



interTwin

D7.6 Updated report on requirements and thematic modules functionalities 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


Particle detector simulation, Lattice QCD, Radio Astronomy, Gravitational Waves, High Energy Physics, Digital Twin

interTwin co-designs and implements the prototype of an interdisciplinary Digital Twin Engine (DTE). The developed DTE will be an open source platform that includes software components for modelling and simulation to integrate application-specific Digital Twins. InterTwin WP7 provides the aforementioned sets of software components, called thematic modules, for the use cases defined in WP4.

The current document consists of an update of the D7.2 deliverable, containing a high-level description of the physics domain use cases and of the related designed thematic modules, specifying their stage of development. This deliverable also includes a description of the data and computing requirements of the thematic modules, which will need to be available in the DTE infrastructure designed in WP5 and in the core modules developed in WP6.



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

Document Description			
D7.6 Updated report on requirements and thematic modules functionalities for the physics domain			
Work Package number WP7			
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	INFN		
Document link	https://documents.egi.eu/document/4097		
DOI	https://doi.org/10.5281/zenodo.13382891		
Author(s)	<ul style="list-style-type: none"> • Kalliopi Tsolaki (CERN) • Sofia Vallecorsa (CERN) • David Rousseau (IN2P3) • Isabel Campos (CSIC) • Yurii Pidopryhora (MPG) • Sara Vallero (INFN) • Francesco Sarandrea (INFN) • Lorenzo Asprea (INFN) • Gaurav Sinha Ray (CSIC) 		
Reviewers	<ul style="list-style-type: none"> • Andrea Manzi (EGI) • Diego Ciangottini (INFN) 		
Moderated by:	Andrea Anzanello (EGI)		



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

Approved by	Daniele Spiga (INFN) on behalf of TCB
--------------------	---------------------------------------

Revision History			
Version	Date	Description	Contributors
V0.1	01/06/2024	ToC, adapted from D7.2	Francesco Sarandrea (INFN), Lorenzo Asprea (INFN)
v0.2	31/07/2024	Updated sections 1, 2, 3, and 5, removed section 4	Francesco Sarandrea (INFN), Lorenzo Asprea (INFN), Kalliopi Tsolaki (CERN), Yurii Pidopryhora (MPG), Gaurav Sinha Ray (CSIC)
v0.3	02/08/2024	Version ready for internal review	Francesco Sarandrea (INFN), Lorenzo Asprea (INFN)
v0.4	21/08/2024	Internal review	Andrea Manzi (EGI), Diego Ciangottini (INFN)
v0.5	23/08/2024	version ready for TCB review	Francesco Sarandrea (INFN), Lorenzo Asprea (INFN)
v0.6	26/08/2024	version reviewed by TCB	Daniele Spiga (INFN)
v0.7	26/08/2024	version ready for QA	Francesco Sarandrea (INFN), Lorenzo Asprea (INFN)
V1.0	28/08/2024	Final	

Terminology / Acronyms	
Term/Acronym	Definition
GW	Gravitational Wave
QCD	Quantum Chromodynamics
GAN	Generative Adversarial Network
HEP	High Energy Physics
MC	Monte Carlo



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

HL-LHC	High Luminosity - Large Hadron Collider
GNN	Generative Neural Networks
DAG	Directed Acyclic Graph

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



Table of Contents

1	Introduction	9
1.1	Scope	9
1.2	Document Structure	9
2	Progress in the design of thematic modules in the physics domain	10
2.1	T7.1 Lattice QCD simulations and data management	10
2.1.1	Advanced Data management for Lattice QCD	10
2.1.2	Generative models using Machine Learning	12
2.2	T7.2 Noise simulation for radio astronomy	14
2.2.1	Empirical block	18
2.2.2	Theoretical block	19
2.2.3	C++ development block	20
2.3	T7.3 GNN-based thematic modules to manage noise simulation, low-latency denoising and veto generation for Gravitational Waves	22
2.3.1	Use-case description	22
2.3.2	High-level architecture of the DT implementation	23
2.3.4	The Training DT subsystem	25
2.3.5	The Inference DT subsystem	26
2.4	T7.7 Fast particle detector simulation with GAN	27
2.4.1	Use case overview	28
2.4.2	3DGAN component implementation	31
3	Requirements for the thematic modules in the physics domain	35
3.1	Input data requirements	35
3.2	Expected data volume	35
3.3	DTE Infrastructure and Core components integration requirements	36
3.3.1	DTE Infrastructure components	36
3.3.2	DTE Core components	36
3.3.3	Computing requirements	37
4	Conclusions	38
5	References	39



Table of Figures

Figure 1 - The modules that make up the openQxD software	11
<i>Figure 2 - Graphical representation of the classical generation of configurations using Monte Carlo algorithms</i>	<i>13</i>
Figure 3 - 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.	13
Figure 4 - Examples of four main types of data “time frames” in a pulsar-observation data set ...	16
Figure 5- Distribution of the four main types of time frames in a typical pulsar data set. A.....	17
Figure 6 - Three parallel interacting subprojects of the task	18
Figure 7 - General outline of the DT structure.	19
Figure 8 - Layered software architecture of the framework ML-PPA	20
Figure 9 - Diagram of the final software product (in the C4 model).....	21
Figure 10 - High-level architecture of the DT	22
Figure 11 - System Context diagram (in the C4 model) of the DT for the veto pipeline.	23
Figure 12 - Airflow DAG for training sub-system, as shown in the Airflow Dashboard	24
Figure 13 - Container diagram (in the C4 model) of the Training subsystem.....	26
Figure 14 - Container diagram (in the C4 model) of the Inference subsystem	27
Figure 15 - Simulation of a particle shower created by the primary particle entering the detector volume, interacting with its material.....	28
Figure 16 - Detailed (full) particle simulation with Geant4 (left) and fast particle simulation using generative ML techniques (right) (R3)	30
Figure 17 - Fast particle detector simulation using ML techniques high level workflow composition	31
Figure 18 - 3DGAN model architecture	32
<i>Figure 19 - Fast particle detector simulation using ML techniques high level workflow composition and its connections with other work packages’ components in C4 format diagram</i>	<i>33</i>



Executive summary

The present deliverable D7.6 consists of an update on the previous D7.2 and D7.5 deliverables on requirements and thematic modules definition for the physics domain Digital Twins applications. It is a collective document written by scientists working on the development of the physics domain thematic modules. This document describes how the different use cases derive the modules that are under development and how such modules are being integrated into the interTwin Digital Twin Engine (DTE). The technical requirements for the different modules were consolidated and are presented in this document. The use-case specific requirements are reported, focusing on the following aspects: input data requirements, expected data volume, core components integration and computing requirements.



1 Introduction

1.1 Scope

This document gives an overview of the physics domain thematic modules (T7.1, T7.2, T7.3, T7.7) and their requirements developed by the interTwin project.

The thematic modules will enhance the capabilities of the core engine running the Digital Twins by adding functionalities to several fields, such as:

- Machine Learning (ML) based analysis for QCD simulation configurations and for time series
- Generative Neural Network (GNN) based analysis for Gravitational Wave (GW) Interferometer data, noise signals classification, noise analysis, de-noising, and veto generation
- Generative Adversarial Networks (GAN) based Lattice QCD configurations generation, noise simulation, particle detector simulation
- Particle physics validation techniques capable of assessing different aspects of model performance
- Fast simulation of High Energy Physics detectors

The design of the thematic modules will follow the specific requirements provided by the WP4 – Technical co-design and validation with research communities, where the use cases are being defined. Additionally, specific requirements for each thematic module will be defined and listed in this document.

1.2 Document Structure

The document is an updated version of the previous **D7.2 Report on requirements and thematic modules definition for the physics domain**. Most of the information are therefore common to the two documents, refer to Sections:

- 2.1.1 (removal of a C4 graph representing the data lake architecture, added details on ILDG integration and normflow developments),
- 2.2 (updated figures, and detailed design in subsection 2.2.1, 2.2.2 and 2.2.3)
- 2.3 (added the latest design of the components of the high-level architecture),
- 2.4.1 (updated the diagram of the high level workflow composition),
- 2.4.2 (included diagram of 3DGAN model architecture),
- 3 (updated the general data and computing requirements),

for significant changes and updates.

The document is structured in the following manner:



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

- **Section 2** includes the description of the current design of the thematic modules developed for each individual physics use case, starting with the lattice QCD simulations and data management (T7.1), the noise simulation for radio astronomy (T7.2), the GAN-based thematic modules to manage noise simulation, low-latency de-noising and veto generation for Gravitational Waves (T7.3), and closing with the fast particle detector simulation with GAN (T7.7).
- **Section 3** collects the requirements for the thematic modules described in section 2. It is divided into several categories: input and output data requirements, data volumes, databases, computing, OS and execution frameworks, machine learning requirements, real-time data acquisition and processing, data formats, software stack, visualisation, and data sharing.
- The deliverable ends with **section 4** that draws conclusions from the perspective of the physics use cases: it summarises studies and analyses performed during the first eight months of project's life, that resulted in the first design of the thematic modules and their technical requirements.

2 Progress in the design of thematic modules in the physics domain

2.1 T7.1 Lattice QCD simulations and data management

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 modern Machine-Learning techniques can improve lattice calculations.

2.1.1 Advanced Data management for Lattice QCD

Lattice QCD simulations are executed at 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

¹ <https://slurm.schedmd.com>



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

configurations, and the statistical analysis of those measurements. All of these steps, especially the generation of configurations, can be highly computationally intensive.

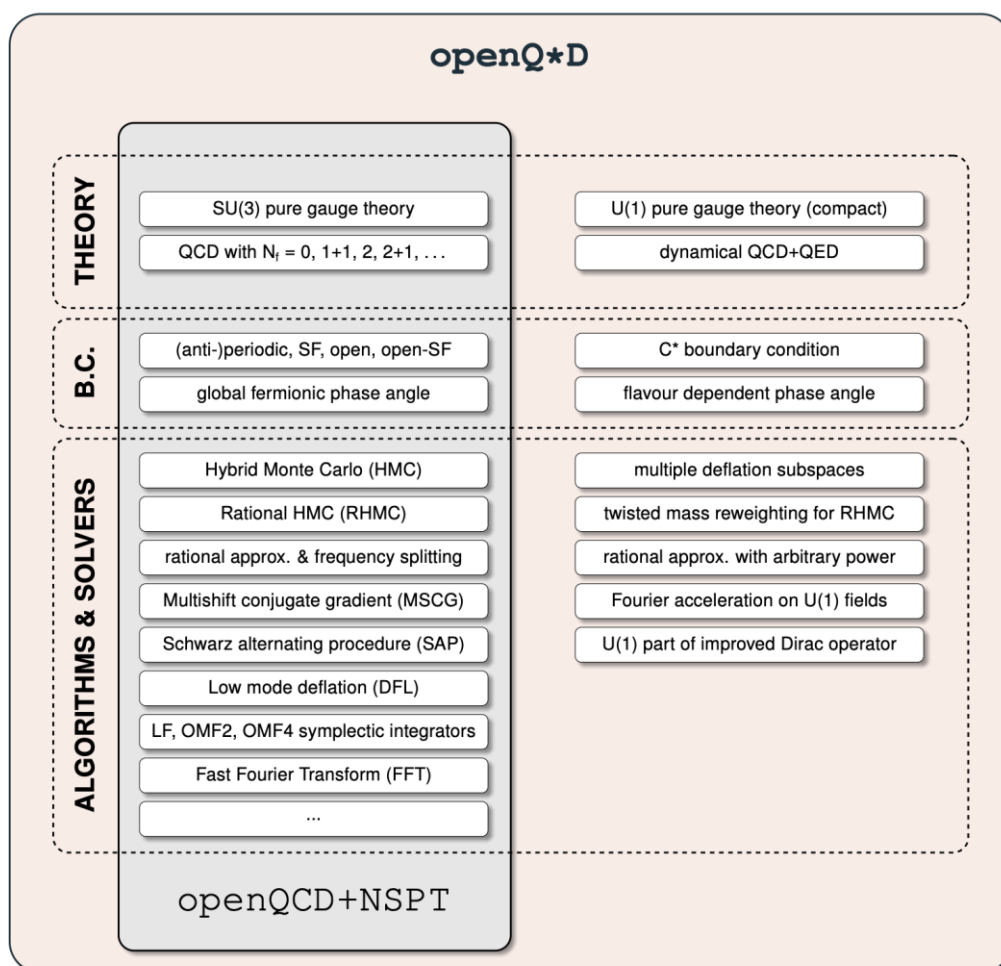


Figure 1 - The modules that make up the openQxD software

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 [R11]. A concise description of the software is given in section 3.1 of D7.2[R16] along with links to further technical documentation. The modules that make up the code are shown in **Figure 1**. It is under active development though most of this work is not being done as part of the interTwin project. Within interTwin our work is focussed on incorporating the recent OCLint integration of WP6's SQAaaS framework, into our prototype openQxD development workflow.

In previous deliverables, we described some of the issues encountered by lattice researchers when trying to store and access their data [R16]. We argued that lattice configurations should be made more easily available to the members of a collaboration.

² <https://gitlab.com/rcstar/openQxD>



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

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 [R17] we described our efforts relating to testing and benchmarking the DataLake prototype with real and toy lattice data [R17]. After providing feedback to the DataLake developers it was decided that the Lattice group should get its own VO in order to satisfy its particular read/write permission specifications.

Since the last update there have been several discussions between the ILDG³ and interTwin's WP5 with regards to the scope of potential interoperability between the data lake and the ILDG Metadata/File catalogues. The current proposal is to extend the ILDG catalogue to support the data lake as a possible source of data. In this scenario, for files (i.e. lattice gauge field configurations) stored in the data lake the ILDG would record a URL that encodes the information a user needs to interact with the data lake. The data lake, appearing to the user as a Rucio instance, would not know to which dataset any file belongs, it would carry out operations only at the level of individual files.

The ILDG can support multiple locations for each file. Each location is recorded as a URL. DataLake support would be included by letting ILDG accept URLs that look like so:

➤ `rucio://rucio.example.org:8443/LatticeQCD/RAY:my_test_file`

where

- `rucio.example.org` identifies the Rucio endpoint,
- `8443` is the port on which Rucio is listening,
- `LatticeQCD` is the name of the Virtual Organisation, and
- `RAY:my_test_file` is the file's Rucio Data Identifier (`RAY` is the file's scope and `my_test_file` is the filename).

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) induce 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 [R12] and

³ <https://hpc.desy.de/ldg/>



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

a block diagram illustrating the method is shown in **Figure 2**). The underlying idea is using Machine Learning techniques to map the theory of interest to a “simpler” theory, easier to simulate. This approach has the potential to become more efficient than traditional sampling especially when the concept of transfer learning is utilised.

However, the costs associated with the (highly complex) sampling from the path integral, are transferred to the training of a model. The question under investigation is therefore how expensive it is to train a model compared with making a classical Monte Carlo simulation.

Several papers, such as [R13], 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 3**) drop fast as the lattice size increases unless better architectures and methods are achieved (see for example [R14]).

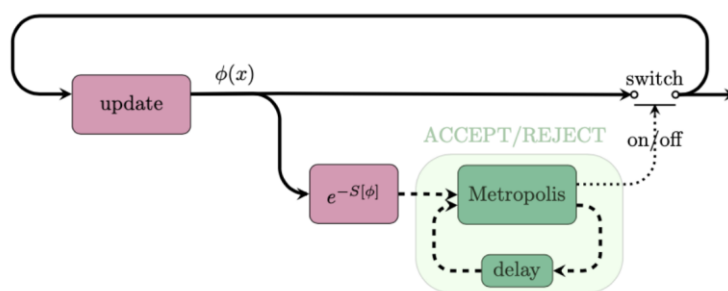


Figure 2 - Graphical representation of the classical generation of configurations using Monte Carlo algorithms

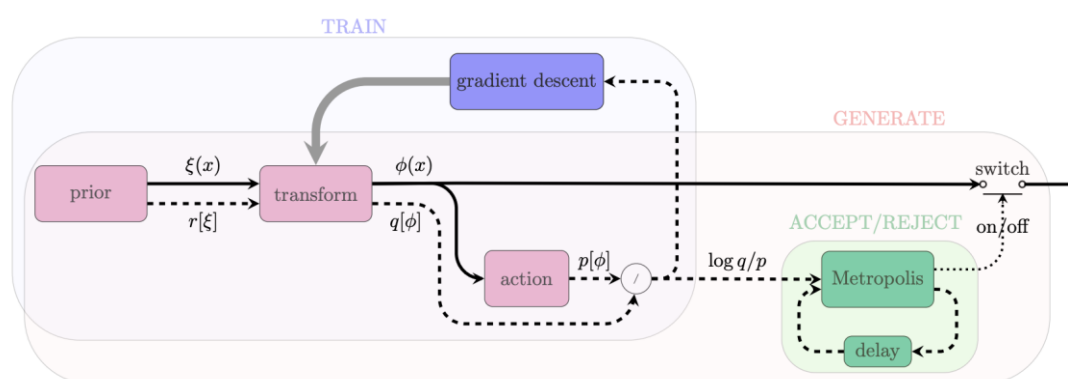


Figure 3 - Graphical representation of the Normalising Flows method including a correcting accept/reject step to account for the fact that the model cannot be perfectly trained.

The purpose of this work is designing better architectures for Machine Learning models so that the acceptance rates become reasonable (~50% or more) as the volume of the lattice increases. The requirements in terms of resources are not as in the classical Monte Carlo simulation since the methodology is still at the proof of concept level.

A typical Jupyter notebook of the type required for these studies can be found in this reference [R15].

Through the development of normflow⁴, we have shown that Machine Learning can speed-up field configuration generation with scalar theories on smaller low dimensional lattices in certain regions of parameter phase space.

2.2 T7.2 Noise simulation for radio astronomy

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 new so-called Square Kilometre Array⁵ "pathfinders", such as South African MeerKAT⁶ or Australian ASKAP⁷, the data acquisition rate increases enormously, these tools can easily produce several petabytes of raw data per week⁸. 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 periodic⁹ sources like pulsars. Since the new telescopes are much more sensitive, they can systematically probe the transient radio sky, which currently is generally

⁴ <https://github.com/jkomijani/normflow>

⁵ SKAO: <https://www.skao.int/en>

⁶ MeerKAT Radio Telescope: <https://www.sarao.ac.za/gallery/meerkat/>

⁷ ASKAP-radio telescope: <https://www.csiro.au/en/about/facilities-collections/atnf/askap-radio-telescope>

⁸ 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).

⁹ 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.



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

unknown. Such studies are very important, since it is believed that the transients¹⁰ 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 ideal¹¹.

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 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).

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 (developed in T4.3), 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.

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

¹⁰ 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.

¹¹ 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 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.



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

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.

We separate two main types of RFI: narrow-band RFI (NBRFI) that is present only on some frequencies of the observation band and broad-band RFI (BBRFI) that covers the whole observation band. Examples of the 4 basic time-frame types are shown in **Figure 4**. Thus, the basic task of the ML classifier is to label each small fragment of the data stream (a time frame) as one of the standard categories: signal, none (noise), NBRFI and BBRFI. In reality a few extra labels need to be introduced: "other" (something that cannot be classified into known categories) and mixed ones (e.g., signal with NBRFI, or NBRFI with short bursts of BBRFI).

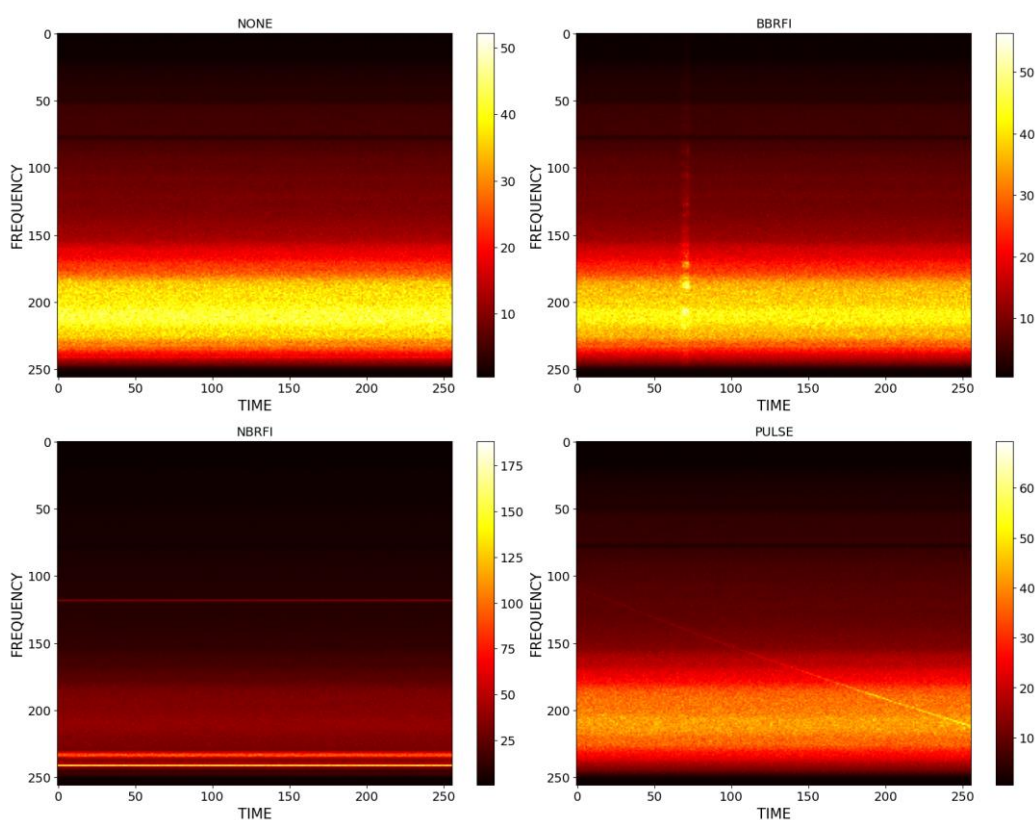


Figure 4 - Examples of four main types of data "time frames" in a pulsar-observation data set, visualised as 256x256 images. Vertical axis corresponds to frequency, horizontal to time, the value of each point is signal intensity. The top left frame is the most common one, "none" or "empty", it contains 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.

D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

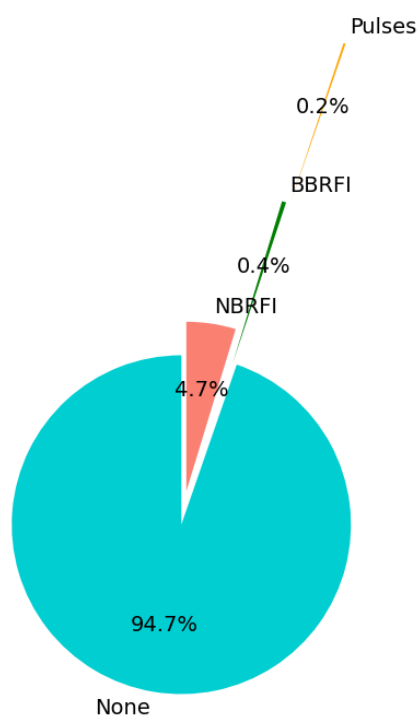


Figure 5- Distribution of the four main types of time frames in a typical pulsar data set. About 20 minutes of data observing one of the brightest and well-studied pulsars, the Crab pulsar, represented as more than 50,000 time frames, the size of the dataset is ~12.2 GB. There are only a few mixed-type or unidentifiable time frames, so their percentage is negligible and they are not included in this distribution

As shown in **Figure 5**, in a typical pulsar data set (in this case collected by the Effelsberg 100m radio telescope¹²) the "none" frames is by far the most common category, with only 5% of various types of RFI and a tiny 0.2% of signal frames. This illustrates the whole concept well: only 0.2% of the data is scientifically significant, everything else can be safely deleted right as it is being observed, with only some basic info (like time duration of empty periods) kept. So of 12 GB of data only 25 MB needs to be kept. It also shows that we cannot rely on the actual data for training the ML expert system, because the distribution of the time frame types in the real data is severely biased.

he developed ML-PPA framework is also being tested on longer data sets of different sources and produced not only by the Effelsberg, but also by one of the SKA pathfinders, MeerKAT¹³, which is the final testbed for this task and, as we hope, will ultimately use our software or its descendants in its actual day-to-day operation.

The work is split into three parallel and interacting subprojects (**Figure 6**) which are described in the following sections.

¹² Radio Telescope Effelsberg: <https://www.mpifr-bonn.mpg.de/en/effelsberg>

¹³ The data comes from ongoing scientific projects led by MPIfR radio astronomers.

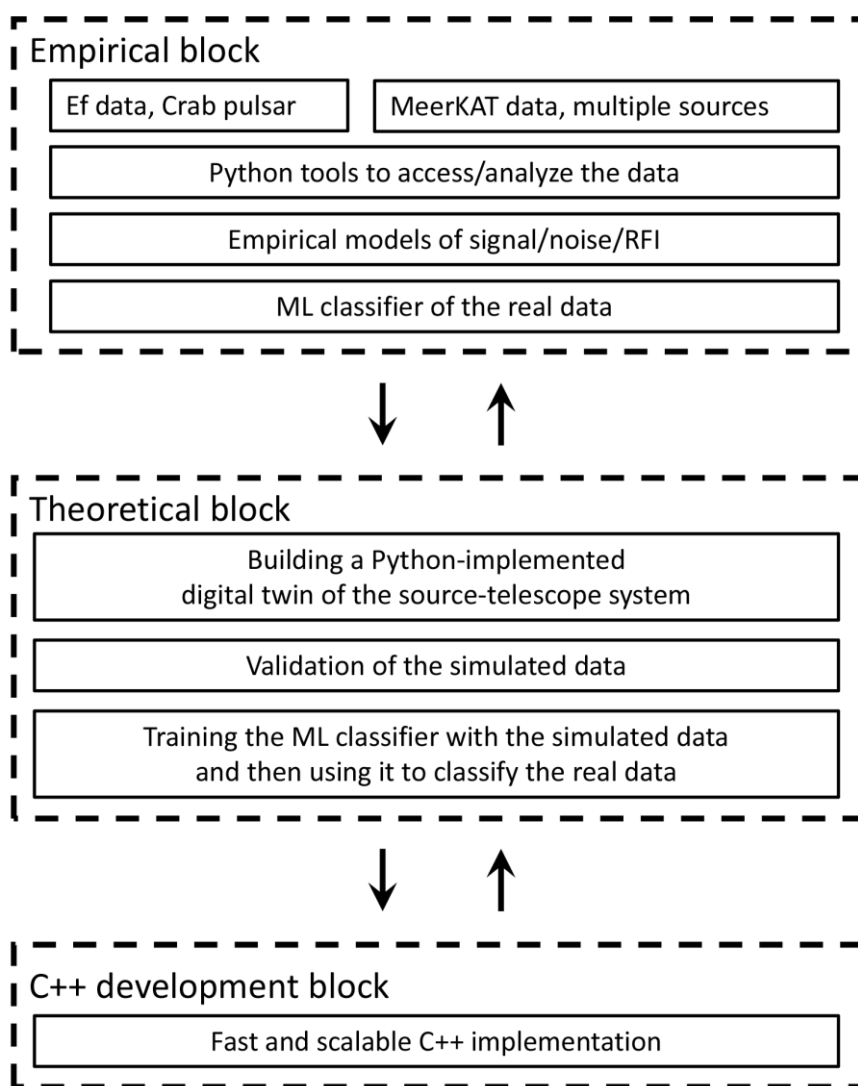


Figure 6 - Three parallel interacting subprojects of the task

2.2.1 Empirical block

This subproject focuses on the real data. The ML classifier (**PulsarRFI_NN**) is being developed to reliably label the dataflow: a convolutional neural network (CNN) based tool, implemented with TensorFlow in Python, that can label the data with 90% or better accuracy. Another aspect of this subproject is following the empirical approach in creating a DT (**PulsarRFI_Gen**) that generates different types of time frames by mimicking available real data (based on the geometry of images, noise characteristics etc.) rather than following the physical first principles as the theoretical block does. This alternative path allows to compare the results with the physical DT and to always have enough training data for the ML classifier during the development stages.

2.2.2 Theoretical block

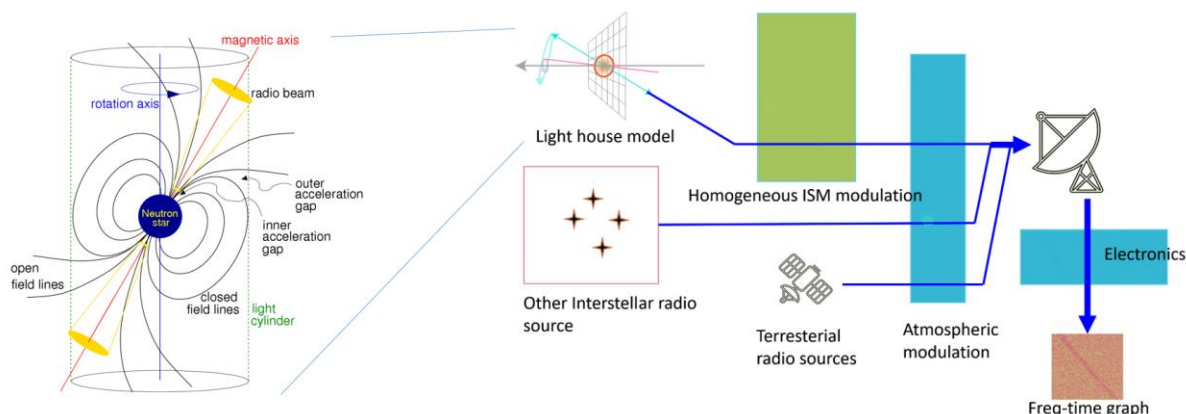


Figure 7 - General outline of the DT structure: modelling the astrophysical source (pulsar), transmission of the signal through the interstellar matter, receiving and processing by a radio telescope, adding sources of both natural and artificial interference and noise.

This is the main digital twin (**PulsarDT**) creation subproject. In it we build a sophisticated physical model of the pulsar signal, anything that interferes with it and the way it is finally recorded by a telescope. This model produces data that is as close as possible to that produced by a real telescope observing a real source, with a number of parameters that can be adjusted. The model (**Figure 7**) starts with the "lighthouse" representation of a pulsar and then adds to it propagation through interstellar matter (ISM) effects, other cosmic sources, influence of the Earth atmosphere, terrestrial sources, and effects of the telescope's receiving and recording equipment.

The purpose of the digital twin development is twofold. On one hand we would like to have an endless supply of training/testing material for the ML-based classifier with parameters that can be adjusted for a particular observation. On the other hand, building a digital twin of a telescope signal with known parameters can help with identification of the real signals in the telescope's data flow. For example, at the end we would like to be able to classify the scientifically significant part of the data not just as a "signal", but "a pulse of a pulsar with such and such physical parameters".

2.2.3 C++ development block

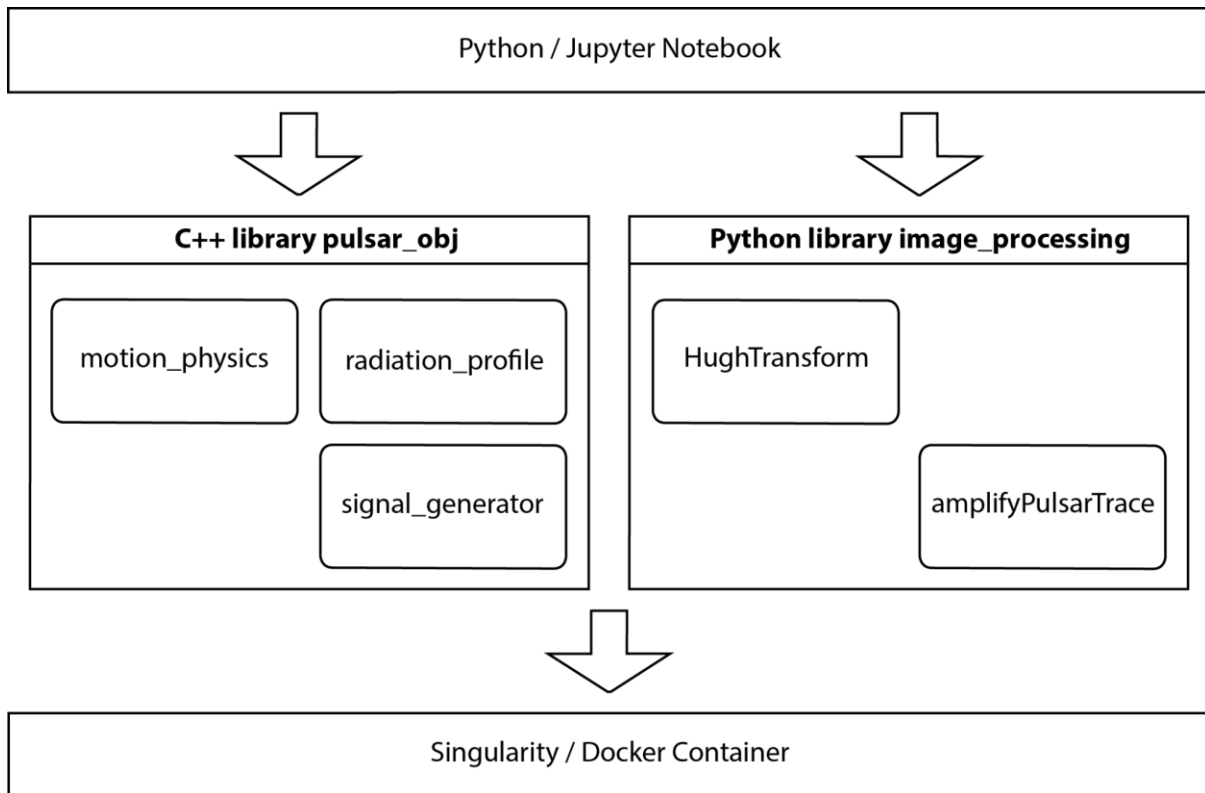


Figure 8 - Layered software architecture of the framework ML-PPA

D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

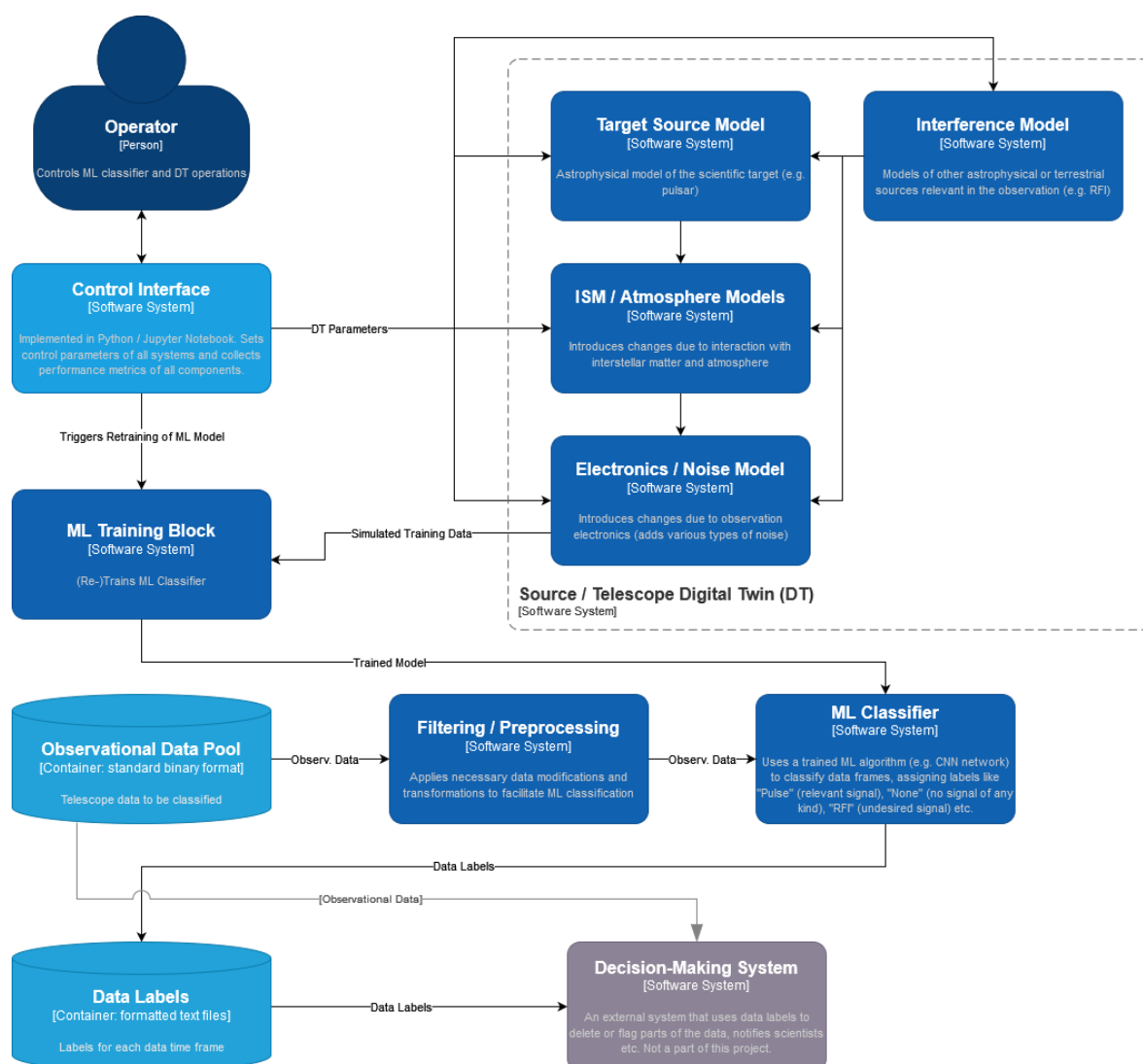


Figure 9 - Diagram of the final software product (in the C4 model)

The stable versions of the ML classifier and the source/telescope digital twin of the activities described [in sections 2.2.2](#) and [2.2.3](#) are then being rewritten in C++ (as **PulsarDT++**), with speed and scalability being a priority. The software ([Figure 8](#) and [Figure 9](#)) has a Jupyter notebook / Python based user interface. All of it included in a Singularity container for easy deployment to any platform.

The final version of the classifier is intended for use with the real data flow of the MeerKAT telescope, and, later, possibly with other telescopes as well. But these goals are already beyond the scope of the current project.



2.3 T7.3 GNN-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 in the 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 10 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).

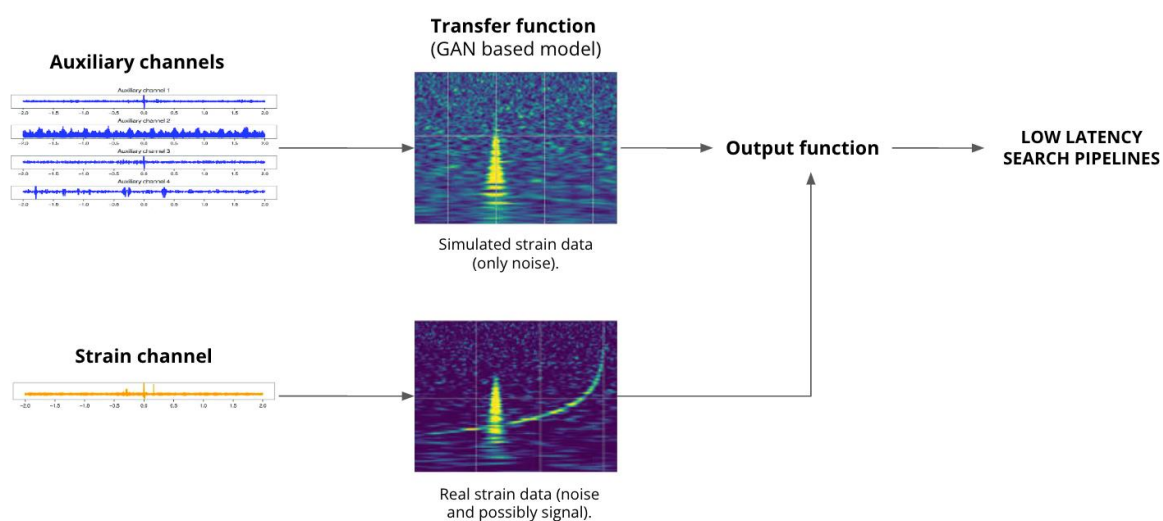


Figure 10 - High-level architecture of the DT

2.3.2 High-level architecture of the DT implementation

Figure 11 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.

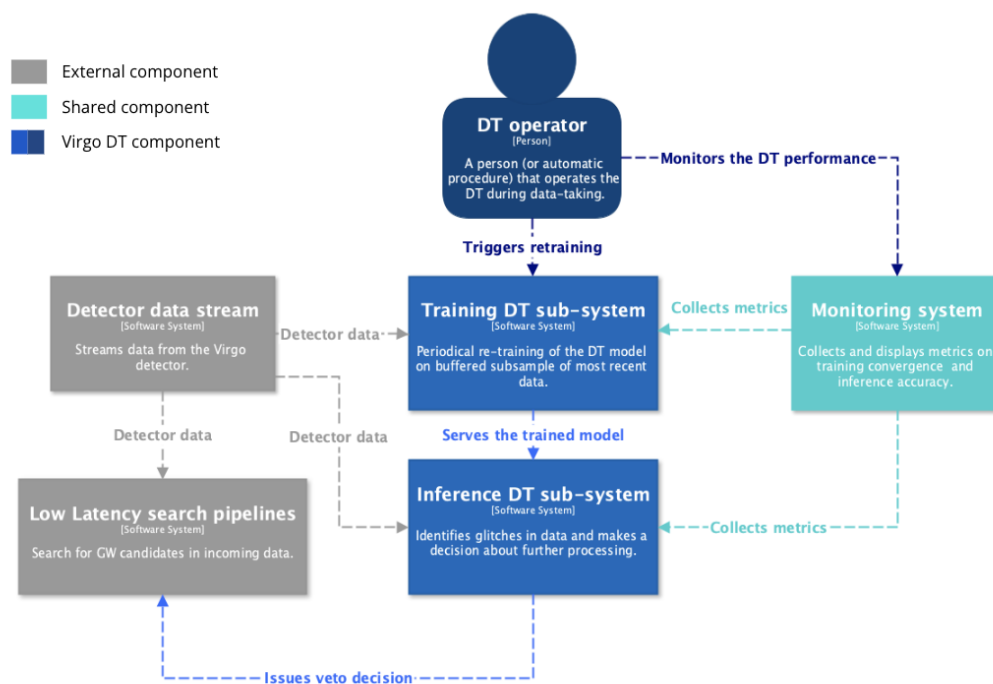


Figure 11 - System Context diagram (in the C4 model) of the DT for the veto pipeline.

Two main subsystems characterise the DT architecture: the *Training DT subsystem* and the *Inference DT subsystem*. The Training subsystem is responsible for the periodical re-training of the DT model on a buffered (on disk) subsample of most recent data. The Inference subsystem processes a stream of incoming data, identifies glitches, and issues a decision about further processing (veto).

The above subsystems are operated by a *DT Operator*, which could be a person or an automated procedure (in a later stage) that operates the DT during data-taking. The *DT Operator* monitors the operations of the DT by leveraging a *Monitoring System* that collects and displays metrics on training convergence and inference accuracy.

The monitoring and coordination of the sub-systems are achieved by implementing the modules as Kubernetes Pods orchestrated by Airflow¹⁴. This is achieved via several interconnected services designed for managing and processing the data, as shown in **Figure 12**.

¹⁴ <https://airflow.apache.org/>



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

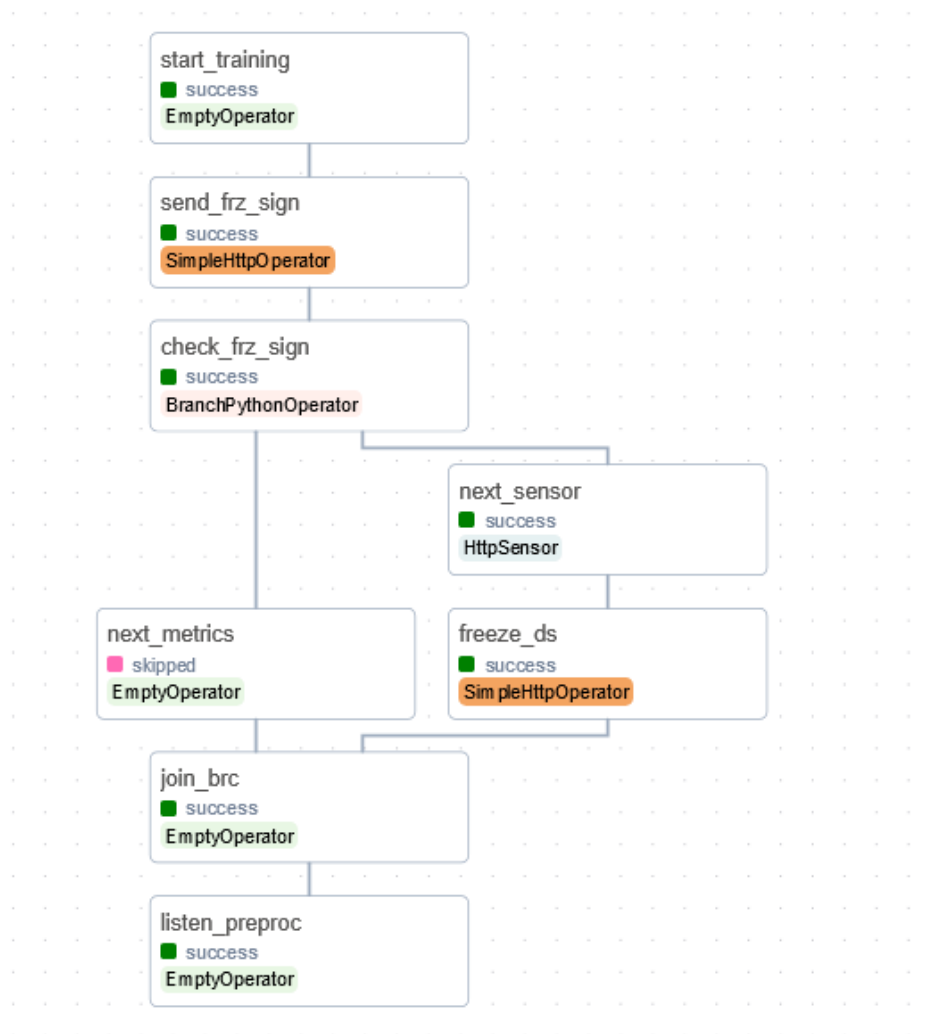


Figure 12 - Airflow DAG for training sub-system, as shown in the Airflow Dashboard

- **The Data Buffer**, implemented in Go¹⁵ using the Gin framework¹⁶, stores GW data and state information, utilising Go's concurrency for thread-safe operations.
- **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.

¹⁵ Go: <https://go.dev/>

¹⁶Gin: <https://gin-gonic.com/>

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

¹⁸Gunicorn: <https://gunicorn.org/>



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

- **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.

The entire system operates within a Kubernetes setup, with the Data Buffer as a stateful set and other services as deployments. Apache Airflow orchestrates these components, managing the workflow through tasks like freezing the datastore and monitoring buffer status, utilising various Airflow operators to enforce flow control and handle conditions.

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

2.3.3 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.

2.3.4 The Training DT subsystem

Figure 13 shows the Container diagram (in the C4 model) of the Training subsystem. The main modules of the subsystem are:

- **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 q-transform, 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.
- **GlitchFlow** is the module which contains the Generative Neural Network (GNN) for generating the glitches. In this first implementation of GlitchFlow, we rely on a ResNet12 architecture, due to it being simple and fast to train.
- **Preprocessing_API** is a module based on Python and proprietary IGWN libraries, which prepares the input data in a format suited for the Training application.

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



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

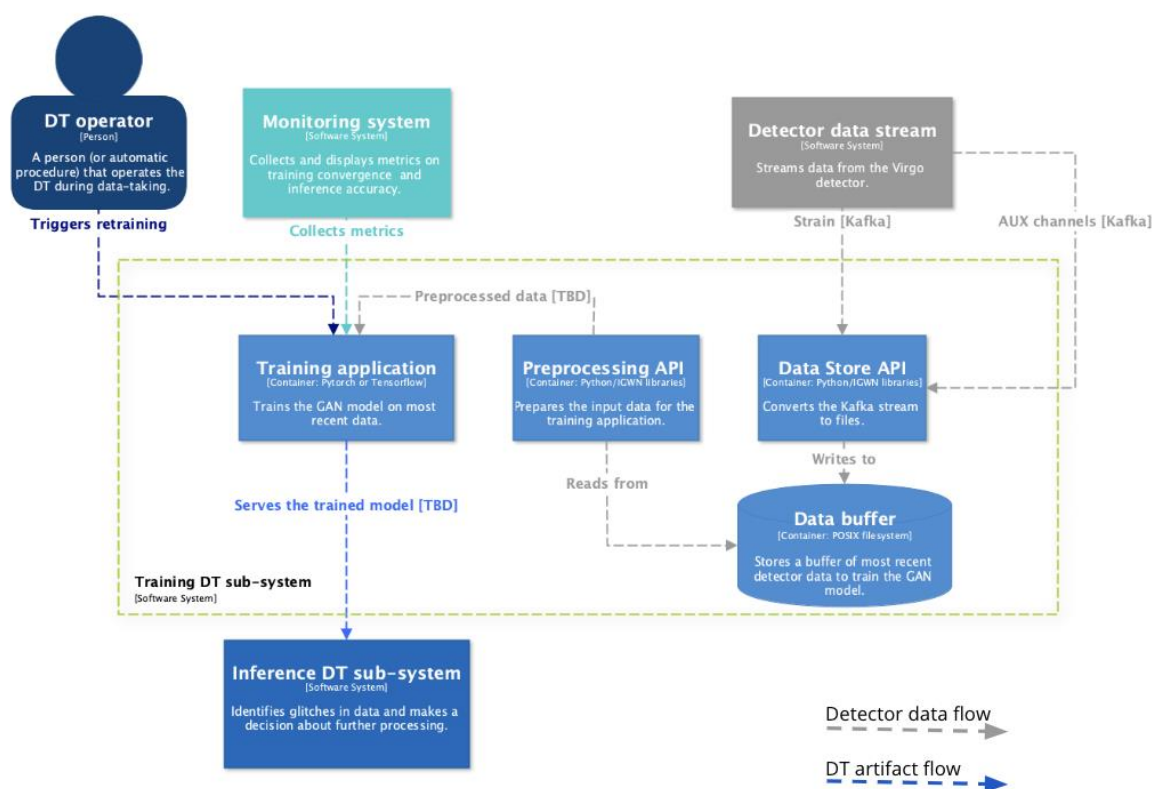


Figure 13 - Container diagram (in the C4 model) of the Training subsystem

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 only produced artefact is the trained model.

2.3.5 The Inference DT subsystem

Figure 14 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.

D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

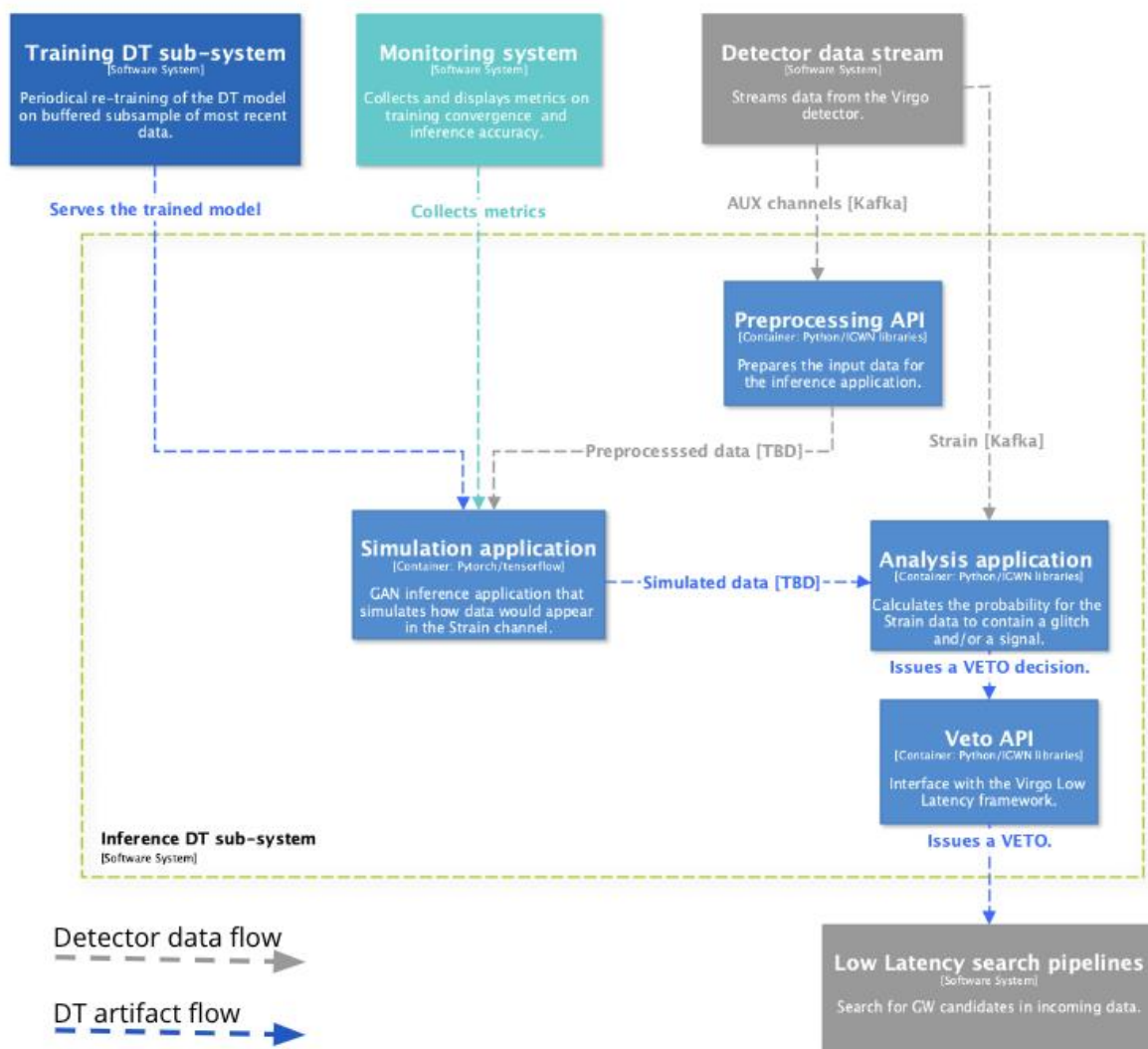


Figure 14 - Container diagram (in the C4 model) of the Inference subsystem

2.4 T7.7 Fast particle detector simulation with GAN

The two components cooperating in the thematic module of T7.7 are:

- the simulation component that incorporates the Monte Carlo-based (MC) simulation framework which produced data that consist of energy depositions in interactions with detector matter (R9)



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

- the deep learning (3D Generative Adversarial Network) component (**R2**), which, by leveraging the MC simulated data, trains deep learning models based on the chosen particle detector set up.

These models are integrated and can be run during the fast simulation process.

More specifically, the objectives of T7.7 are summarised below:

- Optimising the Generative Adversarial Network (GAN)-based model developed for a selected set of **detector geometries**.
- Integrating WP6 tools for distributed training and hyperparameter optimization.
- Implementing validation techniques capable of assessing different performance aspects, such as accuracy and comparison to classical Monte Carlo, uncertainty estimation, coverage of the support space. This activity will be contributing to the development of an agreed validation standard among the HEP community.

2.4.1 Use case overview

Particle detectors measure different particle properties at colliders such as the Large Hadron Collider (LHC). More specifically, the detectors called calorimeters are key components of the whole experimental setup, which are responsible for measuring the energy of the particles. In a collider, the 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 (**Figure 15**).

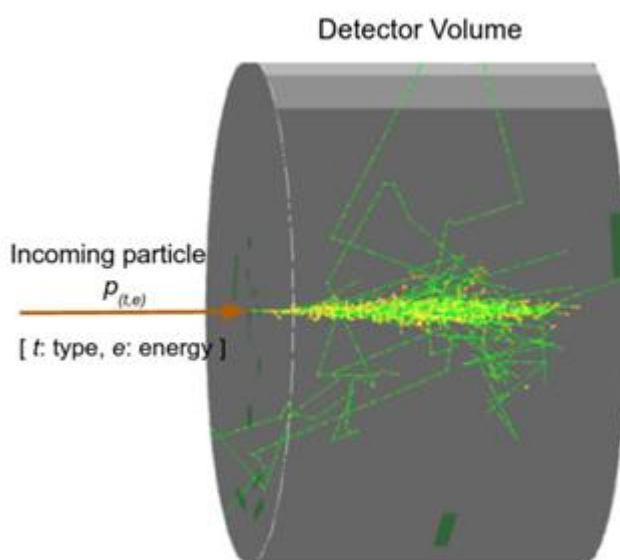


Figure 15 - Simulation of a particle shower created by the primary particle entering the detector volume, interacting with its material.

D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

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. Recent estimates show that the HEP community devotes more than 50% of the WLCG computing Grid²⁰ (which has a total of 1.4 million CPU-cores running 24/7/365 worldwide) to simulation-related tasks (R10). Moreover, Monte Carlo simulations are constrained by the need for accuracy, which will further increase, in the near future with the High Luminosity upgrade of the LHC (HL-LHC²¹). HL-LHC will increase the demand in terms of simulations, and consequently the need for computing resources, as well as the complexity of the associated detector data (R3).

Detector simulation allows scientists to design detectors and perform physics analyses. The simulation toolkit that has been developed to perform particle physics simulations based on MC methods, and is also used in our use case, is **Geant4**. It provides a highly flexible simulation framework in C++. Moreover, Geant4 is used by large scale experiments and projects from the domains of nuclear medicine, astrophysics, and HEP. It constitutes a set of components which include, geometry and tracking descriptions, detector response modelling, event management, user interfaces and much more.

Given the expected HL-LHC requirements in terms of simulation, the community has long since started developing faster alternatives to Monte Carlo, including generative approaches, such as deep learning based techniques (R4, R5, R6).

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 true, unknown distribution describing a finite number of observations. There are several variants of generative models found in literature with the most well-known ones being the Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs) and Autoregressive Models (R1, R7, R8). The thematic module under task 7.7 designed for CERN's use case concerns a fast particle detector simulation paradigm using GANs.

²⁰ WLCG computing Grid : <https://home.cern/science/computing/grid>

²¹ HL-LHC: <https://hilumilhc.web.cern.ch/>



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

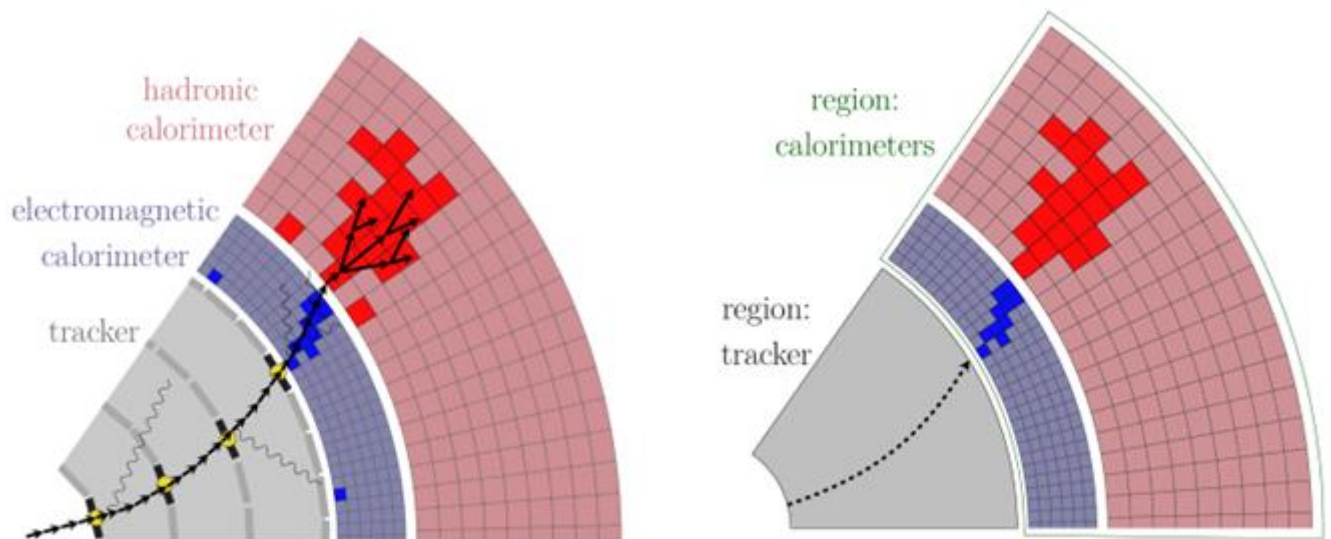


Figure 16 - Detailed (full) particle simulation with Geant4 (left) and fast particle simulation using generative ML techniques (right) (R3)

Compared to other generative approaches, the Generative Adversarial Network approach is able to demonstrate highly realistic and sharp images (R1). A GAN can learn a distribution implicitly as it doesn't rely on the explicit computation of probability densities. This use case leverages a 3D convolutional GAN, 3DGAN, as calorimeter detectors can be regarded as huge cameras taking pictures of particle interactions (R2). The voxels (3D calorimeter cells) are generated as monochromatic pixelated images with the pixel intensities representing the cell energy depositions.

D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

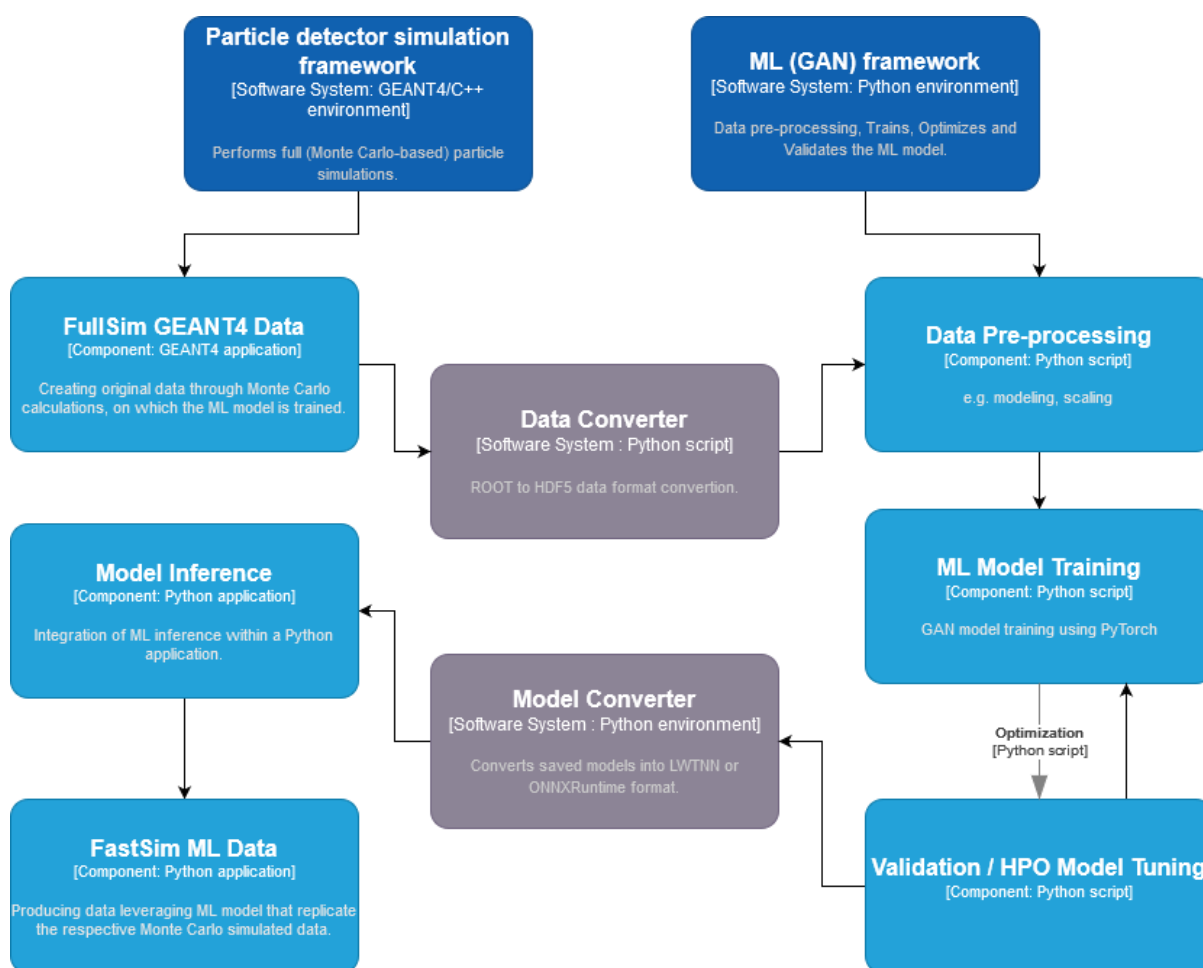


Figure 17 - Fast particle detector simulation using ML techniques high level workflow composition

2.4.2 3DGAN component implementation

An implementation of the 3DGAN approach has been developed and a more detailed description follows. 3DGAN (R2), has been implemented using Tensorflow v2²² and recently has been updated leveraging PyTorch²³. The code is available on GitHub^{24,25} and it has been tested and run locally on a single Linux node using 4 GPUs, but also on HPC nodes in distributed manner.

The 3DGAN architecture can be seen in **Figure 18**. 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 that it entered the detector, concatenated to the latent vector. The generator network then maps the input

²² <https://www.tensorflow.org/>

²³ <https://pytorch.org/>

²⁴ <https://github.com/CERN-IT-INNOVATION/3DGAN/tree/main/Accelerated3DGAN/src/Accelerated3DGAN>

²⁵ <https://github.com/interTwin-eu/DetectorSim-3DGAN>



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

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 LeakyRelu²⁶ 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.

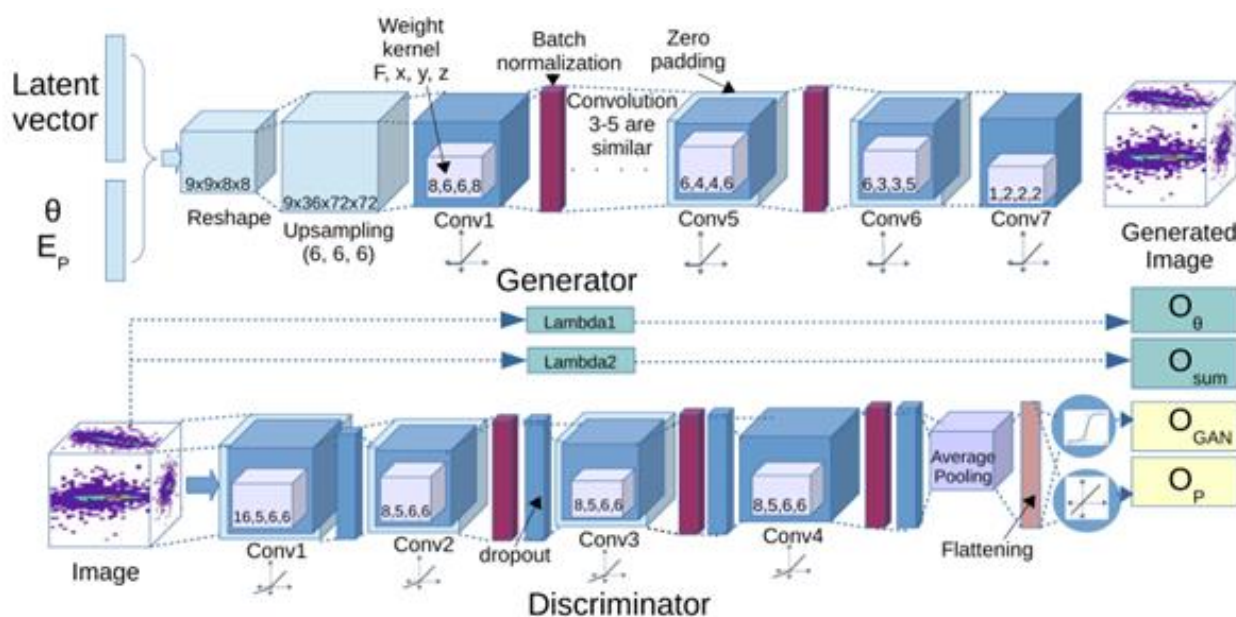


Figure 18 - 3DGAN model architecture

The training of the prototype model was inspired by the concept of transfer learning. Meaning 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. The first training step was run for 130 epochs and the second step was run for 30 epochs, both runs utilised GPU infrastructure.

²⁶ [https://en.wikipedia.org/wiki/Rectifier_\(neural_networks\)](https://en.wikipedia.org/wiki/Rectifier_(neural_networks))

D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

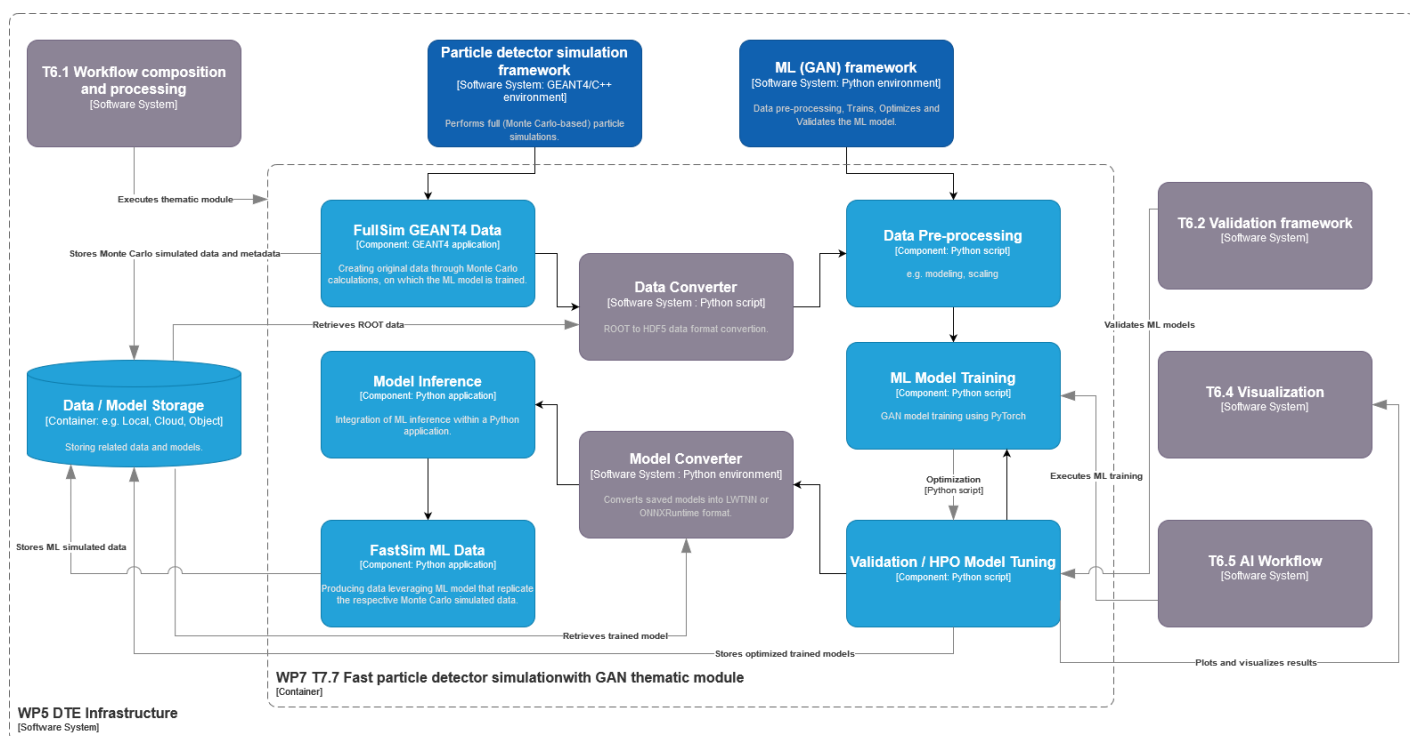


Figure 19 - Fast particle detector simulation using ML techniques high level workflow composition and its connections with other work packages' components in C4 format diagram

Figure 17 shows our digital twin application's workflow, as well as the different components it consists of. Moreover, in Figure 19 the relationships with other work packages and our thematic module are depicted. The ML workflow that includes the 3DGAN model training consists of several other modules, the data pre-processing module, the model definition and training module, the validation and hyperparameter optimization module.

- The preprocessing module 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 preprocessed data are then passed to the GAN model for training.
- The hyperparameter optimization (HPO) tuning module is used for searching for the best set of model hyperparameters (e.g., AutoML²⁷, Optuna²⁸ etc.).

²⁷ AutoML: <https://www.automl.org/automl/>

²⁸ Optuna: <https://optuna.org/>



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

- The validation module verifies the performance through a set of physics-motivated steps, both at single image quality level and at the sample level.
- Finally, the model is converted into ONNX²⁹ format and can be used to generate images that replicate the detector's response or for inference within a potential GEANT4 application (C++ based). The second option for the time being is not under the use case's scope.

Describing in more detail the above processes and the involved components will help us uncover the technical requirements concerning our digital twin particle detector application.

The dataset used for studying and developing the 3DGAN model (**R2**) (**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 ROOT³⁰ files, which need to be converted into a ML-friendly format HDF5 in order to train the model. The datasets and the converted data (ROOT to HDF5) need to reside in cloud file or object storage, as the volumes do not exceed several 100s of GB. The pre-processing of the data (cleaning, scaling, encoding etc.) is being performed in memory.

During pre-processing, simulation inputs are defined and encoded, i.e., the detector geometry, the energy and angle of the incoming particle. The 3DGAN training requires multiple GPU access and the best model weights need to be saved in a model registry repository, in order to be available during inference.

The performance of the model is evaluated in the validation module, which is currently being developed, through the creation of histograms describing particle shower observables. Shower observables are among others, total energy distribution (sum of all cell energy depositions), cell energy depositions distribution, longitudinal profile which represents the energy deposited by a shower as a function of the depth of the calorimeter and lateral profile which represents the energy density distribution as function of the radius of the calorimeter. Moreover, the physics-based validation process includes accuracy verification of those key distributions' first moments and precise evaluation of the tails of distributions that usually require larger amounts of samples. The GEANT4 and 3DGAN distributions are compared during this evaluation process.

Once the model is trained, tested, and validated, it is ready to be deployed in a Python application. The model architecture and the weights are also stored, respectively in the JSON HDF5 formats. These operations can be easily done in Python. Open Neural Network Exchange Runtime (ONNXRuntime) is a framework for neural networks inference. After training, the model can be saved in a format that can potentially be used for inference in a GEANT4 based application. The disk space required for the weights,

²⁹ ONNX: <https://onnx.ai/>

³⁰ ROOT: <https://root.cern/>



saved as HDF5 file, is about several hundreds of MBs and the model's architecture, saved as a JSON file, is hundreds of KBs.

3 Requirements for the thematic modules in the physics domain

Section 3 is dedicated to reporting the requirements concerning the thematic modules to be developed for the physics domain use cases. The four physics domain use cases include lattice QCD simulations (T7.1) and particle detector simulation in High Energy Physics (T7.7), as well as noise simulation in radio astronomy (T7.2) and the VIRGO noise detector in astrophysics (T7.3).

Thematic modules requirements consolidation resulted from the activities performed so far under work package 4 (WP4), which concerns the technical co-design and validation of the digital twin engine among research communities. Under this scope the objective was to introduce use-case specific requirements for the thematic modules, based on the DTE infrastructure (WP5) and core modules (WP6). Digital twin engine core modules described in work package 6 are responsible for capabilities concerning workflow compositions, real-time acquisition of data and processing, quality and uncertainty tracing, data fusion, big data analytics, as well as AI/ML workflow.

3.1 Input data requirements

The majority of the physics thematic modules require file or object-storage or they have already data stored in HPC centres. In particular, the T7.3 thematic module requires space on a POSIX-like filesystem. Same requirements apply for output data storage. Necessary pre and post processing techniques need to be applied for each one of the thematic modules, which are described in more detail in the respective subsections of **section 2**.

3.2 Expected data volume

The data volumes that physics domain thematic modules, in the content of the interTwin project are expecting, range from O(10) GB to at most O(100) TB.

3.3 DTE Infrastructure and Core components integration requirements

DTE Infrastructure and Core Modules designed and developed in work package 5 and 6 respectively, can be used to handle allocation of computing resources, composition of Thematic Modules into workflows, and data management.

3.3.1 DTE Infrastructure components

As documented in deliverable D5.2 a first version of the DTE infrastructure components have been released.

Most of the thematic modules in physics do not incorporate any real-time data acquisition procedure and therefore no related tools are required. Though, capabilities of online data processing are developed by T7.1 and T7.3, and especially T7.3 processes require streaming platforms, e.g., Apache Kafka. T7.2 ultimately aims at the development of a (quasi) real-time data classification tool, but this goal is for the future and is not a part of the current task, which only relies on pre-acquired data.

Different data formats are produced together with different pre-processing and post-processing requirements. Therefore, data and metadata formats concerning T7.7 are ROOT and HDF5, binary and textual data for T7.1, time series data in binary form with text header for T7.2 and gwf files for T7.3.

To support all the related processes from modelling and ML training to inference and validation, Linux based operating systems, containerized environments (i.e., Docker, Singularity) are required at the current stage of the thematic modules.

3.3.2 DTE Core components

As documented in deliverable D6.2 a first version of the DTE core components have been released. In particular the components from WP7 are integrated with the AI framework (itwinai). Most of the physics use cases use Python to perform ML related processes. Therefore, Python ML frameworks, like Tensorflow and PyTorch are being utilised for the development of the individual, use case specific machine learning frameworks. Tensorflow is the one that is used by the majority of the thematic modules. Statistics and neural network-based ML models are used, as well as monitoring tools, such as Tensorboard. Moreover, ML validation frameworks will be implemented for model validation and quality check.

Support for metadata management tools for tracking data lineage and ensuring data quality. Ability to share data across different teams and organisations. Tools such as GitHub and Zenodo are examples of the above.



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

The applications' results are going to be presented using open-source visualisation libraries and dashboards, where plots analysing the produced outcome will be displayed.

Requirements in terms of software stacks to be supported in Core modules include:

- Tensorflow, PyTorch
- JupyterLab
- Conda
- Docker, Singularity
- ML platform: Kubeflow
- Streaming platforms: Apache Kafka
- Big data analytics tools: Apache Spark
- ML monitoring tools: Tensorboard, Prometheus, Grafana

3.3.3 Computing requirements

Concerning computing resources, physics thematic modules require systems with multiple GPU support, HPC centres computing power with MPI infrastructure. As well as support for distributed computing, availability of GPUs for machine learning workloads, and ability to scale up or down computing resources based on demand.



4 Conclusions

The Digital Twin Engine's physics thematic modules concerning tasks T7.1, T7.2, T7.3 and T7.7 have been designed and significant parts of them have been developed throughout the first 24 months of the interTwin project. Additionally, the scientists involved in the analysis activities of WP7 and physics domain tasks have identified the technical requirements that are important for the development of the specific applications; they have been working together with their collaborators involved in WP5 and WP6 to integrate the core modules and underlying infrastructure in their work, leveraging the computing resources that were put at their disposal by the collaboration.

In this deliverable the applications of the four thematic modules are presented, together with a detailed discussion of their main components. The applications concern lattice QCD simulations and data management, noise simulation for radio astronomy, GAN-based thematic modules that manage noise simulation, low-latency de-noising and veto generation for gravitational waves, as well as fast particle detector simulation with GAN.

The presentation of the developed modules and the evolution of their requirements detailed in the document resulted in the following conclusions. Each of the modules presented aims at tackling relevant issues faced by the different communities they refer to; in most cases they do this by employing state-of-the-art deep learning techniques which are starting to show their full potential in a vast number of areas. These include, among others, generative adversarial networks, normalising flows and variational autoencoders. The projects exploit the potential of machine learning both for classification tasks and for generation, often sharing common architecture of the neural networks and the logic of the different high level components. There is a significant variation in the size of the data needed for the training of the neural networks in the different modules, going from tens of Gigabytes to hundreds of Terabytes. All the projects, however, share similar requirements for the formats of the input data. Finally, we observe that the nature of the majority of the use cases require modern frameworks, such as Python and ML based, to be combined with frameworks that require low level programming languages such as C/C++.



5 References

Reference	
No	Description / Link
R1	Generative Adversarial Networks. Ian J. Goodfellow et al. https://arxiv.org/abs/1406.2661 DOI: https://doi.org/10.48550/arXiv.1406.2661
R2	Fast Simulation of a High Granularity Calorimeter by Generative Adversarial Networks. Gul Rukh Khattak et al. https://arxiv.org/abs/2109.07388 DOI: https://doi.org/10.48550/arXiv.2109.07388
R3	https://g4fastsim.web.cern.ch/
R4	Fast Simulation for ATLAS: Atfast-II and ISF. W. Lukas. Technical Report ATL-SOFT-PROC-2012-065, CERN, Geneva, Jun (2012) DOI: https://doi.org/10.1088/1742-6596/396/2/022031
R5	Fast simulation of the CMS detector. 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: https://doi.org/10.1088/1742-6596/219/3/032053
R6	Fast simulation of electromagnetic showers in the ATLAS calorimeter: Frozen showers. E. Barberio et al. J. Phys. Conf. Ser., 160, 012082, (2009). DOI: https://doi.org/10.1088/1742-6596/160/1/012082
R7	Auto-Encoding Variational Bayes. D. P. Kingma et al. DOI: https://doi.org/10.48550/arXiv.1312.6114
R8	Pixel Recurrent Neural Networks. Aaron van den Oord et al. DOI: https://doi.org/10.48550/arXiv.1601.06759
R9	GEANT4: https://geant4.web.cern.ch/
R10	HEP Software Foundation Community White Paper Working Group - Detector Simulation. HEP Software Foundation. DOI: https://doi.org/10.48550/arXiv.1803.04165
R11	openQ*D code: a versatile tool for QCD+QED simulations. I. Campos, P. Fritzscht, M. Hansen, M. K. Marinkovic, A. Patella, A. Ramos & N. Tantalot



D7.6 Updated report on requirements and thematic modules functionalities for the physics domain

	<p>The European Physical Journal C, 80, Article number: 195 (2020) DOI: https://doi.org/10.1140/epjc/s10052-020-7617-3 volum</p>
R12	<p>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: https://doi.org/10.1109/TPAMI.2020.2992934</p>
R13	<p>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 – Published 22 August 2011 https://link.aps.org/doi/10.1103/PhysRevD.100.034515</p>
R14	<p>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 https://link.aps.org/doi/10.1103/PhysRevD.104.094507</p>
R15	<p>Introduction to Normalizing Flows for Lattice Field Theory Michael S. Albergo, Denis Boyda, Daniel C. Hackett, Gurtej Kanwar, Kyle Cranmer, Sébastien Racanière, Danilo Jimenez Rezende, Phiala E. Shanahan https://doi.org/10.48550/arXiv.2101.0817</p>
R16	<p>InterTwin D7.2 Report on requirements and thematic modules definition for the physics domain (Submitted to EC). Kalliopi Tsolaki <i>et al.</i> (2023). DOI: https://doi.org/10.5281/zenodo.10417161</p>
R17	<p>InterTwin D7.4 First version of the thematic modules for the physics domain (Submitted to EC). Gaurav Sinha Ray <i>et al.</i> (2023). DOI: https://doi.org/10.5281/zenodo.10224277</p>

