT, to be used to generate synthetic data to train the ML classifier. This particular component is written in Python as a model of how the different aspects of the physics-based DT can be implemented, while its counterpart, PulsarDT++ implements what has already been well-established in the C++ production version. |
| Users of the Component | Expert Users and Developers |
| User Documentation | https://gitlab.com/ml-ppa/pulsardt |
| Technical Documentation | https://gitlab.com/ml-ppa/pulsardt |

D7.8 Final version of the thematic module for the physics domain

| | |
|-------------|--|
| Responsible | ML-PPA collaboration Contact: Yurii Pidopryhora yurii@mpifr-bonn.mpg.de |
| Licence | GNU AGPLv3 |
| Source code | https://gitlab.com/ml-ppa/pulsardt |

3.3.1 Release notes


The second internal version (v. 0.2) has been released. Assorted related materials, including Jupyter notebooks with use examples, are available [at GitLab](#). For a wider context and theory behind the whole ML-PPA (including detailed explanations with regard to the status of each component) one should refer to [R10]. A brief description of the current functionality is as follows: users can specify a number of model parameters, like pulsar geometry, distance to Earth, noise characteristics etc. The program produces an image, including the time-frequency domain image as if seen by a telescope plus a mask, selecting only what corresponds to the pulsar data and excluding the noise.

3.3.2 Future plans

The plans are to constantly build upon the current model, both during this final year of the project and later. Ultimately the goal of this component is to supply its counterpart, PularDT++, with algorithms and implementation strategies, so it is used as a testing ground and is constantly modified. The interstellar matter (ISM), telescope and electronics noise models are to be constantly updated. There will also be more sophisticated sources available e. g. more complex beam structures, pulsars in double systems etc.

3.4 T7.2 PulsarDT++

Table 4 - PulsarDT++

| | |
|---------------------------|--|
| Component name and logo |  PulsarDT++ |
| Page on interTwin website | https://www.intertwin.eu/article/thematic-module-pulsardt-2 |
| Description | Physics-based DT, simulation of the propagation of pulsar signals from the source to antennas and generation of synthetic data – written in C++. |



D7.8 Final version of the thematic module for the physics domain

| | |
|-------------------------|--|
| Value proposition | PulsarDT implemented in C++ in order to improve its speed and allow for parallelization, easily deployable in a singularity container. |
| Users of the Component | Expert Users and Developers |
| User Documentation | https://gitlab.com/ml-ppa/pulsardtpp |
| Technical Documentation | https://gitlab.com/ml-ppa/pulsardtpp |
| Responsible | ML-PPA collaboration Contact: Yurii Pidopryhora yurii@mpifr-bonn.mpg.de |
| Licence | GNU AGPLv3 |
| Source code | https://gitlab.com/ml-ppa/pulsardtpp |

3.4.1 Release notes

The second internal version (v. 0.2) has been released. Assorted related materials, including Jupyter notebooks with use examples, are available [at GitLab](#). For a wider context and theory behind the whole ML-PPA (including detailed explanations with regard to the status of each component) one should refer to [R10]. This component is an efficient parallel-computing capable implementation of PulsarDT, a layered architecture with a Python-based user interface on the top of various modules containerized using Singularity.

3.4.2 Future plans

Ultimately the main goal is to combine the whole ML-PPA package, both the physics-based DT (PulsarDT) and the ML-classifier (PulsarRFI_NN), in a single easily-deployable package, efficiently implemented and scalable, plus supply them with a convenient user interface. Another goal is to work on the efficient use of parallel computing.

3.5 T7.2 PulsarRFI_Gen

Table 5 - PulsarRFI_Gen

| | |
|---------------------------|---|
| Component name and logo |  PulsarRFI_Gen |
| Page on interTwin website | https://www.intertwin.eu/article/thematic-module-pulsardt-3 |



D7.8 Final version of the thematic module for the physics domain

| | |
|-------------------------|--|
| Description | Empirical DT, generating “timeframes”, 2D images (time-frequency) with various classes of pulsar and RFI signals. This DT creates various types of telescope signals by mimicking available real data rather than generating them from the physical first principles as PulsarDT does. |
| Value proposition | By using an alternative and fundamentally different method of DT creation this tool provides comparison for PulsarDT/DT++ and substitute training data for the ML classifier. |
| Users of the Component | Expert Users and Developers |
| User Documentation | https://gitlab.com/ml-ppa/pulsarrfi_gen |
| Technical Documentation | https://gitlab.com/ml-ppa/pulsarrfi_gen |
| Responsible | ML-PPA collaboration Contact: Yurii Pidopryhora yurii@mpifr-bonn.mpg.de |
| Licence | GNU AGPLv3 |
| Source code | https://gitlab.com/ml-ppa/pulsarrfi_gen |

3.5.1 Release notes

The first internal version (v. 0.1) has been released. Assorted related materials, including Jupyter notebooks with use examples, are available [at GitLab](#). For a wider context and theory behind the whole ML-PPA (including detailed explanations with regard to the status of each component) one should refer to [\[R10\]](#). The tool is well developed and already includes most of the functionalities that it should have. It can simulate a wide range of various 2D time-frequency “timeframes” very close to those that are produced based on real observations. 4 basic types of timeframes are generated: pulse, two kinds of radio-frequency interference and just noise, plus two basic types that can be combined to create “hybrid” types. Each type can be finely tuned to produce output with different signal-to-noise ratios, noise characteristics, signal parameters etc. Timeframes, generated by this component, have already been successfully used to train the ML-classifier, which then can classify the real data with high efficiency. So essentially this is a fully functional DT.

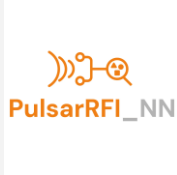


3.5.2 Future plans

The tool is already working as expected, with little space for improvement and additional functionality. It may still be updated in the future, but we do not expect any drastic changes.

3.6 T7.2 PulsarRFI_NN

Table 6 - PulsarRFI_NN

| | |
|---------------------------|--|
| Component name and logo |  PulsarRFI_NN |
| Page on interTwin website | https://www.intertwin.eu/article/thematic-module-pulsarrfi_nn |
| Description | The ML classifier. It is a CNN-based tool for the identification of various types of pulsar and RFI signals in the “timeframes”, 2D images (time-frequency). |
| Value proposition | The main tool of the framework, it plays a key role in the ML-PPA, see Figure 5 . |
| Users of the Component | Expert Users and Developers |
| User Documentation | https://gitlab.com/ml-ppa/pulsarrfi_nn |
| Technical Documentation | https://gitlab.com/ml-ppa/pulsarrfi_nn |
| Responsible | ML-PPA collaboration Contact: Yurii Pidopryhora yurii@mpifr-bonn.mpg.de |
| Licence | GNU AGPLv3 |
| Source code | https://gitlab.com/ml-ppa/pulsarrfi_nn |

3.6.1 Release notes

The first internal version (v. 0.1) has been released. Assorted related materials, including Jupyter notebooks with use examples, are available [at GitLab](#). For a wider context and theory behind the whole ML-PPA (including detailed explanations with regard to the status of each component) one should refer to [[R10](#)]. The main functionality of this component is assigning labels to 2D time-frequency “timeframes”. Each timeframe can



D7.8 Final version of the thematic module for the physics domain


be classified into one of the 4 main categories: pulse, two kinds of RFI, or empty, i.e. just noise. To do this the tool must be first trained using either real data or that generated by either physics-based DT (PulsarDT) or empirical DT (PulsarRFI_Gen).

3.6.2 Future plans

This release is stable and can be considered final. However, the main weakness of this component is dealing with data with a low signal-to-noise ratio, there is always room for improvement in this regard, and that is going to be the focus of the development efforts in the future, both in the last year of the project and beyond it. There is also work to be done with regard to better implementing distributed training. In a more distant future the plans are for the ML classifier to be able to distinguish much more than just 4 basic categories of data, ideally to detect physical properties of the signal and RFI components.

3.7 T7.3 Virgo DT Datastore Services

Table 7 - Virgo DT Datastore Services

| | |
|---------------------------|---|
| Component name and logo | Virgo DT Datastore Services including  GlitchFlow |
| Page on interTwin website | https://www.intertwin.eu/article/thematic-module-glitchflow |
| Description | DAGS for the Virgo DT. |
| Value proposition | Main component of Training subsystem |
| Users of the Component | Expert users and developers |
| User Documentation | https://github.com/intertwin-eu/DT-Virgo-dags/blob/main/Release/README.md |
| Technical Documentation | https://github.com/intertwin-eu/DT-Virgo-dags/blob/main/Release/README.md |



| | |
|-------------|---|
| Responsible | INFN Contacts: Sara Vallero (svallero@to.infn.it), Francesco Sarandrea (francesco.sarandrea@to.infn.it), Lorenzo Asprea (lorenzo.asprea@to.infn.it) |
| Licence | MIT |
| Source code | https://github.com/interTwin-eu/DT-Virgo-dags/tree/main/Release |

3.7.1 Release notes

The Virgo DT Datastore services contain the source code of the services responsible for storing gravitational waves data. Additional software tools for infrastructure testing are included. The services have been designed to be executed inside a cluster running Kubernetes. Their Docker images can be found under the repository²². The Virgo DT Datastore services comprise of:

- **Data Buffer:** a storage service written in Go. It stores gw data as JSON files and collects metadata about the infrastructure.
- **Datastore Logic:** a web application written with Flask. It performs the datastore operations using the functionalities of the Data Buffer service. The service runs inside a gunicorn server.
- **GlitchflowAPI:** API layer of the infrastructure. Currently Developed using the FastAPI Python framework.
- **GWclient:** a simple python script for sending data from gw public catalogues to the datastore. During platform tests it has been executed from a shell interacting with a pod. It relies on the gwpy framework.
- **Client:** A collection of shell scripts for sending requests to the datastore. They can be used like the gwclient script.

3.7.2 Future plans

The next step is to implement an analogous structure for the Inference subsystem, achieve full integration with the DTE and update the training API modules with more sophisticated deep learning models.

3.8 T7.7 3DGAN

In this deliverable the machine learning (ML) model, 3DGAN, is reported. In preparation for the integration of the 3DGAN model with the first version of the AI workflow toolkit, we have developed a 3DGAN version based on the PyTorch Lightning ML framework.


²² <https://hub.docker.com/repositories/romanoa77>



D7.8 Final version of the thematic module for the physics domain

Older previous versions also exist, which are based on the Tensorflow v1 and v2 framework. The model-AI workflow tool integration can be found under the repository²³.

Table 8 - 3DGAN

| | |
|---------------------------|---|
| Component name and logo |  <p>3DGAN (Deep Learning model for generation of images of calorimeter energy depositions)</p> |
| Page on interTwin website | <p>https://www.intertwin.eu/article/thematic-module-3dgan</p> |
| Description | <p>3DGAN is a generative adversarial network approach that generates High Energy Physics (HEP) calorimeter output. Calorimeters are special HEP detectors that record particles through the measurement of the energies deposited by them [R11].</p> |
| Value proposition | <p>Detector simulations allow scientists to design detectors and perform physics analyses. The simulation toolkit that has been developed and performs particle physics simulations based on Monte Carlo (MC) methods is Geant4.</p> <p>The detailed particle MC simulations are inherently slow. Simulations have a crucial role in HEP experiments, and at the same time are very resource-intensive from the computing perspective. Therefore, HEP community is highly motivated to explore fast alternatives, with deep learning based fast simulation being the most promising.</p> <p>3DGAN consists of a fast alternative to MC, with remarkable results in terms of</p> |

²³ https://github.com/intertwin-eu/itwinai/tree/3dgan_analysis/use-cases/3dgan



| | |
|-------------------------|--|
| | speed up. 3DGAN was the first effort where the detector output was generated employing three dimensional convolutions, an approach for retaining correlations in all three spatial dimensions [R11]. |
| Users of the Component | Expert Users and Developers |
| User Documentation | https://github.com/interTwin-eu/DetectorSim-3DGAN/tree/main |
| Technical Documentation | https://github.com/interTwin-eu/DetectorSim-3DGAN/tree/main |
| Responsible | CERN & CNRS Contacts: Vera Maiboroda (vera.maiboroda@cern.ch), Matteo Bunino (matteo.bunino@cern.ch), Kalliopi Tsolaki (kalliopi.tsolaki@cern.ch), Sofia Vallecorsa (sofia.vallecorsa@cern.ch), David Rousseau (rousseau@ijclab.in2p3.fr) |
| Licence | MIT |
| Source code | https://github.com/interTwin-eu/DetectorSim-3DGAN/tree/main |

3.8.1 Release notes

The fast particle detector simulation with GAN thematic module consists of two inseparable components as we have already discussed in [section 2.4](#), as well as in D7.2. These two components are the machine learning framework and the particle simulation framework. An implementation of the 3DGAN approach has been developed and a more detailed description follows. The code is available on GitHub and it has been tested and run on a single Linux node using GPU infrastructure.

3DGAN is being trained to produce images similar to the ones that are produced by Monte Carlo simulations. As the calorimeter detectors consist of layers of cells, those cells are modelled as monochromatic pixelated images with the cell energy depositions being the 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 the real images. Each network is being trained using 3-dimensional convolution layers to represent the 3 spatial dimensions of the calorimeter images.



D7.8 Final version of the thematic module for the physics domain

The generator network implements stochasticity through a latent vector drawn from a Gaussian distribution. The generator input includes the primary particle's initial energy and the angle at which it entered the detector, concatenated to the latent vector. The generator network then maps the input to a layer of linear neurons followed by 3D convolutional layers. The discriminator input is an image while the network has only 3D convolutional layers. Batch normalisation is performed between the layers and the LeakyRelu²⁴ activation function is used for the discriminator layers while the Relu¹³ activation function is used for the generator layers. The model's loss function is the weighted sum of individual losses concerning the discriminator outputs and domain-related constraints, which are essential to achieve high-level agreement over the very large dynamic range of the image pixel intensity distribution in a HEP task. The training of this model was inspired by the concept of transfer learning. This means that the 3DGAN was trained first for images in a limited energy range and after the GAN converged, the same trained model was further trained with the data from the whole available energy range.

Currently, the 3DGAN training workflow consists of several other processes, the data pre-processing process, the model definition, and the training process. The validation and hyperparameter optimization processes are under research.

The dataset used for studying and developing the 3DGAN model [R11] ([public dataset](#)) consists of calorimeter 3D images/arrays of energy depositions with shape 51x51x25, which represent the particle showers. These images were created from simulations performed with Geant4 software. The output of the Geant4 simulation is ROOT²⁵ files, which need to be converted into a ML-friendly format HDF5 in order to train the model.

The preprocessing is responsible for preparing (cleaning, scaling, etc.) and converting into a suitable format (HDF5 format) the simulated data created by Geant4 (ROOT format). It also encodes the input information such as the calorimeter's geometry identifier, the energy of the primary particle initiating the shower, the angle at which the particle enters the detector, and also its type and/or initial position. The pre-processed data are then passed to the GAN model (currently developed using Tensorflow v1 and v2²⁶, as well as in PyTorch Lightning²⁷) for training. The validation process will verify the performance through a set of physics-motivated steps, both at the single image quality level and at the sample level.

During pre-processing, simulation inputs are defined and encoded, i.e. the detector geometry, the energy and angle of the incoming particle. The performance of the model is evaluated during validation processes through the creation of histograms describing particle shower observables. Shower observables are among others, total energy distribution (sum of all cell energy deposits), cell energy deposits distribution, longitudinal profile which represents the energy deposited by a shower as a function of the depth of

²⁴ [https://en.wikipedia.org/wiki/Rectifier_\(neural_networks\)](https://en.wikipedia.org/wiki/Rectifier_(neural_networks))

²⁵ ROOT: <https://root.cern/>

²⁶ Tensorflow: <https://www.tensorflow.org/>

²⁷ PyTorch Lightning: <https://lightning.ai/>



D7.8 Final version of the thematic module for the physics domain

the calorimeter and lateral profile which represents the energy density distribution as a function of the radius of the calorimeter. Moreover, the physics-based validation process will include accuracy verification of those key distributions' first moments and precise evaluation of the tails of distributions that usually require larger amounts of samples. The original data coming from Geant4 and the 3DGAN data distributions will be compared during this evaluation process.

Concerning the particle simulation framework, there have been testbeds developed at CERN that incorporate different ML models than the 3DGAN. Our current focus doesn't include integrating the 3DGAN model in the simulation framework that uses the Geant4 environment. An example of the use of ML techniques for the fast detector simulation and how to incorporate inference libraries into Geant4 is the Par04 example developed by the Geant4 community and can be found on CERN Gitlab²⁸. The ML model used in this example is a Variational Autoencoder (VAE), trained externally in Python on full Geant4 detector simulation response data.

The training of 3DGAN implemented in PyTorch Lightning and itwinai²⁹, was conducted on JSC computing resources, utilizing restricted energy range (100-200 GeV) data for 130 epochs and full energy range (2-500 GeV) data for 30 epochs per half of the dataset due to memory constraints. The model conditioning includes theta angles (60°-120°) and primary energy levels (2-500 GeV), ensuring uniform distribution across the dataset. The trained weights are available for inference. The dataset consists of electron showers in a 51×51×25 voxel grid.

The initial prototype was implemented in TensorFlow/Keras, and its parameterization was precisely replicated during the transition to PyTorch Lightning. While the PyTorch version does not perform as well as the Keras prototype, it still captures most underlying event generation patterns effectively.

Before inference, the training loss was closely monitored, with the best PyTorch weights selected based on validation loss. Both TensorFlow and PyTorch-trained weights can be used for inference, providing flexibility. The model's performance was evaluated using physics analysis plots, including 2D projections, which compare GAN-generated events to Geant4 (G4) simulated data. The results, available for both frameworks, highlight the model's capability to reproduce complex shower patterns (see [Figure 12](#) and [Figure 13](#)).

²⁸ <https://gitlab.cern.ch/geant4/geant4/-/tree/master/examples/extended/parameterisations/Par04>

²⁹ <https://github.com/interTwin-eu/itwinai>



D7.8 Final version of the thematic module for the physics domain

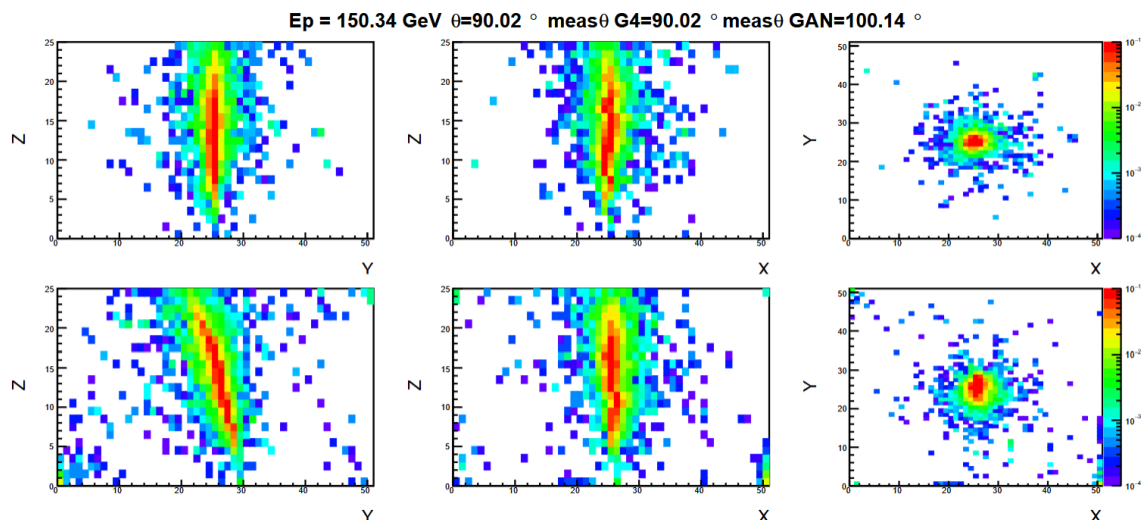


Figure 12 - 2D Projections of Electromagnetic Showers Generated Using PyTorch-Trained 3DGAN Weights. The projections are shown for the **XY, XZ, and YZ** planes at **$E_p = 150.34 \text{ GeV}$** and **$\theta = 90.02^\circ$** , comparing Geant4 simulated data (top row) with GAN-generated events (bottom row). The color scale represents the normalized energy deposition intensity.

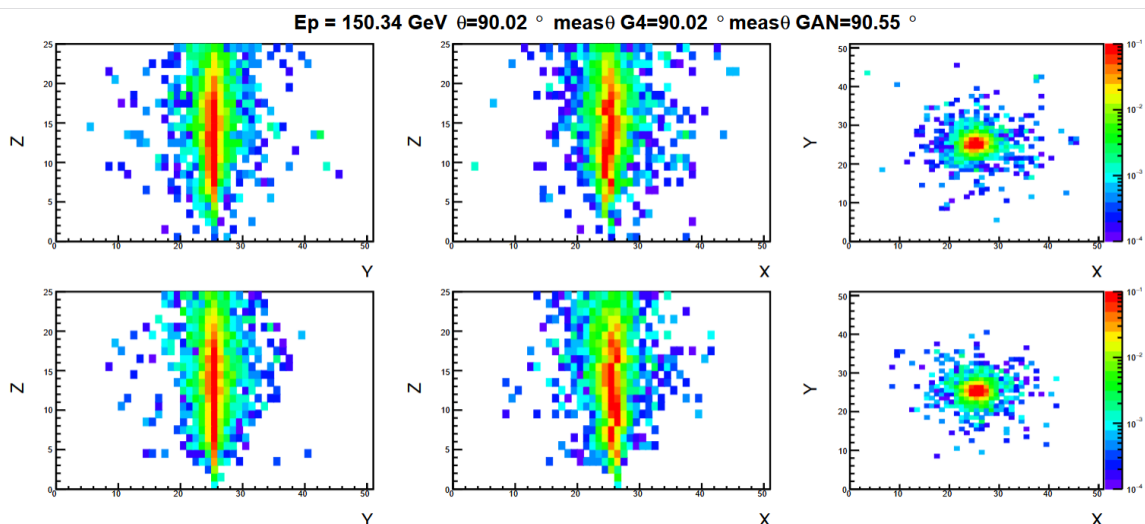


Figure 13 - 2D Projections of Electromagnetic Showers Generated Using TensorFlow-Trained 3DGAN Weights. The projections are shown for the **XY, XZ, and YZ** planes at **$E_p = 150.34 \text{ GeV}$** and **$\theta = 90.02^\circ$** , comparing Geant4 simulated data (top row) with GAN-generated events (bottom row). The color scale represents the normalized energy deposition intensity.

The TensorFlow model exhibits a tighter, more concentrated core in the projections, closely matching the G4 simulation, particularly in the XZ and YZ planes. The energy deposition is more symmetric with smoother gradients around the core. The PyTorch model shows increased dispersion, with the core appearing broader and less uniform, especially in the YZ projection. This suggests the PyTorch model current state captures the overall shower pattern but struggles with fine-grained details. Though, both models capture the high-intensity core (red regions).

D7.8 Final version of the thematic module for the physics domain

While the PyTorch model effectively replicates the general shower structure, the TensorFlow model demonstrates superior performance in terms of spatial accuracy, angular precision, and smoother energy deposition profiles. This aligns with the observation that the PyTorch version, despite capturing the main patterns, currently underperforms compared to the Keras prototype, especially in the angular reconstruction and shower shape fidelity. Further tuning of the PyTorch model — adjusting learning rates, loss functions, or regularization techniques — could help bridge this performance gap.

3.8.2 Future plans

The developments of the thematic module are almost completed from the aspect of the 3DGAN component. Studies might be conducted on existing solutions for parallel training and hyperparameter optimization, but are not essential. The implementation of the CaloINN component is in progress. Simultaneously, the validation of the CaloINN algorithm on different detector simulations is also underway. In collaboration with the HEP community, different validation techniques are studied with the goal of identifying the technique most aligned with the needs of the fast detector simulation use case.

4 Conclusions

This report provides an overview of the final version of the thematic modules for the physics domain. It presents the status of the developments for each thematic module in T7.1 Lattice QCD simulations, T7.2 Noise simulation for radio astronomy, T7.3 GAN-based thematic modules to manage noise simulation, low-latency de-noising, and veto generation for gravitational waves, and T7.7 Fast particle detector simulation with GAN.

The development of all thematic modules has significantly advanced, and in some cases can be considered completed in the sense that the released software is stable and the main development goals as related to the corresponding use cases have been achieved (but there still may be small improvements and updates made in this last year of the project).

Basic information about the 8 thematic modules in the physics domain has been presented in this report through a common template, along with release notes and future plans. It is important to mention that, even though this document presents the final version of the thematic modules, additional developments are expected in the remainder of the project, driven among other things by the integration with the DTs applications and the rest of the DTE. The final developments will be presented in the integration report D7.10.

In terms of integration current activities, the physics thematic modules are already using some infrastructure (WP5) and core (WP6) components from the project and stronger integration is planned for the remainder of the project.



D7.8 Final version of the thematic module for the physics domain

The normflow package of T7.1 has been integrated with the SQAaaS module (T6.2) and has a Silver SQAaaS badge at present. In T7.7 workflow execution by OSCAR (T6.1) was integrated to showcase the inference process execution. Several thematic modules of WP7's physics domain are integrating the itwinaï framework (T6.5) capabilities. In particular, in T7.1 integration is focused on the distributed training and logging features provided by itwinaï, with the lattice use case now available in the itwinaï [repository](#). See [Figure 3](#) for an overview of the structure of WP7.1 and its integration with the other WPs. In T7.2 there is progress towards using itwinaï distributed training with PulsarRFI_NN. In T7.7 the training and inference processes of the model have been integrated with the itwinaï framework, and can be found under the following [repository](#). The integration of DTE modules enabled Particle Detector DT development by offering tailored services and platforms designed to optimize data and AI workflow management. The AI workflows framework, facilitated by itwinaï, supports efficient model training and testing on FZ Juelich resources. Along with 3DGAN, another detector simulation generative model with a different architecture, CaloINN, is currently being integrated to the itwinaï framework.

At the infrastructure level (WP5), evaluation of the components on the project testbeds (e.g. DESY and Vega HPC centers) started. Interactions with WP5 are also ongoing as part of the data needed by the DTs has been made available from the project data lake (T5.2). For T7.1, a small test data set was given to DESY for data lake testing. Now file transfer over an FTS connection between DESY and CESSGA is being tested as a precursor to the establishment of a Lattice Data Lake. In T7.2 the data lake / Rucio was tested by providing a small test dataset to DESY and Vega HPC centers. Also, for both of them the deployment of the ML-PPA components was successfully tested. In T7.3, some preliminary integration was performed with the DTE core modules by moving a subset of the proprietary data, the auxiliary channels of the Virgo interferometer, on the Vega HPC, which is part of the data lake. This was achieved by creating a dedicated Virtual Organisation (VO) in Rucio, which allows the transfer of the data and makes it visible exclusively to the people affiliated with the Virgo experiment. In T7.7, project-related data is added to and made accessible via the project's data lake, to ensure streamlined data availability, while federated computing capabilities were enabled through InterLink (T5.1), further enhancing inference tasks.

To conclude, the key activities foreseen as next steps will focus on:

- improvement and, where necessary, finalization of the thematic modules capabilities required to fully support the physics use cases and DT applications from WP4;
- strengthening and completing integration with the interTwin DTE core (WP6) and infrastructural (WP5) components.



5 References

| Reference | |
|------------|--|
| No | Description / Link |
| R1 | interTwin D7.2 Report on requirements and thematic modules definition for the physics domain (Version Final). K. Tsolaki et al., (2023). DOI: 10.5281/zenodo.8036996 |
| R2 | interTwin D7.4 First version of the thematic modules for the physics domain (V1 Under EC review). G. Sinha Ray et al., (2023). DOI: 10.5281/zenodo.10224276 |
| R3 | interTwin D7.6 Update report on requirements and thematic modules functionalities for the physics domain (V1 Under EC review). S. Vallero et al., (2024). DOI: 10.5281/zenodo.13382890 |
| R4 | openQ*D code: a versatile tool for QCD+QED simulations. I. Campos, P. Fritzsche, M. Hansen, M. K. Marinkovic, A. Patella, A. Ramos & N. Tantalo The European Physical Journal C , 80, Article number: 195 (2020) DOI: 10.1140/epjc/s10052-020-7617-3 |
| R5 | Flow-based generative models for Markov chain Monte Carlo in lattice field theory. M. S. Albergo, G. Kanwar, and P. E. Shanahan Phys. Rev. D 100, 034515 (2019) DOI: 10.1103/PhysRevD.100.034515 |
| R6 | Normalizing Flows: An Introduction and Review of Current Methods. Ivan Kobyzev; Simon J.D. Prince; Marcus A. Brubaker. IEEE Transactions on Pattern Analysis and Machine Intelligence (Volume: 43, Issue: 11, 01 November 2021) DOI: 10.1103/PhysRevD.100.034515 |
| R7 | Efficient modeling of trivializing maps for lattice Φ^4 theory using normalizing flows: A first look at scalability. Luigi Del Debbio, Joe Marsh Rossney, and Michael Wilson Phys. Rev. D 104, 094507 – Published 15 November 2021 DOI: 10.1103/PhysRevD.104.094507 |
| R8 | Generative models for scalar field theories: how to deal with poor scaling? Javad Komijani and Marina K. Marinkovic; The 39th International Symposium on Lattice Field Theory, 2022, Bonn, Germany; DOI: 10.48550/arXiv.2301.01504 |
| R9 | Normalizing flows for SU(N) gauge theories employing singular value decomposition; Javad Komijani and Marina K. Marinkovic; The 41th International Symposium on Lattice Field Theory, 2024, Liverpool, UK; DOI: 10.48550/arXiv.2501.18288 |
| R10 | ML-based Pipeline for Pulsar Analysis (ML-PPA); Kazantsev, A., Oelkers, T., Pidopryhora, Y., Saha, T., Trattner, H., and Heßling, H.; |



| | |
|------------|--|
| | https://gitlab.com/ml-ppa/gitlab-profile/-/blob/main/PUNCH_interTwin_project.pdf |
| R11 | Fast simulation of a high granularity calorimeter by generative adversarial networks. Khattak, G.R., Vallecora, S., Carminati, F. <i>et al.</i> Eur. Phys. J. C 82, 386 (2022). DOI: 10.1140/epjc/s10052-022-10258-4 |
| R12 | Generative Adversarial Networks. Ian J. Goodfellow et al. (2014) DOI: 10.48550/arXiv.1406.2661 |
| R13 | interTwin D4.2 First Architecture design of the DTs capabilities for High Energy Physics, Radio astronomy and Gravitational-wave Astrophysics (Version Final) A. Manzi et al., (2023) DOI: 10.5281/zenodo.8321133 |
| R14 | CaloChallenge 2022: A Community Challenge for Fast Calorimeter Simulation; C. Krause et al. (2024) DOI: 10.48550/arXiv.2410.21611 |
| R15 | Normalizing Flows: An Introduction and Review of Current Methods; I. Kobyzev, S. J.D. Prince, and M. A. Brubaker (2020) DOI: 10.48550/arXiv.1908.09257 |
| R16 | Normalizing Flows for High-Dimensional Detector Simulations; F. Ernst et al. (2025) DOI: 10.48550/arXiv.2312.09290 |

