D7.4 First version of the thematic modules for the physics domain

Status: FINAL
Dissemination Level: public

Funded by the European Union

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 current status of the development of the thematic modules in the physics domain.

Document Description			
D7.4 First version of the thematic module for the physics domain			
Work Package number 7			
Document type	Deliverable		
Document status	FINAL	**Version**	1
Dissemination Level	Public		
Copyright Status	This material by Parties of the interTwin Consortium is licensed under a Creative Commons Attribution 4.0 International License.		
Lead Partner	CSIC		
Document link	https://documents.egi.eu/document/3958		
DOI	https://doi.org/10.5281/zenodo.10224277		
Author(s)	• Gaurav Sinha Ray (CSIC) • Isabel Campos (CSIC) • Yurii Pidopryhora (MPG) • Sara Vallero (INFN) • Alberto Gennai (INFN) • Sofia Vallecorsa (CERN) • Kalliopi Tsolaki (CERN)		
Reviewers	• Liam Atherton (STFC) • Michał Orzechowski (Cyfronet)		
Moderated by:	• Gaurav Sinha Ray (CSIC) • Sjomara Specht (EGI)		
Approved by	AMB		

Revision History

Version	Date	Description	Contributors
V0.1	14/09/2023	ToC	Andrea Manzi (EGI), Gaurav Sinha Ray (CSIC)
V0.2	01/11/2023	Contributions from all partners	Gaurav Sinha Ray (CSIC), Isabel Campos (CSIC), Yurii Pidopryhora (MPG), Sara Vallero (INFN), Alberto Gennai (INFN), Sofia Vallecorsa (CERN), Kalliopi Tsolaki (CERN)
V0.3	09/11/2023	Introduction, executive summary, integration status, and conclusions added. Draft sent to reviewers.	Gaurav Sinha Ray (CSIC), Isabel Campos (CSIC), Yurii Pidopryhora (MPG), Sara Vallero (INFN), Alberto Gennai (INFN), Sofia Vallecorsa (CERN), Kalliopi Tsolaki (CERN)
V0.4	27/11/2023	Incorporated feedback from reviewers.	Gaurav Sinha Ray (CSIC), Isabel Campos (CSIC), Yurii Pidopryhora (MPG), Sara Vallero (INFN), Alberto Gennai (INFN), Sofia Vallecorsa (CERN), Kalliopi Tsolaki (CERN), Javad Komijani (ETHZ)
V0.5	28/11/2023	Send for Q phase	
V1.0	29/11/2023	**Final**	

Terminology / Acronyms

Term/Acronym	Definition
GAN	Generative Adversarial Network
PoC	Proof of Concept
DTE	Digital Twin Engine
ML	Machine Learning

Terminology / Acronyms: **https://confluence.egi.eu/display/EGIG**

Table of Contents

Table of Figures

Executive summary

This deliverable, D7.4, is a report on the status of the development of the thematic modules 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, radio astronomy, and gravitational wave astronomy. The modules will be integrated into the Digital Twin Engine (DTE) that forms an important part of the interTwin project. Each module's functionality is described and contextualised by reference to the relevant DT and its use cases. Included is a technical summary of each software module that includes basic information such as its software licence and release notes.

1 Introduction

1.1 Scope

Previous reports identified the technical requirements that are important for the development of the thematic modules in WP7 [R1]. This deliverable is an update that summarises the development status 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 describes the software components included in each thematic module and their purpose and role in the context of their corresponding use case. **Section 3** summarises the basic specifications for each software component listed in **Section 2** and, where possible, includes links to more detailed technical and user documentation. Also included in this section for each component are their release notes and a short description of their future plans. **Section 4** outlines the progress of the integration of the thematic modules with the other Work Packages in interTwin.

2 Thematic Modules Architecture for the Physics Domain

2.1 T7.1 LQCD simulations and data management

The purpose of T7.1 is to develop software to carry out Lattice QCD simulations and integrate it with software modules and data infrastructures being developed by other groups under the interTwin banner. Two software modules fall under this task,

- **normflow**
- **openQxD.**

The openQxD[1] simulation software is a C code that can be used to simulate QCD+QED (or QCD) quantum field theories [R2]. The code allows one to choose among a wide variety of temporal and spatial boundary conditions when configuring particle simulations.

As described in D7.2[2] the more efficient analysis of lattice data would be facilitated by the adoption of a data sharing model following the data lake architecture being developed by WP5.2 [R1]. A testbed built along these lines is currently being operated at DESY and sample lattice data has been provided as part of the testing process. This will eventually involve the automatic copying of data and the integration of the FAIR principles into lattice data storage and retention policies.

Improvements to the software development workflow are being made by the integration of the SQAaaS module being developed by WP6.2. Software quality assurance in this context involves making sure releases are tagged properly etc. The project is providing feedback to WP6.2 to expand and improve its utility. This is intended to be a springboard to the integration into the OpenQxD development framework of a CI/CD pipeline with unit tests.

The software module normflow[3], a ML inspired lattice DT also described in D7.2, is being developed for use case T4.1. The main idea behind the method of normalising flows is to build and train a neural network for mapping a theory of interest to another one that is easier to simulate, ideally to a theory in which the degrees of freedom are decoupled. Once such a map is found, one can efficiently draw many samples from the theory of interest. Work is currently ongoing on the construction of gauge-equivariant transformations and development of a model for SU(2) and SU(3) gauge theories in four spacetime dimensions.

[1] https://gitlab.com/rcstar/openQxD
[2] https://zenodo.org/record/8036997
[3] https://github.com/jkomijani/normflow

2.2 T7.2 Noise simulation for radio astronomy

Task 7.2 aims to develop a framework for extracting pulsar signals from radio-astronomical observatory data streams. It is a ML-based data-labelling system that reads the data flow coming from a radio 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. This includes both scientifically valuable data, simulating the path of pulsar signals from source to measurement by telescopes, and various interference and noise signals. In particular, types of the Radio Frequency Interference (RFI) signals, i.e. background noise emanating from devices such as cell phones or satellites, are modelled. The resulting DT-generated data is to be used to train the ML data-classification tool. The DT is physics-based: a set of the control parameters will allow adjustment of the output to different sources, detection instruments, and observing conditions.

Although ultimately there will be two software modules (the labelling system and the physics-based DT), currently four modules are being developed, all under the umbrella designation of ML-PPA (Machine Learning-based Pipeline for Pulsar Analysis):

- PulsarDT
- PulsarDT++
- PulsarRFI_Gen
- PulsarRFI_NN

PulsarDT: physics-based DT, simulation of the propagation of pulsar signals from the source to antennas and generation of synthetic data – written in Python, to test algorithmic strategies for physical models of pulsars, interstellar medium, telescopes, interference, and noise, that will be later implemented in PulsarDT++.

PulsarDT++: At the moment this is PulsarDT implemented in C++ plus a general test of architecture of the whole ML-PPA package. This is what is going to be the final production release (it will also include a C++ version of PulsarRFI_NN).

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. The purpose of this tool is to have enough training data for the ML classifier while the much more complicated physics-based DT is still being developed.

PulsarRFI_NN: the ML classifier. It is a CNN-based tool for identification of various types of pulsar and RFI signals in the "timeframes", 2D images (time-frequency). Just like PulsarDT it is also a Python-based testing ground for various algorithms that later will contribute to PulsarDT++.

A detailed overview of the current state of the project can be found in this document: **https://gitlab.com/ml-ppa/gitlab-profile/-/blob/main/PUNCH_interTwin_project.pdf**

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

Aim of this task is to develop the thematic module for the simulation of transient noise in the Virgo Interferometer using Generative Adversarial Networks (GANs) [R5]. Using interTwin software modules and infrastructures we plan to implement two subsystems: the Training DT (Digital Twin) subsystem and the Inference DT subsystem. These 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 checking relevant metrics on training convergence and inference accuracy.

The Training DT subsystem is responsible for the periodical re-training of the DT model on a buffered subsample of 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.

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 to use 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 design of the Training DT subsystem comprises the following modules:

- Data Store API. It allows the DT operator or other modules to interact with the Data Store. Upon receiving the “Training” event trigger, it sends a “Freeze” event to the Data Store, freezing the current data in the buffer, i.e. no new data will be written in the Data Store, and no data will be deleted. When the Data Store enters the “Frozen” condition, the Data Store API sends a “Frozen” event to the Pre-processing module.
- Pre-Processing module. The pre-processing module spawns upon reception of the “Frozen” event from the Data Store API. The pre-processing module reads data from the Data Store and builds time series of the appropriate length for training (using crop or join methods). It applies standard-preprocessing (de-noising) and q-transform to produce time-frequency spectrograms, and then turns them into 2D images. The preprocessing functions are heavily based on the use of the Python package GWpy.
- Training module. Python-based module that trains the GAN model and stores the model in the Model Catalog.

The Inference DT subsystem works in a similar way as the Training DT subsystem, but it processes a stream of incoming data, rather than using data from the buffer. The incoming data will contain time-series of the Virgo strain channel and the relevant auxiliary channels. The data will be pre-processed by the Pre-Processing module in order to produce images in the same format as those used by the training module. Inference is then applied using the latest GAN model retrieved from the Model Catalog. The aim of this component is to identify glitches and issue a decision about further processing (veto or denoising).

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) [R5]. This thematic module consists of two components: the simulation component that incorporates the Monte Carlo based simulation framework, Geant4[4] and the deep learning (3D Generative Adversarial Network -3DGAN [R4]) component, which will produce deep learning models based on a specified particle detector set up. These models are integrated and can be run during the simulation step.

The two components will be linked together to support 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:

- use the Geant4 application to simulate particles passing through a specific detector setup (full/Monte Carlo-based simulation)
- pre-process the simulated data
- 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 within the Geant4 application during the inference step (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[5]). It describes the key steps, from particle simulations to event generation, and subsequent

[4] GEANT4: https://geant4.web.cern.ch/
[5] https://zenodo.org/record/8321134

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 innovative simulation application.

This application consists of two components, the component that incorporates the Geant4-based simulation framework and the deep learning component, which uses deep generative models based on 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 1. Below, the application functionalities and their specifications included in each workflow are described.

The Geant4 simulation toolkit, which consists of an important component of CERN's application, performs particle physics simulations based on Monte Carlo (MC) methods. The training workflow design includes the following functionalities, which will run on HPC systems managed by Kubeflow containerized components. 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 its trajectory angle with respect to the detector volume, and other metadata. The produced data, in ROOT format, will be stored at different data centres, with CERN currently serving as the primary storage site. The Geant4 application will run on HPC systems in a containerized environment.

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 data centres. Following the ROOT to HDF5 format conversion, the HDF5 data is further pre-processed and transformed into numpy arrays, a process currently incorporated within the model training scripts.

A GAN is trained [R4] 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 currently being explored.

At the end, the training workflow stores the optimised models, selected based on validation results, and converts them into the ONNX format for use during inference. Currently, the transformation of the model architecture and weights is performed within a Python script. The model registry where the GAN models will be stored is managed by Task 6.5.

During inference workflow processes, the Geant4 application initiates a particle, guiding it through the detector until it reaches the bottleneck detector part (the calorimeter), at which the GAN model performs inference. This functionality will be incorporated within the Geant4 application, which of course requires the retrieval of the stored ONNX formatted models. The model's output undergoes a detector-specific transformation to convert it into a Geant4 suitable input: the 3D images that the model generates are mapped into the so-called "hits" data consisting of the position (x, y, z coordinates) in the detector (i.e. the sensors positions) and the corresponding energy measurements.

The transformed data is used by the Geant4 framework to complete the process of generating events, simulating the passage of particles through the remaining components of the detector. 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 workflows allow for greater flexibility and adaptability in tuning the detector's responses to various particle interactions.

2.5

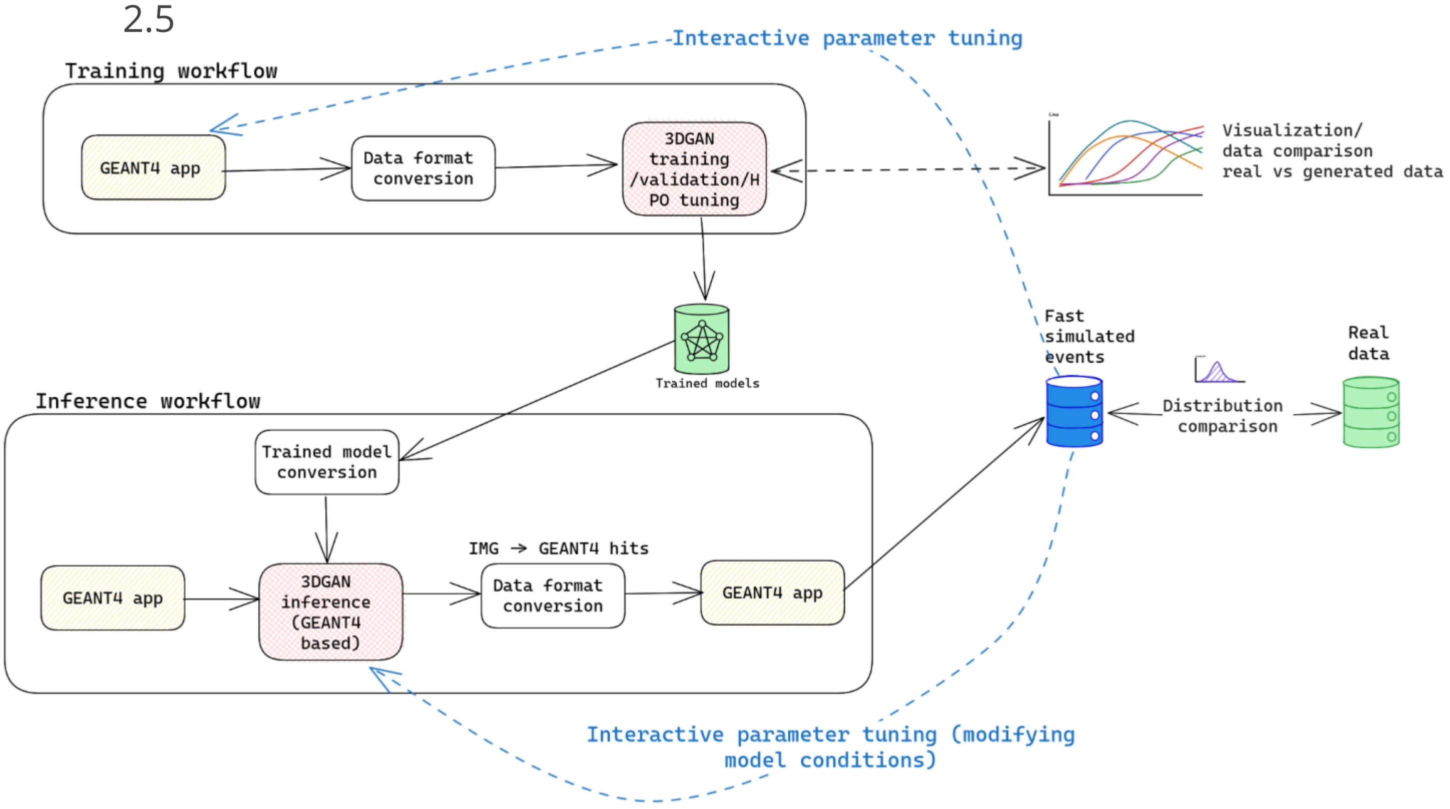

Figure 1 – Detailed graph representation of the training and inference workflows composition (as described above) of the fast particle detector simulation DT utilising 3DGAN approach

3 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

Component name	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 repository[6].

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 normflow

Component name	normflow

[6] https://gitlab.com/rcstar/openQxD/-/blob/master/CHANGELOG

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_**

3.2.1 Release notes

This package contains utilities for the implementation of the method of normalising flows as a generative model for lattice field theory. The method of normalising flows is a powerful approach in generative modelling that aims to learn complex probability distributions by transforming samples from a simple distribution through a series of invertible transformations. It has found applications in various domains, including generative image modelling. Normflow currently supports scalar theories in any dimension, and we are extending the package to accommodate gauge theories.

3.2.2 Future plans

We plan to extend our work to the more physically relevant SU(2) and SU(3) gauge theories in four spacetime dimensions. We need to construct and train a neural network to map the link variables of a gauge theory to a theory with decoupled link variables. However, a naive transformation of the link variables may lead to a distribution that is not symmetric under gauge transformation. To avoid this problem one should construct a neural network that respects the gauge symmetry. We are currently working on the construction of neural networks that encode these gauge-equivariant transformations.

3.3 T7.2 PulsarDT

Component name	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**
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

This is the first internal release. Assorted related materials, including Jupyter notebooks with use examples, are available **at GitLab** or will be added soon. 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 **the paper**. 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 like this, including the timeframe 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:

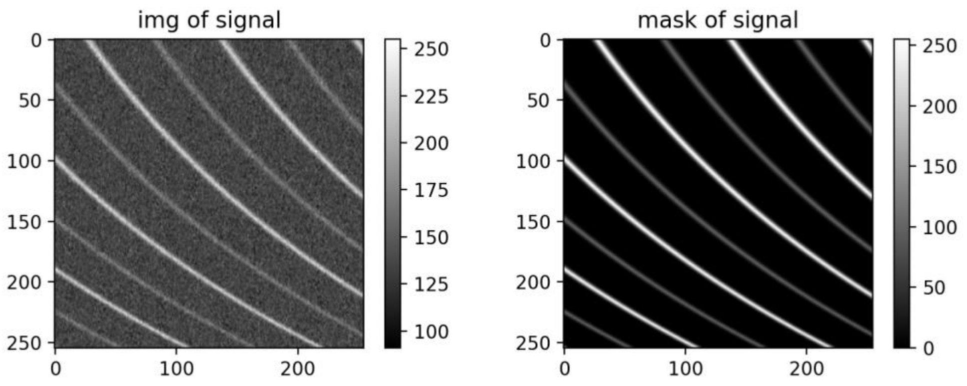

Figure 2 - example of an image and mask pair

3.3.2 Future plans

The plans are to constantly build upon the current model. Ultimately the goal of this component is to supply its counterpart, PusarDT++, with algorithms and implementation strategies, so it is used as a testing ground and is constantly modified. In the near future the interstellar matter (ISM) and electronics noise models are to be replaced by more sophisticated ones. The telescope model, which is now assumed to be a single antenna setup, will be expanded to include antenna arrays. Further, more complexity will be added to the source, e. g. pulsars in double systems will be considered.

3.4 T7.2 PulsarDT++

Component name	PulsarDT++
Description	Physics-based DT, simulation of the propagation of pulsar signals from the source to antennas and generation of synthetic data – written in C++.
Value proposition	PulsarDT implemented in C++. This is going to be the production version of the whole ML-PPA package and will also include a C++ version of the ML-classifier (PulsarRFI_NN).
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

This is the first internal release. Assorted related materials, including Jupyter notebooks with use examples, are available **at GitLab** or will be added soon. 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 **the paper**. This component is an efficient parallel-computing capable implementation of the whole ML-PPA, including the ML-classifier and DT, a layered architecture with a Python-based user interface on the top of various modules containerized using Singularity. This version includes a C++ implementation of PulsarDT as its main module.

3.4.2 Future plans

Ultimately this component will contain the whole ML-PPA package, both the physics-based DT (PulsarDT) and the ML-classifier (PulsarRFI_NN). Including stable versions of both plus a user interface is the main development goal at the moment, and then updating these two tools as the Python prototypes are updated. Another goal is to work on the efficient use of parallel computing.

3.5 T7.2 PulsarRFI_Gen

Component name	PulsarRFI_Gen
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	The purpose of this tool is to have enough training data for the ML classifier while the much more complicated physics-based DT is still being developed.
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

This is the first internal release. Assorted related materials, including Jupyter notebooks with use examples, are available **at GitLab** or will be added soon. 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 **the paper**. 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 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 it should, with little space for improvement and additional functionality. It will be updated in the future, but there will probably be no drastic changes. Originally it was planned that only the physics-based DT (PulsarDT) would be added to the final production version of the ML-PPA package (PulsarDT++), but since PulsarRFI_Gen has already shown itself quite useful, it may also be added to PulsarDT++ as an alternative tool for production of training data for the ML-classifier.

3.6 T7.2 PulsarRFI_NN

Component name	PulsarRFI_NN
Description	The ML classifier. It is a CNN-based tool for identification of various types of pulsar and RFI signals in the "timeframes", 2D images (time-frequency).
Value proposition	The main tool of the framework.
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

This is the first internal release. Assorted related materials, including Jupyter notebooks with use examples, are available **at GitLab** or will be added soon. 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 **the paper**. The main functionality of this component is assigning labels to 2D time-frequency "timeframes". Each timeframe can be classified into one of the 4 main categories: pulse, two kinds of interference, 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). Users are also supplied with Jupyter notebooks that illustrate the use of this component.

3.6.2 Future plans

This tool accomplishes its primary purpose. However, its performance when dealing with data with a low signal-to-noise ratio still needs to be improved, and that is going to be the focus of the development efforts in the near future. In particular, the ML approach is being changed (distributed training: HeAT, Horovod) and noise is to be handled more efficiently. And in a more distant future the plans are for it to be able to classify much more than just 4 basic categories of data, ideally to detect physical properties of the signal and interference components.

3.7 T7.3 GWpy

Component name	GWpy
Description	A python package for gravitational-wave astrophysics
Value proposition	Standard functions for preprocessing of time series for GW data
Users of the Component	Expert users and developers
User Documentation	**https://gwpy.github.io/docs**
Technical Documentation	**https://gwpy.github.io/**

Responsible	Ligo and Virgo Collaborations
Licence	GPL-3.0-or-later
Source code	**https://github.com/gwpy/**

3.7.1 Release notes

GWpy is a stable and actively maintained package developed by the Ligo-Virgo-Kagra collaboration. It is a key dependency of the modules to be developed in the context of WP 7.3. WP specific packages have not been released yet.

3.7.2 Future plans

There is no released WP module and the functionalities developed so far, a proof of concept of the Preprocessing API and of the Training and Inference subsystems, are implemented in Jupyter notebooks[7]. The next step is to release a set of Python packages in a separate repo implementing the required functionalities and deploy the workflow via Docker containers.

3.8 T7.7 3DGAN

In this deliverable the machine learning (ML) model, 3DGAN, developed to be integrated into the simulation toolkit, 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. Older previous versions also exist, which are based on the Tensorflow v1 and v2 framework. The model-AI workflow tool integration PoC can be found under the repository[8].

Component name	3DGAN (Deep Learning model for generation of images of calorimeter energy depositions)
Description	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 [R4].
Value proposition	Detector simulations allow scientists to design detectors and perform physics analyses. The simulation toolkit that has

[7] https://github.com/interTwin-eu/DT-Virgo-notebooks
[8] https://github.com/interTwin-eu/itwinai/tree/41-integrate-cern-use-case/use-cases

	been developed and performs particle physics simulations based on Monte Carlo (MC) methods is Geant4. 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. 3DGAN consists of a fast alternative to MC, with remarkable results in terms of 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 [R4].
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 Contacts: Kalliopi Tsolaki (**kalliopi.tsolaki@cern.ch**), Sofia Vallecorsa (**sofia.vallecorsa@cern.ch**)
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 3DGAN component will be integrated into the particle simulation application. 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.

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 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 LeakyRelu[9] activation function is used for the discriminator layers while the Relu[13] 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. 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.

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

The dataset used for studying and developing the 3DGAN model [R4] (**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 ROOT[10] 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

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

[10] ROOT: https://root.cern/

passed to the GAN model (currently developed using Tensorflow v1 and v2[11] , as well as in PyTorch Lightning[12]) for training. The hyperparameter optimization (HPO) tuning processes will be used for searching for the best set of model hyperparameters (e.g. AutoML[13], Optuna[14] etc.). The validation process will verify the performance through a set of physics-motivated steps, both at single image quality level and at the sample level. Finally, the model will be converted into ONNX[15] format and used for inference within the Geant4 application.

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 will be 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 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 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. At inference time, a secondary validation will be performed by the Geant4 application to ensure that the fast simulation is accurate after mapping the inferred energies to positions in the calorimeter.

Concerning the particle simulation framework component of our thematic module, there have been testbeds developed that are incorporating different ML models than the 3DGAN. Therefore, our future efforts will focus on 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 Gitlab[16]. The ML model used in this example is a Variational Autoencoder (VAE), trained externally in Python on full Geant4 detector simulation response data.

3.8.2 Future plans

The development of the thematic module will continue, focusing on aspects such as parallel model training, hyperparameter optimization and model validation. Studies will be conducted on existing solutions for parallel training and hyperparameter optimization that will lead to the selection and implementation of the best solution based on the specific use case. In collaboration with the HEP community, different validation techniques will be studied with the goal to identify the technique most aligned to the

[11] Tensorflow: https://www.tensorflow.org/

[12] PyTorch Lightning: https://lightning.ai/

[13] AutoML: https://www.automl.org/automl/

[14] Optuna: https://optuna.org/

[15] ONNX: https://onnx.ai/

[16] https://gitlab.cern.ch/geant4/geant4/-/tree/master/examples/extended/parameterisations/Par04

needs of the specific fast detector simulation use case. Moreover, we'll continue with the integration of the 3DGAN model with the simulation framework, as well as with the integration of our thematic module with the other DTE modules, such as the AI workflow toolkit.

4 Summary of Integration Status

4.1 T7.1

A copy of the openQxD code is being tested with the SQAaaS module being developed in T6.2. A small test data set has been given to DESY for data lake testing within T5.2 and there will be further integration when the HPC storage at VEGA is integrated into the data lake testbed. The normflow package is standalone and is the entire software basis for the use case T4.1. We do anticipate potential integration with other WPs when the time comes to deploy simulations on larger systems.

4.2 T7.2

Currently there is no integration with other modules in development within interTwin. One of the main reasons for this is that most of our work in the first year was just a proof of concept roughly implemented in Python (three out of four currently released components are in Python and the main, C++ module is in its current form just a test of the overall architecture), so by necessity all of the work was done within our local group. However, as the work on the main production version progresses this year, we are planning to work on integration with the DTE core modules.

4.3 T7.3

At the moment there is no integration with the DTE core modules and the proof of concept runs on local resources at the INFN Torino computing centre. The main showstopper is the fact that the required input data, the auxiliary channels of the Virgo interferometer, is proprietary and work is in progress to publicly release a subset of those to be used in the integration activities.

4.4 T7.7

During the first year of the project there have been tests on running a previous version of 3DGAN model on the FZ Juelich resources provisioned in the framework of partnership between CERN and FZ Juelich under T6.5: AI workflow. Moreover, as a PoC the training process of our model has been integrated with the AI workflow DTE core module developed in T6.5, itwinai framework, and can be found under the following repository: **https://github.com/interTwin-eu/itwinai/tree/dev** . We remain in collaboration with our colleagues from CERN and the rest of interTwin partners to continue the DTE core modules integration efforts for the coming year.

5 Conclusions

The DTE's physics thematic modules covering tasks T7.1, T7.2, T7.3, and T7.7 have been continuously developed and improved throughout the first year of the interTwin project.

In an earlier report the scientists involved in the analysis activities of WP7 identified the technical requirements that are important for the development of the thematic modules, particularly as they relate to the use cases of WP4. Basic information about the 8 thematic modules currently under development in the physics domain have been presented in this report along with release notes and future plans. Some modules have stable releases with plentiful documentation such as the openQxD and GWpy modules. Others have unstable releases that are under development with some public documentation such as the 3DGAN, normflow, and Pulsar modules. T7.3 anticipates the creation of more WP-specific modules in the future. New modules as well as updated versions of the ones presented here will be reported in the deliverable D7.8 "Final version of the thematic module for the physics domain".

The lack of integration with other WPs in some of the modules is largely due to the early stage of development most of the modules find themselves in. As the thematic modules develop and progress over the coming months towards production level running and deployment, they may become better situated to exploit the core module functions, which will have similarly progressed during that time frame. The DTE core modules developers have been receiving feedback and guidance from the physics (and environment) domain developers to ensure they have the functionality and flexibility developers require. This exchange of information is vital and should continue to be a priority moving forward.

6 References

Reference	
No	**Description / Link**
R1	**interTwin D7.2 Report on requirements and thematic modules definition for the physics domain** (V1 Under EC review). K. Tsolaki et al., (2023). DOI: 10.5281/zenodo.8036996
R2	**openQ*D code: a versatile tool for QCD+QED simulations**. I. Campos, P. Fritzsch, M. Hansen, M. K. Marinkovic, A. Patella, A. Ramos & N. Tantalo The European Physical Journal C , 80, Article number: 195 (2020) DOI: https://doi.org/10.1140/epjc/s10052-020-7617-3
R3	**Flow-based generative models for Markov chain Monte Carlo in lattice field theory**. M. S. Albergo, G. Kanwar, and P. E. ShanahanPhys. Rev. D 100, 034515 (2019) DOI: https://link.aps.org/doi/10.1103/PhysRevD.100.034515
R4	**Fast simulation of a high granularity calorimeter by generative adversarial networks.** Khattak, G.R., Vallecora, S., Carminati, F. *et al*. Eur. Phys. J. C 82, 386 (2022). DOI: https://doi.org/10.1140/epjc/s10052-022-10258-4
R5	**Generative Adversarial Networks.** Ian J. Goodfellow et al. https://arxiv.org/abs/1406.2661 DOI: https://doi.org/10.48550/arXiv.1406.2661