D4.3 Advances in HPDA and AI for Global Challenges

Date: May 23, 2024

Document Identification			
Status	Final	Due Date	30/04/2024
Version	1.0	Submission Date	23/05/2024

Related WP	WP4	Document Reference	D4.3
Related Deliverable(s)	D4.1, D5.3	Dissemination Level (*)	PU
Lead Participant	ICCS	Lead Author	Giorgos Stamou (ICCS)
Contributors	ICCS, PSNC, UNISTRA, MTG, SZE, FAU	Reviewers	Ákos Kovács (SZE)
			Dennis Hoppe (USTUTT)

Keywords:
Artificial Intelligence (AI), High-Performance Data Analytics (HPDA), data analytics, neural networks, machine learning, postprocessing data analysis

Document Information

List of Contributors	
Name	Partner
Giorgos Stamou	ICCS
Dimitrios Tsoumakos	ICCS
Angeliki Dimitriou	ICCS
George Filandrianos	ICCS
Nikolaos Chalvantzis	ICCS
Vasiliki Kostoula	ICCS
Michal Kulczewski	PSNC
Javier Cladellas	UNISTRA
Luis Torres	MTG
Zoltán Horváth	SZE
Harald Koestler	FAU

Document History			
Version	Date	Change editors	Changes
0.1	08/03/2024	Angeliki Dimitriou, George Filandrianos, Nikolaos Chalvantzis (ICCS)	Table of Contents added.
0.15	10/03/2024	Marcin Lawenda (PSNC) Harald Koestler (FAU)	ToC, timeline and responsibilities approved
0.2	04/04/2024	Nikolaos Chalvantzis, Vasiliki Kostoula (ICCS), Michal Kulczewski (PSNC), Javier Cladellas (UNISTRA)	First draft of section 2 submitted. Sub-section 4.1 submitted. Sub-section 4.3 submitted.
0.3		George Filandrianos, Angeliki Dimitriou (ICCS) Luis Torres (MTG)	First draft of Section 3 submitted. Subsection 4.4 submitted.
0.4		Zoltán Horváth (SZE)	Subsection 4.2 submitted.

0.5		Angeliki Dimitriou, George Filandrianos, Nikolaos Chalvantzis (ICCS)	Drafts of sections 1, 5, 6 submitted.
0.6		Angeliki Dimitriou, George Filandrianos, Vasiliki Kostoula, Nikolaos Chalvantzis (ICCS)	Integrated available SZE sections.
0.7	30/04/2024	Angeliki Dimitriou, George Filandrianos, Vasiliki Kostoula, Nikolaos Chalvantzis (ICCS)	Format finalised.
0.8	10/05/2024	Angeliki Dimitriou, George Filandrianos (ICCS)	Review comments incorporated for AI.
0.8.2	10/05/2024	Vasiliki Kostoula, Nikolaos Chalvantzis (ICCS)	Review comments incorporated for HPDA.
0.8.3	17/05/2024	Harald Köstler (FAU)	Subsection 4.5 submitted.
0.9	20/05/2024	Zoltán Horváth (SZE)	Submitted remaining subsection of 4.2.
0.98	20/05/2024	Angeliki Dimitriou, George Filandrianos, Vasiliki Kostoula, Nikolaos Chalvantzis (ICCS)	Version to be submitted for formal check
0.99	21/05/2024	Harald Koestler	Quality assurance check
1.0	23/05/2024	Marcin Lawenda	Final check

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable Leader	Giorgos Stamou (ICCS)	20/05/2024
Quality Manager	Harald Koestler (FAU)	21/05/2024
Project Coordinator	Marcin Lawenda (PSNC)	23/05/2024

Table of Contents

List of Tables

List of Figures

List of Acronyms

Insert here all the acronyms appearing along the deliverable in alphabetical order.

Abbreviation / acronym	Description
AI	Artificial Intelligence
API	Application Programming Interface
AUC	Area Under the Curve
CFD	Computational Fluid Dynamics
CPU	Central Processing Unit
CSV	Comma-Separated Values
DAG	Directed Acyclic Graph
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DNN	Deep Neural Network
Dx.y	Deliverable number y belonging to WP x
EC	European Commission
EDA	Exploratory Data Analysis
EuroHPC JU	EuroHPC Joint Undertaking
GCN	Graph Convolutional Network
GIF	Graphics Interchange Format
GIS	Geographic Information System
GNN	Graph Neural Network
GPU	Graphics Processing Unit
HPDA	High Performance Data Analytics
HDFS	Hadoop Distributed File System
HPC	High-Performance Computing
ICGC	Cartographic and Geologic Institute of Catalonia
JSON	JavaScript Object Notation
LOD	Level of Detail
MLP	Multilayer Perceptron
NLP	Natural Language Processing
PCA	Principal Component Analysis
PID	Process Identifiers

RES	Renewable Energy Sources
ROC	Receiver Operating Characteristic (curve)[1]
SDLC	Systems/Software Development Life Cycle
SQL	Structured Query Language
UAP	Urban Air Project
UTM	Universal Transverse Mercator
WP	Work Package
WSF	Wind Sheltering Factor
WUI	Wildland-Urban Interface

[1] a graphical plot that illustrates the performance of a binary classifier model.

Executive Summary

Deliverable D4.3 develops foundational methodologies for integrating High-Performance Data Analytics (HPDA) and Artificial Intelligence (AI) into the workflows of pilot applications addressing global challenges. This document provides a blueprint for enhancing performance through advanced data analytics and AI across various stages of application deployment, from model creation to result post-processing. Throughout the first phase of the HiDALGO2 project, these methodologies have been applied to pilot applications running on major computational platforms. The focus has been on improving accuracy, reducing simulation times, and enhancing result interpretation, emphasising generic applicability to ensure that the methodologies can be adapted across different domains.

Initial findings suggest a promising potential for developed AI solutions to significantly enhance both the Wildfire and Urban Buildings pilots in terms of accuracy and efficiency. Specifically, both challenges and strengths were uncovered concerning wildfire data. While the unique trajectories of wildfires pose obstacles for similarity algorithms, the consistent alignment of fire data in PCA analysis underscores the potential significance of handcrafted features, particularly those extracted during the initial timestep, for understanding fire behaviour patterns. For Urban Buildings Model, early GNN model experiments consistently yield high AUC scores, suggesting robustness on affected building prediction even without full optimisation, with trends indicating the efficacy of undirected graphs, and promising correlation between predicted affected buildings and their proximity. These solutions are poised for further enhancements to comprehensively support each pilot within the HiDALGO2 project. In the future, they will undergo adaptation and benchmarking within the HiDALGO2 infrastructure to ensure optimal performance and integration.

1 Introduction

1.1 Purpose of the document

The deliverable “D4.3 Advances in HPDA and AI for Global Challenges” is developed within the framework of WP4 and aims to enhance HiDALGO2 pilot applications and supportive actions through HPDA/AI solutions. This document outlines the HiDALGO2 HPDA/AI methodology that ensures the reproducibility and validation of the gathered results. It further delineates the support provided to HiDALGO2 pilots by harnessing HPDA capabilities alongside AI techniques. This collaboration aims to refine multiple workflow stages, including model creation, data pre-processing, and result interpretation.

1.2 Relation to other project work

Deliverable D4.3 summarises initial findings regarding the pilot applications developed in WP5, described in deliverable D5.3, “Research Advancements for the Pilots,” using HPDA and AI. Deliverable D4.3 is related to activities within WP4, detailing HPDA and AI methods to support HiDALGO2 applications, which will eventually leverage the infrastructure described in D4.1, “Data Management and Coupling Technologies.” It is the first of a series of reports focusing on enhancing applications by HPDA and AI (D4.3 in M18, D4.3 in M36, and D4.3 in M48).

1.3 Structure of the document

This document is structured in 6 major chapters.

Chapter 2 presents the data analytics approach and details the HPDA tools and frameworks.

Chapter 3 presents the AI methodologies and frameworks that are utilised.

Chapter 4 presents detailed descriptions of the methodologies applied across all pilot projects.

Chapter 5 presents the adaptation to HiDALGO2 computational resources.

Chapter 6 presents the conclusions of the deliverable.

2 HiDALGO2 HPDA methodology

Global Challenge applications, by nature, produce massive data. Task 4.2 undertakes the challenge of utilising highly scalable analytics runtimes and algorithms to process data in time to be handled or provisioned by HPC applications. In this section of the present technical report, we present our general approach to developing High-Performance Data Analytics (HPDA) applications and our selected toolset comprising state-of-the-art frameworks and technologies.

2.1 Approach to Data Analytics

In this sub-section, we will delve into our approach to the challenge of developing Data Analytics applications within the HiDALGO2 framework and its pilots. The methodology that has guided our efforts will be discussed and thoroughly documented, providing a high-level overview of the steps taken in each of our use cases.

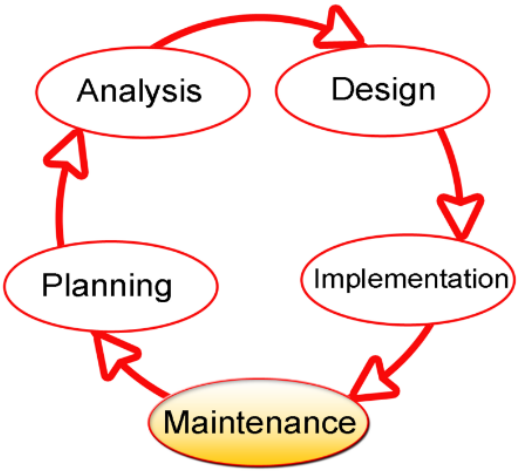

Figure 2-1 The classic SDLC model

In general, to assist us in developing multiple applications tailored to address the diverse scenarios and use cases arising from the HiDALGO2 pilots, we've followed a roadmap consistent with a classical approach to the **systems/software development life cycle** (SDLC) [1] model depicted in Figure 2-1. The actions involved in each of the development phases are detailed in the following paragraphs.

Planning: In the HiDALGO2 HPDA development context, the initial planning phase involves two key tasks: i) gathering data and ii) documenting use case scenarios. These activities are carried out through bilateral meetings with pilot owners. Our conversations usually start with a deep dive into the objectives of the pilots. Furthermore, we use data samples shared by pilot owners to kick off discussions on potential use cases for HPDA. The primary goal in this phase is to pinpoint issues that HPDA methods can address. Later iterations of this phase involve collecting feedback from the pilot owners and reassessing the development progress.

Analysis: A thorough requirement analysis is conducted once the identified challenges are established. This analysis involves deconstructing the problems into smaller segments and scrutinising existing literature for cutting-edge solutions. This review encompasses both algorithmic and theoretical perspectives, as well as evaluations of technical standards, including scalability and consistency.

Design: During this stage, we proceed with selecting tools for our implementation. Following a comprehensive analysis of the identified challenges in the preceding phase, we determine the algorithms and frameworks to be employed. Moreover, in certain instances where the use of basic prototypes to validate our approach is deemed essential, the design phase also encompasses the blueprinting of these small-scale components.

Implementation: The implementation process typically unfolds in several distinct phases. In the case where a small-scale prototype is involved, we commence by developing that, employing a portion of the available data to assess the accuracy and robustness of our methodologies. Once the efficacy of our methods has been verified, further iterations can focus on advancing to implementations reliant on distributed systems and execution frameworks, as delineated in subsection 2.2.

Maintenance: The maintenance phase is initiated upon deployment of a full-scale prototype of the HPDA application on the designated HiDALGO2 HPDA infrastructure. As an integral part of this phase, we undertake the necessary measures for deploying our applications in an environment accessible to stakeholders within HiDALGO2. Moreover, this phase encompasses addressing issues such as bug-fixing and implementing minor updates aimed at improving the quality-of-life and user experience of the deployed applications.

It is important to note that the development process involves multiple iterations of this cycle. During certain iterations, some documented phases might be omitted to enhance flexibility and expedite progress. Nonetheless, when and where it might be necessary, we plan to regularly revisit and reassess our design and implementation choices and actions. For each of the HiDALGO2 pilots, we will report the actions taken so far and explain our progress and future steps in the context of the development cycle presented here.

2.2 HPDA tools and frameworks

In the rapidly evolving landscape of High-Performance Data Analytics, the significance of selecting the most appropriate tools and frameworks cannot be overstated. The effectiveness of data analytics projects hinges on this critical decision-making process, where several factors must be considered to ensure the success of our endeavours:

- Scalability
- Performance
- Ease of Integration
- Adaptability

Scalability stands as a paramount consideration. As datasets grow exponentially, our chosen tools' ability to scale effectively determines the feasibility and efficiency of processing large volumes of data. The capacity to maintain performance levels while

scaling up or down according to data volume and computational demand is essential for the adaptability and sustainability of all data analytics projects.

Equally important is **performance**; the selected tools must not only be able to handle large datasets but also do so in a timely manner. This provides the agility required in today's fast-paced analytical environments, where the speed of data processing can significantly impact decision-making and operational efficiency. This aspect can prove life-critical in certain scenarios, such as in the Wildfires pilot use case.

Another crucial consideration is the **ease of integration**. Data analytics ecosystems are inherently complex, consisting of various data sources, processing engines, and storage systems. The tools and frameworks we select must seamlessly integrate with existing components, facilitating a cohesive and efficient workflow. Additionally, we aim to be able to integrate our services with the existing software infrastructure they will need to interact with.

The last element to be considered is the software infrastructure's **adaptability to future technological advancements** and data analytics trends. The landscape of HPDA is constantly changing, and our toolkit must be flexible enough to adapt to new requirements and capabilities over time.

Given these considerations, it is imperative not only to select tools that best meet these criteria but also to understand the landscape of available technologies. The subsequent sections will introduce the tools chosen for their distinct advantages in HPDA – such as Hadoop Filesystem, Apache Spark, Apache Flink and Ray – but also explore viable alternatives that were considered.

2.2.1 Hadoop Distributed File System

The Hadoop Distributed File System (HDFS) [2] is a distributed file system designed to store and manage large volumes of data reliably and efficiently across clusters of computers. In the architecture of our HPDA framework, we have strategically chosen the HDFS, a core component of the Apache Hadoop [3] ecosystem, as our primary file system for storing data. This decision is rooted in HDFS's unparalleled ability to store and manage vast amounts of data across a distributed computing environment. As a cornerstone of modern big data and analytics solutions, HDFS provides a robust and scalable foundation for our data storage needs, aligning perfectly with our project's objectives and technical requirements. Its commanding position in the Hadoop ecosystem, the seamless integration it provides with other components in our HPDA technology stack – such as Apache Spark [4] (cf. Section 2.2.2) – and the optimisations it supports make it an ideal candidate.

Scalability: HDFS exemplifies scalability, designed from the ground up to support massive data sets. It achieves this by distributing data across multiple nodes within a cluster, facilitating not only the storage of large volumes of data but also enabling efficient parallel processing. This approach ensures seamless scalability, as the

system can be expanded by adding more nodes to the cluster without any downtime or loss of performance.

Performance: HDFS performance is optimized for high-throughput access to data, especially suited for applications that manage large datasets. It is built on a batch processing model, where data is written once and read many times, thus minimising disk seek times. Engineered to operate on commodity hardware, HDFS is both cost-effective and adept at handling enormous datasets with high efficiency.

Ease of Integration: HDFS's integration capabilities are extensive, offering compatibility with a broad spectrum of data processing frameworks within the Apache Hadoop ecosystem, such as MapReduce, Apache Hive, and Apache HBase. Its open-source nature is supported by a vibrant community, which provides ample documentation and tools for managing data within HDFS, thereby simplifying integration efforts.

Technical Considerations: Technically, HDFS operates on a master/slave architecture, with the *NameNode* managing the file system namespace and *DataNodes* managing storage on the nodes they run on (depicted in Figure 2-2). This setup supports file storage in a namespace. User data can be stored in files, internally divided into blocks stored across *DataNodes*. This design ensures fault tolerance and system reliability by replicating data across multiple nodes.

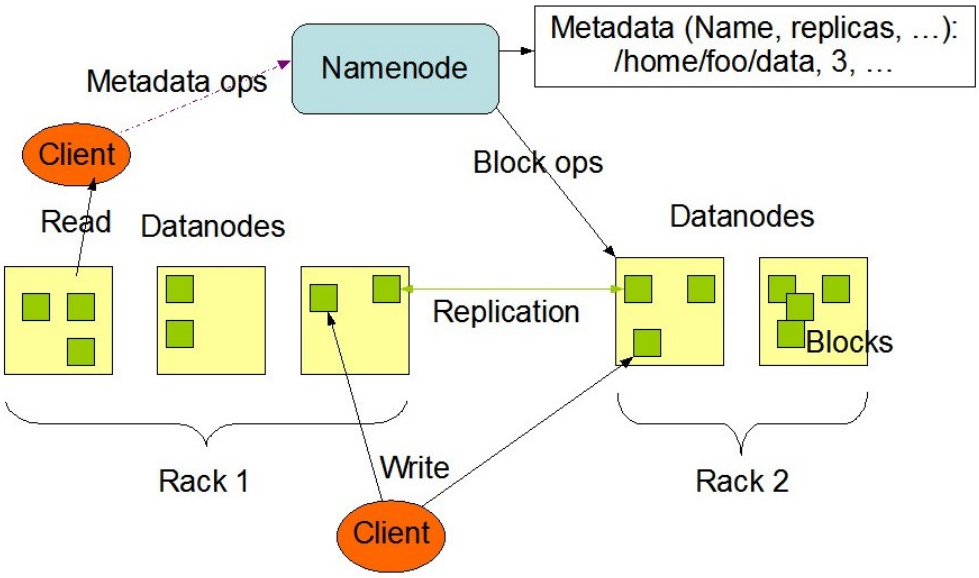

Figure 2-2 HDFS Architecture

Adaptability to Future Technological Advancements: The architecture of HDFS is both robust and adaptable, allowing it to stay relevant in the face of evolving data storage and processing technologies. Its compatibility with diverse data types and formats, along with integration capabilities with emerging machine learning and analytics tools, ensures that HDFS remains at the forefront of data processing technology.

By selecting the Hadoop Distributed File System as our foundational storage solution, we underscore our commitment to leveraging leading-edge technology to meet the rigorous demands of high-performance data analytics. This choice reflects our strategic focus on scalability, performance, ease of integration, and future-proofing our data analytics infrastructure.

2.2.2 Apache Spark

Apache Spark [4] is a powerful open-source unified analytics engine for large-scale data processing, with built-in modules for streaming, SQL, machine learning, and graph processing. Spark has emerged as one of the most significant technologies in the big data analytics domain, offering both versatility and high performance for data processing tasks.

Scalability: Apache Spark is renowned for its exceptional scalability. It is capable of processing petabytes of data across thousands of nodes. It achieves this through its advanced DAG (Directed Acyclic Graph) execution engine, which optimises workflow and resource allocation. Spark's in-memory computation capabilities drastically reduce the need for disk I/O, enabling it to scale efficiently and handle increasing data loads without sacrificing performance.

Performance: Spark is designed for speed, offering up to 100 times faster performance for certain applications, particularly those that can benefit from in-memory processing, compared to other big data technologies like Hadoop MapReduce [5]. This performance advantage is especially notable in data analytics tasks, where large datasets are common. Spark's ability to cache data in memory between operations can significantly accelerate complex data transformations and aggregations. This capability is crucial for analytics workflows involving repetitive data processing tasks, such as joining large datasets, filtering, and summarisation, where traditional disk-based processing would incur significant overheads. By optimising these operations, Spark ensures rapid insights into data, enhancing the productivity and efficiency of data analytics pipelines.

Ease of Integration: Spark offers robust integration capabilities, particularly with the Hadoop ecosystem, including seamless compatibility with HDFS. In conjunction with other HPDA tools, Apache Spark functions particularly well when integrated with the Hadoop Distributed File System (HDFS), where it can leverage HDFS for massive storage capabilities while providing powerful processing speeds. This symbiosis enhances both storage and analysis capabilities, making Apache Spark an indispensable component of any data analytics framework. Moreover, Spark supports multiple data sources like JSON [6], CSV [7], and Parquet [8], and can be integrated

with higher-level libraries and frameworks, enhancing its versatility and ease of use in diverse data environments.

Technical Considerations: Apache Spark's architecture is built around the concept of RDDs (Resilient Distributed Datasets), which are fault-tolerant collections of elements that can be operated on in parallel. Its cluster mode offers several deployment options, including standalone, Apache Mesos, and Kubernetes, with the

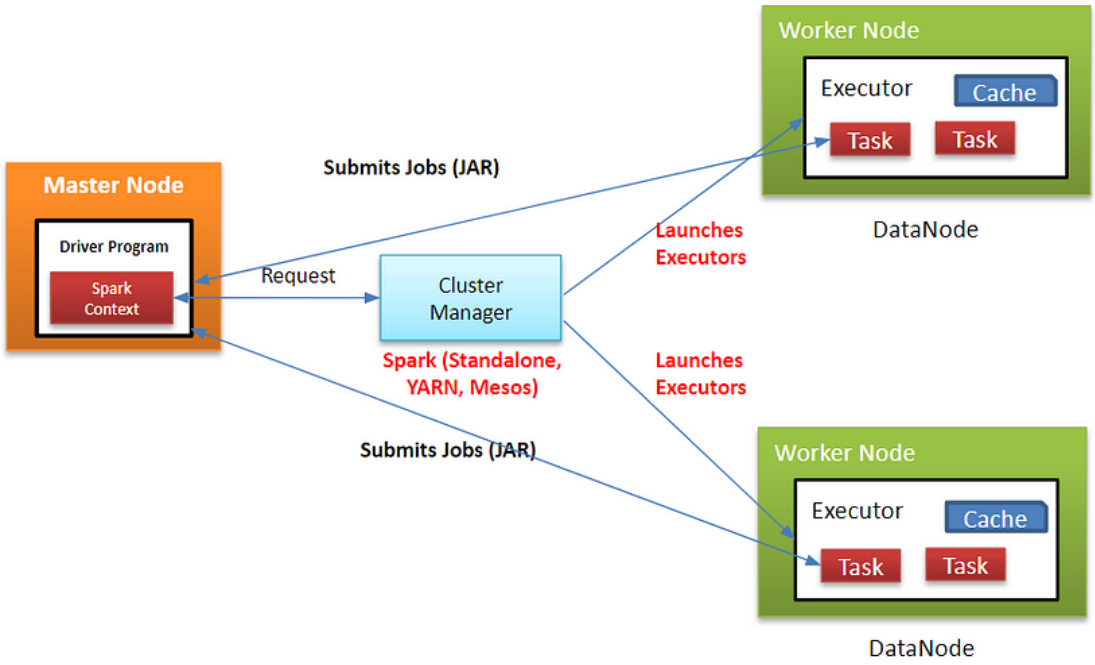

Figure 2-3 Apache Spark Architecture

ability to run atop HDFS (cf. Section 0), providing a flexible and powerful computing platform. Spark's API is available in multiple programming languages, including Scala, Java, Python, and R, making it accessible to many developers and data scientists.

Some key components of the Apache Spark architecture, depicted in Figure 2-3, are the following:

- **Driver Program**: The driver program is responsible for orchestrating the execution of the Spark application. It communicates with the cluster manager to acquire resources and coordinate the execution of tasks.
- **Cluster Manager**: Spark can run on various cluster managers such as Apache Mesos [9], Hadoop YARN [10], or Spark's standalone cluster manager. The cluster manager allocates resources and manages the execution of tasks across the cluster.
- **Executors**: Executors are worker nodes in the Spark cluster responsible for executing tasks. Each executor is assigned a portion of the cluster's resources (CPU cores, memory) and is responsible for running computations and storing data in memory or disk.

- **Tasks**: In Apache Spark, tasks are the smallest units of work that are scheduled by the Spark scheduler and executed by the executor nodes in a Spark cluster. Tasks are responsible for processing partitions of data in parallel across the cluster.

Adaptability to Future Technological Advancements: Spark's design is inherently adaptable, with a vibrant community continuously working on enhancements and integrations to keep pace with the rapid advancements in data analytics. Its modular architecture allows for the easy addition of new functionalities and improvements in many areas.

By incorporating Apache Spark into our data analytics framework, we harness its high-performance processing capabilities alongside its scalability and ease of integration, ensuring that our analytics operations are not only efficient but also poised for future expansions and technological integrations.

2.2.3 Apache Flink

Apache Flink [11] is an open-source stream processing framework with powerful batch processing capabilities designed to handle large-scale data analytics and real-time processing tasks. Known for its low-latency, high-throughput processing, Flink is a versatile tool for building real-time analytics applications and data pipelines.

Scalability: Apache Flink is built for horizontal scalability, allowing it to handle massive data volumes with ease. It achieves scalability through its distributed stream processing model, where data is partitioned and processed in parallel across a cluster of machines. Flink's dynamic task allocation and fault tolerance mechanisms ensure that processing resources are efficiently utilised, making it suitable for both small-scale and large-scale deployments.

Performance: Flink is optimised for low-latency and high-throughput processing, making it ideal for real-time analytics and event-driven applications. It achieves high performance through its pipelined execution model, where data is processed in a continuous stream, minimising processing latencies. Flink's support for stateful computations enables it to maintain its application state efficiently, facilitating real-time complex event processing and analytics.

Ease of Integration: Apache Flink offers seamless integration with other data processing frameworks, including HDFS (0) for distributed storage and Apache Spark (2.2.2) for batch processing. Its connectors for various data sources and sinks enable easy integration with external systems, allowing data to be ingested and processed from diverse sources. Flink's support for standard APIs like SQL and DataStream API simplifies application development, while its ecosystem of libraries and tools enhances productivity and flexibility.

Technical Considerations: Flink's architecture is based on a distributed dataflow model, where data is processed as streams of records flowing through operators. Its runtime consists of a cluster manager responsible for resource allocation and coordination, task managers executing processing tasks, and a job manager coordinating the overall execution of jobs. Flink provides support for both batch and stream processing, enabling unified analytics across different data types.

In terms of system architecture, a depiction of an Apache Flink cluster is presented in Figure 2-4. The cluster contains two types of nodes: the **JobManager** and the **TaskManagers**. The JobManager is responsible for coordinating and scheduling jobs – it decides when to schedule the next task (or set of tasks), reacts to finished tasks or execution failures, coordinates checkpoints, and coordinates recovery from failures, among others. The JobManager assigns tasks to TaskManagers and monitors their progress. The TaskManagers execute the actual processing tasks. The level of parallelism a Flink cluster can offer depends on the total number of task slots available.

Adaptability to Future Technological Advancements: Apache Flink is designed to adapt to evolving data processing requirements and technological advancements. Its support for stateful processing, event time semantics, and advanced windowing operations positions it as a leading framework for complex event processing, real-time analytics, and machine learning. Flink's active community and commitment to innovation ensure that it remains at the forefront of stream processing technology.

In conjunction with other HPDA tools, Apache Flink integrates seamlessly with HDFS for distributed storage and Apache Spark for batch processing. Its compatibility with these frameworks allows for a cohesive analytics ecosystem, where data can be processed and analysed in real-time or batch mode, depending on the requirements of the application.

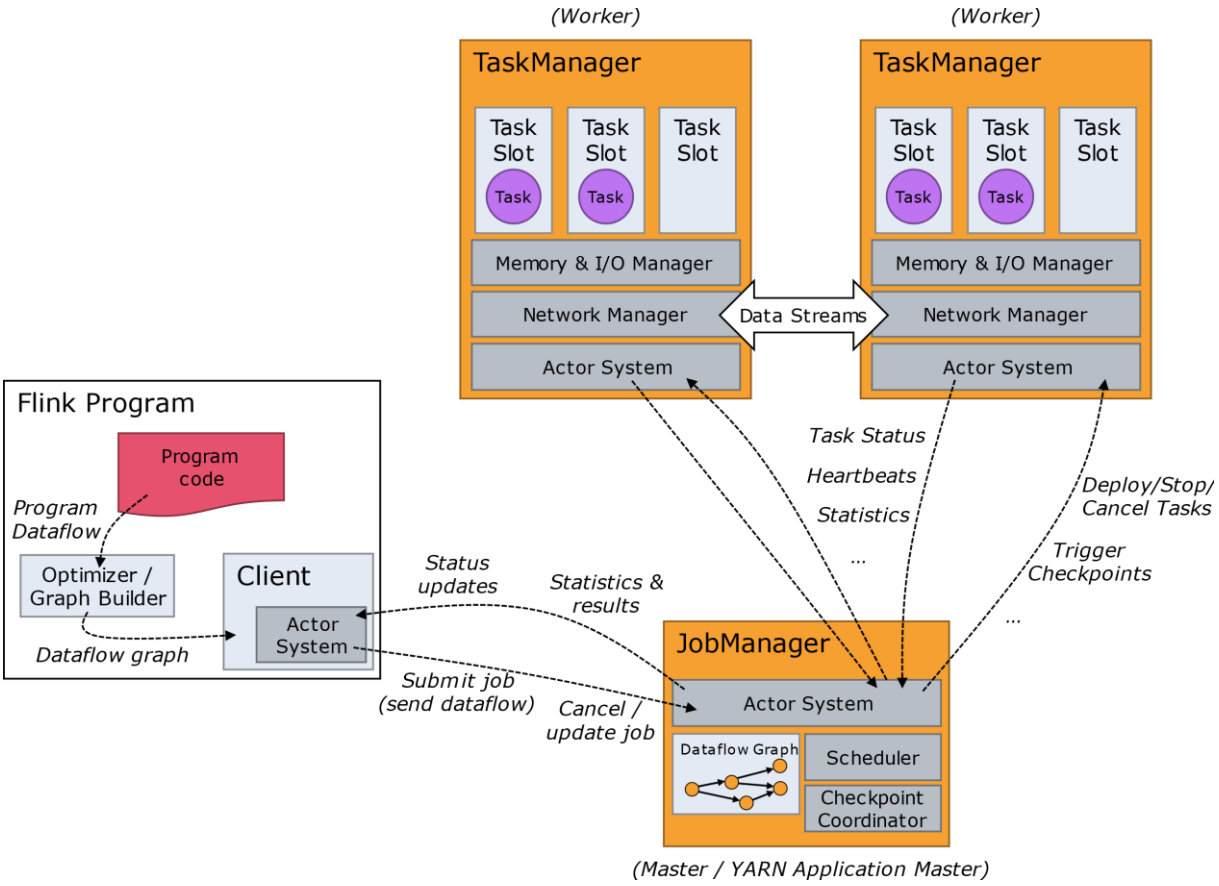

Figure 2-4 Apache Flink Architecture

2.2.4 Ray

Ray [12] is an open-source distributed execution framework designed for high-performance and scalable Python applications. Developed by the RISELab at UC Berkeley, Ray provides a flexible and efficient platform for building and deploying distributed applications, ranging from reinforcement learning algorithms to distributed data processing tasks.

Scalability: Ray excels in scalability, offering seamless scaling capabilities for Python applications across a cluster of machines. It achieves scalability through its distributed execution model, where tasks are dynamically allocated and executed across a cluster of workers. Ray's lightweight and flexible architecture enables it to handle diverse workloads, from small-scale experiments to large-scale production deployments, with ease.

Performance: Ray is optimized for high performance, leveraging efficient task scheduling and resource management mechanisms to minimize execution latencies. Its asynchronous execution model allows tasks to execute concurrently, maximizing resource utilization and throughput. Ray's support for distributed data processing and

computation caching further enhances performance, making it suitable for demanding analytics and machine learning workloads.

Ease of Integration: Ray offers seamless integration with existing Python libraries and frameworks, allowing developers to leverage their favourite tools and APIs within Ray-based applications. It provides native support for popular Python libraries such as NumPy, Pandas, and TensorFlow, enabling easy adoption and integration into existing workflows. Ray's intuitive API and comprehensive documentation streamline the development process, reducing the learning curve for new users.

Technical Considerations: Ray's core abstraction is the Actor model, where lightweight, stateful objects (Actors) communicate asynchronously via message passing. This model enables flexible and scalable distributed computation, allowing developers to express complex distributed workflows with ease. Ray's task scheduling and resource management capabilities ensure efficient utilisation of cluster resources, optimising performance and scalability.

Adaptability to Future Technological Advancements: Ray is designed to adapt to evolving technological advancements and requirements in distributed computing. Its modular architecture and extensible API make it easy to integrate with new technologies and frameworks, ensuring compatibility with emerging trends in data analytics and machine learning. Ray's active community and ongoing development efforts ensure that it remains at the forefront of distributed computing innovation.

Compatibility with Other Tools: Ray complements other HPDA tools, such as HDFS and Apache Spark, providing a flexible and scalable platform for distributed Python applications. It seamlessly integrates with Apache Spark for distributed data processing and analytics, allowing Python-based applications to leverage Spark's capabilities. Similarly, Ray can interface with HDFS for distributed storage, enabling efficient data access and processing within Ray-based applications.

2.2.5 Alternative Tools and Final Selection

In our selection process, several alternative tools were evaluated alongside our chosen technologies. Each offers unique features and capabilities that are beneficial under different circumstances. Here, we explore some of these alternatives, comparing them to our selected tools to elucidate why they were not chosen as our primary solutions.

Dask: As an alternative to Apache Spark, Dask supports complex parallel computations for analytics, particularly effective in the Python ecosystem. While Dask excels in integrating with Python data science tools like Pandas and NumPy, it lacks the extensive JVM ecosystem support that Spark provides, which is crucial for our existing infrastructure.

NVIDIA RAPIDS: Focused on GPU acceleration, NVIDIA RAPIDS could be an alternative to Ray for specific machine learning and data processing tasks. It excels in environments that can leverage GPU resources for substantial performance improvements. However, its dependency on specific hardware configurations makes it less versatile and scalable across diverse deployment scenarios compared to Ray.

Apache Storm: For real-time data processing, Apache Storm presents a robust alternative to Apache Flink. It is known for its high processing speed and fault tolerance. Nonetheless, Flink provides superior consistency and state management features, which are critical for our complex real-time analytics requirements.

Considering these alternatives ensures a well-rounded decision-making process. Each tool was assessed not only on its technical merits but also on its alignment with strategic goals and operational contexts. The final selection of tools for high-performance data analytics was based on a thorough evaluation of specific use cases and the characteristics of the data involved. This evaluation also took into account the available expertise, ensuring that the chosen tools could integrate well with existing workflows and enhance efficiency. By aligning tool choices with both data requirements and team capabilities, the aim is to build an analytics infrastructure that is robust and adaptable to future needs.

3 HiDALGO2 AI methodology

Within the framework of the HiDALGO2 project, the integration of Artificial Intelligence (AI) stands as a pivotal driver for enhancing the efficiency and effectiveness of pilot applications developed in WP5, along with supportive actions in WP6. This section delineates the structured approach employed in harnessing AI techniques to bolster pilot workflows, aligning with the project's overarching objectives. With a focus on model creation, data pre-processing, model execution, and result post-processing, the HiDALGO2 AI methodology aims to unlock the full potential of AI across various stages of the pilot workflows. By leveraging both machine learning (including deep learning) and rule-based approaches, this methodology anticipates notable enhancements in accuracy, simulation time, time-to-first-fit, and result interpretation. By seamlessly integrating AI tasks into simulation applications or workflows, these hybrid workflows offer unprecedented opportunities to expedite simulations while maintaining acceptable levels of accuracy.

The process of developing and utilising AI methodologies encompasses five essential components: Model Formalization, Data acquisition, Exploratory Data Analysis (EDA), Model Design, and Model Execution. The graphical illustration presented in Figure 3-1 depicts a schematic depiction of the iterative process involved in an artificial intelligence (AI) project cycle. Within this cycle, Model Formalization and Data Acquisition emerge as pivotal stages, exerting profound influence over subsequent procedural facets. The initial phases have been subjected to thorough examination in the following sections, concurrent with the description of High-Performance Data Analytics (HPDA) procedures. Consequently, the focus herein is directed towards delineating the remaining three sequential steps integral to the development of an AI project.

EDA serves as the initial phase, allowing stakeholders to gain critical insights into the underlying data landscape, and facilitating informed decisions regarding feature selection, anomaly detection, and relationship identification. Model Design follows, wherein suitable algorithms and architectures are selected and tailored to address specific objectives, ensuring optimization for accuracy, efficiency, and interpretability. Finally, Model Execution marks the operational deployment of the designed models, involving rigorous evaluation and validation to assess performance against predefined metrics and benchmarks, while also enabling scalability and adaptability to real-world scenarios. These steps are fundamental in developing AI methodologies as they provide a structured framework for understanding data, designing effective models, and deploying them in operational workflows. Through EDA, Model Design, and Model Execution, AI methodologies are refined iteratively, fostering innovation, and driving impactful outcomes across various domains. In the subsequent sections, we delve into each component, elucidating their significance and methodologies within the context of developing AI-driven solutions.

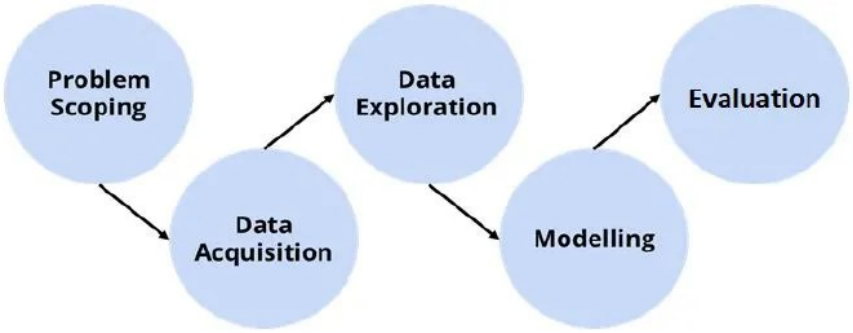

Figure 3-1 AI project pipeline

3.1 Exploratory Data Analysis

Exploratory Data Analysis (EDA) is a critical step in the data science process and is closely aligned with the development of AI methodologies [13]. It allows us to understand the data we are working with, which is crucial for any AI model. By understanding the data, we can make informed decisions about the modelling process, such as which features are important, what type of model to use, and how to tune the model parameters. The questions that this procedure usually must answer are the following.

What are the types of datasets?

The first step in EDA is examining the types of data. This is important because different types of data require different handling techniques. For example, categorical data may need to be encoded, while numerical data may need to be normalised. Most algorithms require a consistent type of input; for example, some algorithms only accept integer inputs, while others may require floating-point data. In the literature, several methods have been proposed for changing the type of data in order to normalise it in a unified way. One common approach for handling categorical values is one-hot encoding. This technique transforms categorical variables into a binary format, with each category represented as a separate binary column. Another method is label encoding, which assigns a unique numerical value to each category. However, it is important to note that these methods have their own advantages and limitations, and the choice of encoding technique should be based on the specific characteristics of the dataset and the requirements of the modelling task.

What is the type distribution of each feature of the dataset?

Next, we examine the data distribution. This step is crucial as it provides insights into the range and central tendencies of our data, which in turn can profoundly influence

the choice of modelling techniques. Different data distributions can impact the modelling process and the performance of models in various ways. For instance, data with a normal distribution, characterised by a symmetrical bell-shaped curve, is often well-suited to certain types of models like linear regression, where assumptions of normality are inherent. In contrast, skewed distributions, such as positively or negatively skewed data, may require transformations or specialised modelling approaches to account for the asymmetry and ensure accurate predictions. Similarly, multimodal distributions, where data exhibits multiple peaks or modes, may necessitate the use of more complex modelling techniques capable of capturing the underlying structure of the data. Additionally, heavy-tailed distributions, characterised by a high frequency of extreme values, may pose challenges for traditional statistical models and may require robust modelling techniques that are less sensitive to outliers. A thorough analysis of the dataset's distribution is essential to selecting the appropriate algorithm for training. Additionally, this analysis can provide useful insights about the dataset, as seen in the case of the MTG pilot, where correlations were identified between the timestep of a fire and commonly used fire shape descriptors.

Are there any missing values in the dataset?

Handling missing values is another common and crucial step in EDA. Missing data poses a pervasive challenge in datasets and can profoundly affect the performance and reliability of an AI model. Failure to address missing values adequately can introduce biases and inaccuracies into the model, compromising its effectiveness in generating meaningful insights. Moreover, the extent of missing values within a dataset can greatly impact its overall quality and necessitate different methodologies for handling them effectively. For instance, datasets with a high proportion of missing values may require more sophisticated imputation techniques or alternative modelling approaches to mitigate the impact on the final results.

Does the dataset contain outliers?

Detecting and managing outliers is another essential aspect of EDA. Outliers, or data points that deviate significantly from the rest of the dataset, can exert a substantial influence on the performance and accuracy of our models. These anomalies have the potential to distort the underlying patterns and relationships within the data, leading to biased or inaccurate model predictions. Therefore, it is imperative to identify outliers and implement appropriate strategies to mitigate their impact during the modelling process. Additionally, the presence and nature of outliers can vary across datasets, necessitating tailored approaches for detection and handling. Robust techniques such as trimming, winsorization, or the use of robust statistical measures can help identify and mitigate the effects of outliers, ensuring the integrity and reliability of the model outputs.

Does the dataset contain duplicate instances?

Handling duplicate data is another critical step in EDA that warrants attention. Duplicate data, if left unaddressed, can introduce biases and challenges in the modelling process, potentially leading to overfitting. Overfitting occurs when a model learns to capture noise or irrelevant patterns in the training data, resulting in excellent performance on the training set but poor generalisation to new, unseen data. By removing duplicate data, analysts can mitigate the risk of overfitting and ensure that the model learns from a diverse set of examples, thereby improving its ability to generalise to unseen data and make accurate predictions. Additionally, eliminating duplicate data enhances the efficiency of the modelling process by reducing redundancy and streamlining the dataset. This ensures that computational resources are utilised optimally and that the resulting model is more robust and reliable.

Are there any biases or correlations among the data's features?

Assessing biases and correlations among features is a crucial aspect of EDA that demands thorough examination. Biases or correlations within the data can significantly impact the performance and reliability of AI models, potentially leading to skewed or inaccurate predictions. Analysts can ensure the integrity and fairness of the model outputs by identifying and addressing biases or correlations during the EDA process. Biases, such as sampling bias or measurement bias, may arise from systematic errors in the data collection process and can introduce inaccuracies or distortions in the modelling results. Similarly, correlations among features can lead to multicollinearity, where two or more variables are highly correlated, making it difficult to distinguish their individual effects on the target variable. Detecting and addressing biases or correlations enables analysts to make informed decisions regarding feature selection, model design, and interpretation of results, ultimately enhancing the reliability and generalizability of the AI models. Moreover, mitigating biases and correlations fosters transparency and fairness in AI-driven decision-making processes, ensuring equitable outcomes across diverse populations and contexts. However, certain correlations among data features may be useful in model development. For instance, in the case of the MTG pilot, the goal is to develop a search algorithm to find the closest fire to a simulated one. Revealed correlations between specific features and fire are valuable in developing this algorithm. For example, if there is a clear correlation between the fire and wind direction, it may be wise to emphasize this feature to enhance the training phase. Additionally, this knowledge is useful for evaluating the model's performance beyond standard metrics. For instance, if the model does not learn this high correlation, there may be an error in the training phase. Furthermore, if there is a spurious correlation between the features, it is crucial to identify it to remove it, add more data to mitigate it, or employ algorithms that automatically ignore such spurious correlations.

Is there a necessity for the normalisation of data?

Assessing the need to normalise the data is a crucial step in EDA that merits careful consideration. Normalising and scaling the data can have a profound impact on the performance and efficacy of AI models. Many machine learning algorithms, particularly those based on distance metrics or gradient descent optimisation, perform more effectively when the features are on a similar scale. Normalisation ensures that all features contribute equally to the model's learning process, preventing features with larger scales from dominating the optimisation process. In literature, for instance, various descriptors have been introduced to illustrate the progression of a fire [14]. These include the fire's area and its angle, determined by the minimal bounding box calculated using the rotating callipers method. Typically, the area of a fire covers hundreds of thousands of square meters, whereas the angle varies from 0 to 360 degrees. If these two measurements are employed to characterise a fire without any form of normalisation, minor differences in the area will dominate, overshadowing any significant variations in the angle. By scaling the data, analysts can facilitate the model's ability to learn the relationships between different features more effectively, leading to improved performance and predictive accuracy. Additionally, normalisation enhances the stability and convergence of optimization algorithms, thereby expediting the training process and reducing the risk of overfitting. Identifying the need for normalisation during EDA allows analysts to proactively address potential scaling issues and optimize the dataset for subsequent modelling stages.

3.2 Model Design

Model Design is a pivotal phase in the development of AI methodologies, where the foundation laid during EDA is transformed into actionable insights and solutions. This section delves into the intricate process of crafting robust and effective models tailored to address the objectives and challenges of the task at hand. Model Design encompasses a diverse array of techniques, algorithms, and methodologies, each meticulously selected and tailored to optimise performance, interpretability, and scalability. By leveraging insights gleaned from EDA, analysts embark on a journey of model formulation, iteration, and refinement, with the ultimate goal of delivering solutions that resonate with the complexities of real-world scenarios. The significance of Model Design lies not merely in developing predictive models, but in crafting solutions that encapsulate the nuances of the data landscape and align with the overarching objectives of the project.

The first step of model design is the formalisation of the problem statement. This foundational stage involves precisely defining the objectives, constraints, and variables of the problem at hand. By establishing a clear and structured problem statement, analysts provide a roadmap for the entire modelling process. This step guides the selection of appropriate algorithms, feature engineering techniques, and evaluation

metrics, ensuring that the resulting model is tailored to address the specific challenges and requirements of the task. Additionally, formalising the problem statement facilitates effective communication and collaboration among stakeholders, fostering alignment of expectations and objectives. Overall, this initial step sets the stage for the subsequent phases of model design, playing a pivotal role in shaping the direction and outcomes of the modelling process. Two primary features, among others, that should be considered and may impact the modelling process at this stage.

Objective Definition

Suppose the objective is to minimise the computational cost of a simulation while maintaining a certain level of accuracy. In this case, the problem definition influences the choice of model design by prioritising efficiency over complexity. The model may focus on identifying critical parameters or features that significantly impact the simulation's outcome and develop simplified approximations or surrogate models to replicate the behaviour of the original simulation at a fraction of the computational cost. Techniques such as dimensionality reduction [15], simplified physics-based models, or meta-modelling approaches like Gaussian Process Regression [16] can be employed to achieve this objective. By clearly defining the objective as minimising computational cost while preserving accuracy, the model design can prioritise strategies that strike a balance between efficiency and fidelity.

Constraint Consideration

If there are constraints on computational resources or time limitations, the problem definition will influence the design of the model by imposing restrictions on the complexity and computational requirements of the solution. For instance, if there are limitations on the available computing resources, the model design may need to focus on developing lightweight algorithms or parallelisation techniques to distribute the computational workload efficiently. Additionally, time constraints may necessitate the use of iterative optimisation algorithms that converge quickly or the adoption of approximation methods that provide near-optimal solutions within a limited timeframe. By incorporating these constraints into the problem definition, the model design can adapt to the practical realities of the simulation environment and deliver solutions that are feasible within the specified constraints.

Another significant choice in Model Design revolves around the representation and processing of input data, with different formats presenting distinct challenges and opportunities. Among these formats, images, categorical data, floating-point data, vector data, and graphs each require tailored approaches to model development. Moreover, integrating multimodal data, such as images and categorical data, adds complexity but can enhance the models' ability to analyse the data and yield more robust results.

Images, for instance, demand specialised convolutional neural networks (CNNs) capable of efficiently capturing spatial relationships and visual features. Categorical data necessitates techniques such as one-hot encoding or embedding layers to transform qualitative variables into a numerical format amenable to modelling. Floating-point data typically lends itself well to traditional machine learning algorithms like linear regression or decision trees, which operate effectively on continuous numerical data. Vector data, characterised by arrays of numerical values, often benefits from techniques such as dimensionality reduction or clustering to uncover underlying patterns. Notably, graphs present a unique challenge due to their complex, interconnected nature, requiring specialised graph neural networks (GNNs) or graph embedding techniques to effectively capture the relational information encoded within the graph structure. Efficient modelling of graphs is paramount, as they serve as powerful representations for diverse real-world phenomena such as social networks, biological systems, and infrastructure networks. By leveraging specialised techniques tailored to each data format, Model Design ensures that the resulting models are equipped to handle the intricacies of the input data, maximising their effectiveness and impact across various domains and applications.

Another key factor in Model Design is the integration of domain knowledge [17], which plays a pivotal role in shaping the methodology and approach employed. Especially in tasks where a profound understanding of the problem domain exists, leveraging domain expertise becomes imperative for ensuring the relevance and effectiveness of the models developed. This integration allows analysts to encode nuanced insights and constraints directly into the model, guiding its behaviour and decision-making process in alignment with real-world considerations. One of the primary ways domain knowledge manifests in Model Design is through the development of handcrafted rules and the extraction of domain-specific features. These tailored approaches enable the model to capture the intricacies and complexities inherent in the problem domain, facilitating more accurate and contextually relevant predictions. Furthermore, the depth and breadth of domain knowledge directly influence the extent to which it should be applied in the modelling procedure, with higher levels of expertise necessitating more sophisticated and nuanced approaches. By embracing domain knowledge as a foundational element of Model Design, analysts can develop models that not only exhibit superior performance but also resonate with the intricacies of the domain, ultimately leading to more impactful and actionable outcomes.

The integration of domain knowledge from each pilot into the model design is considered pivotal. Specifically, both defining the problem and incorporating the pilots' domain expertise into each task are deemed essential during the development of AI methodologies. Another critical aspect is ensuring proper analysis, inspection, control, and collaboration between the AI team and each pilot to craft tailored AI solutions. This underscores the importance of establishing clear communication channels from the project's outset. Hence, a shared document accessible to both the AI team and each

pilot was established. This document served as a platform for comprehensive communication, facilitating a thorough understanding of every facet of the problem, including specific constraints, proposed methodologies, relevant literature, and the experimental findings and results generated by the AI team.

In Model Design, the significance of properly modelling the input data cannot be overstated. While datasets may often contain features in standard formats such as floating-point tensors, there are instances where additional structural information is embedded within the data, presenting unique challenges and opportunities for modeling. For example, in the UNISTRA pilot, where the dataset pertains to building information, spatial structural information is inherent in the data. Rather than relying on classical algorithms, leveraging this spatial information by transforming the data into a graph format allows for the utilisation of graph algorithms, thus enabling a more effective solution to the problem at hand. A similar approach was observed in the MTG pilot, where simulation data was complemented with handcrafted features derived from domain knowledge. In such cases, the use of tailored approaches that accommodate the specific characteristics of the data becomes essential. This may entail the development of mapping functions to transform the dataset into a format conducive to the modelling algorithm, ensuring that all aspects of the information provided are effectively utilised for better problem modelling and solution generation.

3.3 Model Execution

Model Execution represents the crucial phase where developed AI models transition from theory to practice, poised to deliver tangible outcomes in real-world scenarios. In this section, some of the steps of model execution are analysed, delving into the intricate process of deploying, evaluating, and refining these models. The imperative of proper performance evaluation and interpretability is emphasised in driving meaningful insights and informed decision-making.

Training of Models is a critical phase in the Model Execution process, where the focus shifts to developing and optimising models using training data to learn intricate patterns and relationships within the dataset. This phase is pivotal in ensuring the effectiveness of the models in making accurate predictions or generating valuable insights. By iteratively exposing the models to training data, they learn to discern underlying patterns, correlations, and trends, thereby enhancing their predictive capabilities. During training, various algorithms and techniques are employed to optimise model parameters and minimise errors, ensuring that the resulting models are robust and reliable. Moreover, training allows for the exploration of different model architectures and hyperparameters, enabling analysts to fine-tune the models for optimal performance across diverse datasets and use cases. Ultimately, the training of models serves as the cornerstone of Model Execution, laying the groundwork for the deployment of effective and efficient AI solutions in real-world applications.

Additionally, performance evaluation is a critical aspect of model execution, involving the rigorous assessment of the models' performance against predefined metrics and benchmarks to gauge their accuracy, reliability, and efficiency. This evaluation process is pivotal in ensuring that the models effectively align with the objectives and requirements of the task at hand. Given that Model Design shapes the model's architecture and ultimate goal during training, it is imperative to evaluate not only the model's performance but also its alignment with the task itself. This entails assessing how well the model performs in accomplishing the desired task or solving the problem at hand. For instance, in the case of the Urban Buildings (UB) pilot, it is crucial to evaluate not only the performance of the graph algorithms but also their effectiveness in addressing the specific objectives of the pilot project. By evaluating both the performance of the model and its alignment with the task, users and developers of the models can gain a comprehensive understanding of their capabilities and limitations, enabling informed decision-making and iterative refinement.

Furthermore, another aspect of model execution involves gaining information about the inner workings of the models after the evaluation procedure. This is usually referred to as the features of interpretability and explainability [18], which play a crucial role in ensuring transparency and trustworthiness in the decision-making process of AI models, particularly in high-risk tasks where the model's decision has a significant impact. These concepts provide insights into the model's decision-making process and rationale, allowing users and developers to understand how and why certain predictions or decisions are made. By offering transparency into the inner workings of the model, interpretability, and explainability not only foster trust among stakeholders but also enable the identification and mitigation of biases, errors, and erroneous decision-making processes [19], [20], [21]. In high-risk tasks such as those undertaken in the Meteogrid project, where the consequences of model inaccuracies can be severe, interpretability and explainability become even more critical. They empower developers and users to scrutinise model outputs, identify potential weaknesses or errors, and take corrective actions to ensure the reliability and safety of the AI-driven systems deployed in such contexts.

3.4 AI Frameworks

Python [12] serves as the primary programming language for the entirety of the experiments, including not only EDA but also model design and execution. This choice is made due to Python's readability, flexibility, and extensive ecosystem of libraries. Python's intuitive syntax facilitates rapid development and prototyping of scripts, streamlining various processes. Additionally, its extensive library support allows seamless integration with specialised tools for data analysis, model training, and evaluation, enhancing the efficiency and effectiveness of the experimental workflows.

NumPy [22] provides essential functionalities for numerical computing, including powerful array objects and a collection of mathematical functions. The array data structure offered by NumPy enables efficient storage and manipulation of large datasets, enhancing computational performance. Its comprehensive set of mathematical functions simplifies complex numerical operations, such as statistical calculations and linear algebra, essential for thorough data analysis during EDA.

Pandas [23] is a fundamental library for data manipulation and analysis, offering high-level data structures and intuitive tools for handling structured data. The DataFrame object provided by Pandas facilitates the exploration and manipulation of tabular data, enabling seamless data cleaning, transformation, and summarisation. Its powerful indexing capabilities streamline data selection and manipulation tasks, while built-in functionalities for handling missing data enhance data preprocessing efficiency, which is crucial for robust EDA. This is particularly beneficial for every pilot, but it is crucial for MTG pilots where the use of NumPy is essential. The ability to quickly calculate fire shape descriptors using NumPy's efficient functions gives it an advantage over other libraries that lack such a comprehensive and optimised suite of methods.

Matplotlib [24] is a versatile plotting library that enables the creation of a wide range of static, interactive, and publication-quality visualisations. Matplotlib's extensive gallery of plot types and customisation options empowers analysts to generate insightful visualisations, revealing patterns, trends, and relationships within the data. Its integration with Pandas facilitates seamless plotting of DataFrame objects, enhancing interpretability and aiding in communication of findings during the EDA process. Furthermore, Matplotlib's compatibility with various output formats enables easy incorporation of visualisations into reports, presentations, and interactive dashboards, enhancing the impact of EDA results. The Matplotlib library enhances the ability to utilize more advanced visualization techniques, enabling a thorough understanding of the data and results. It also allows for the creation of GIFs (MTG pilot) that include both the input and the features. This facilitates both the pilot leader and the AI team in inspecting the outcomes of the methodology to identify any potential flaws and errors.

PyTorch [25] is a widely-used deep learning framework that provides tensors and dynamic neural networks in Python with strong GPU acceleration. In the context of AI projects, PyTorch serves as the backbone for both model design and execution. Its flexibility and ease of use make it ideal for building and training complex neural network architectures tailored for specific tasks. PyTorch enables researchers to experiment with various network architectures, loss functions, and optimization techniques to achieve the desired performance.

PyTorch Geometric (PyG) [26] is an extension library for PyTorch designed specifically for handling graph-structured data and implementing graph neural

networks (GNNs). In the context of AI projects, PyTorch Geometric allows researchers to model and analyse data represented as graphs. This could include social networks, citation networks, or knowledge graphs. By leveraging PyTorch Geometric, researchers can develop GNN-based models to capture complex relationships and dependencies within the data, enhancing the quality of AI-generated outputs.

NetworkX [27] is a Python library for creating, analysing, and visualising complex networks. It provides tools for the study of the structure and dynamics of networks, making it valuable for modelling and analysing various types of network data. In the context of AI projects, NetworkX can be used to construct and analyse networks representing relationships between entities, such as social networks, communication networks, or biological networks. By incorporating NetworkX into the workflow, researchers can gain insights into the underlying structure of the data and leverage this information to design more effective AI models.

Hugging Face Transformers [28] is a popular library for natural language processing (NLP) tasks, providing access to pre-trained models and a high-level interface for working with transformer-based architectures. In the context of AI projects, Hugging Face Transformers enables researchers to leverage state-of-the-art language models for tasks such as text generation, sentiment analysis, or language translation. By incorporating Hugging Face Transformers into the workflow, researchers can harness the power of transformer-based models to generate high-quality textual outputs and enhance the overall quality of AI-generated results. To the best of our knowledge, no other library offers as extensive a collection of models and algorithms for natural language processing, image and speech modelling, multimodal algorithms, and generative networks as Hugging Face Transformers.

4 HiDALGO2 HPDA and AI Integration per Pilot

4.1 Renewable Energy Sources (RES)

4.1.1 Pilot description

RES aims at providing different scenarios in the field of renewable energy sources, in particular assessing the amount of electrical energy produced by the wind farms, solar panels, or to estimate probability of damages which may happen to infrastructure (electrical overhead network, PV panels, etc.) due to extreme events (e.g. excessive wind speed and gusts).

As described in D5.3, RES is a multiscale application based on community models: WRF [29] and EULAG [30]. There is a one-way coupling from WRF to EULAG, where data from finished WRF simulations is used as input for initial and intermediate state of EULAG simulation. Both models are storing data in NetCDF4 [31].

EULAG volumetric output data (tape.custom.nc file)

The file, written in NetCDF4 format, consists of multiple physical parameters including all the saved time steps in a single file. This means that the multidimensional arrays in the file are, in fact, time series. The arrays contain values saved for each mesh node. The file can be read with the NetCDF4 python library by creating a Dataset object. The most important attributes of the object are:

- dimensions: they contain all the spatial and temporal dimensions to which the saved arrays' shape correspond, for example the number of nodes along each axis, like '`x`', '`y`' '`z`' or the number of timesteps ('`time`'),
- variables: here are saved the arrays. For 3D data, like three velocity components: '`u`', '`v`' and '`w`', the arrays are 4-dimensional (they include `x`, `y`, `z`, and time coordinates).

The order of dimensions in all the arrays is as follows: array[`time`, `z`, `y`, `x`] (for 4-dimensional ones, like '`u`' or '`v`'). If any of the dimensions is skipped, the order of the rest is unchanged (for example array[`z`, `y`, `x`] for '`imb`').

What is important, `x` and `y` values are expressed in meters and are the result of Universal Transverse Mercator projection (UTM) of Earth surface.

Variables which are the most important for the purpose of analyses done with RES are:

- '`x`': one-dimensional array of `x` coordinates along `x` axis expressed in meters,
- '`y`': one-dimensional array of `y` coordinates along `y` axis expressed in meters,
- '`z`': one-dimensional array of `z` coordinates along `z` axis expressed in meters,

- ‘time’: one-dimensional array of time coordinates expressed in minutes counted from the start of the simulation,
- ‘u’: 4-dimensional array with u component of velocity expressed in meters per second,
- ‘v’: 4-dimensional array with v component of velocity expressed in meters per second,
- ‘w’: 4-dimensional array with w component of velocity expressed in meters per second,
- ‘imb’: 3-dimensional array (x, y, z coordinates) of mesh nodes treated as obstacles (immersed boundary method); used just for post-processing of images; 0 – not obstacle, 1 – obstacle,
- ‘pres’: 4-dimensional array with pressure value expressed in Pascals,
- ‘temp’: 4-dimensional array with temperature value expressed in Kelvins,
- ‘prec’: 4-dimensional array with precipitation value (the sum of precipitation for each time interval, not accumulated) expressed in g/kg.

WRF volumetric output data (wrfout* files)

Similarly to EULAG, WRF also writes volumetric data to NetCDF4 files. However, the main difference is that the data for each timestep is written as a separate file, so the arrays inside a single file are no longer time series. The same format is used in the Wildfires pilot; thus, a detailed description is omitted here.

4.1.2 HPDA application

RES is run daily to produce damage predictions on the electrical overhead network in one of the largest Polish cities. Forecasts on the amount of energy produced will be run similarly. In either case, the data produced does not require HPDA techniques. These will be required once ensemble runs are conducted. The RES pilot plans to run ensembles with different sets of parameters – different mesoscale weather prediction models, different parametrisation, different input parameters, etc. Ensembles are now used in RES damages scenario to check how wind speed and direction affect the probability of the damages to the overhead electrical network. For efficient parameter value generation and orchestration of ensembles runs within the HPC environment, the MUQSA toolkit is used (described in Deliverable D2.1). While it provides basic data analytics capabilities, such as calculating the average value of the damages probability, a proper HPDA solution is required to analyse dozens and hundreds of 3D and 4D data in the most efficient way to provide, e.g. mean results or probability distribution.

4.1.3 AI application

RES plans for AI usage concern i) speeding up the simulation by replacing some parts with surrogate model (based on AI), and ii) providing new functionality (correlation between weather forecasts and amount of produced electrical energy).

Surrogate modelling

AI can be used to substitute some parts of the simulations, so that rather than solving some equations, an approximate solution can be used with a given accuracy. Figure 4-1 depicts two different approaches which may be used.

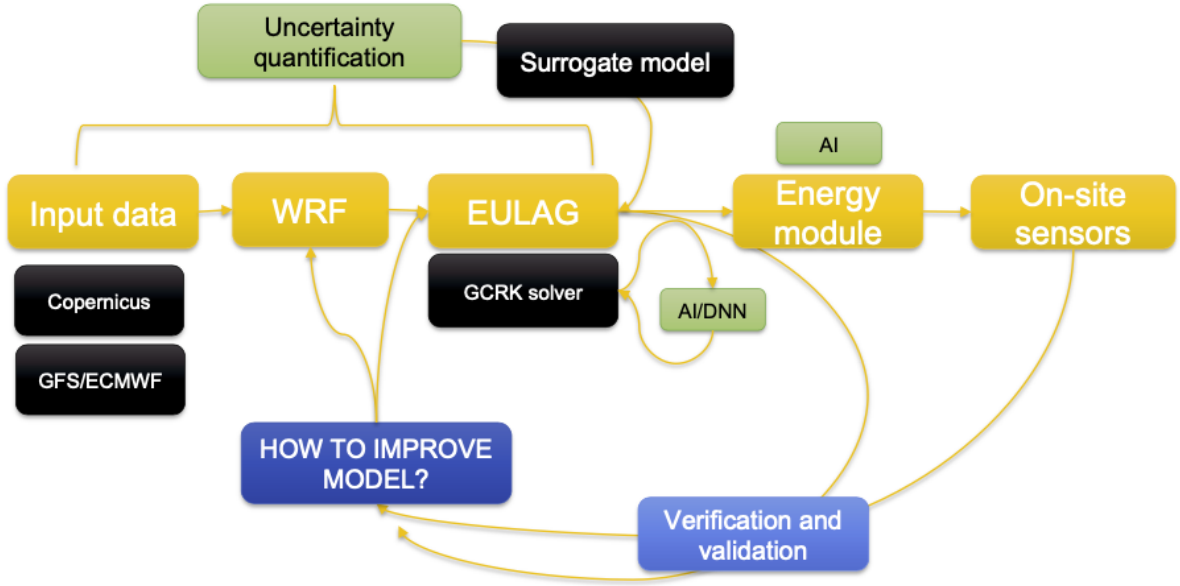

Figure 4-1 RES components

The first approach is to replace the model (or part of it) with a surrogate model. It is planned to use EasySurrogate [32], which facilitates the creation of such models. Depending on the results, another approach may be used (AI/DNN in the figure). The goal would be to use DNN techniques to support the iterative solver - some of the iterations are computed by solving the equations. At the same time, some are replaced by the DNN, until the desired level of accuracy is reached.

Energy module

Applying AI is one of possible techniques to find a correlation between weather forecast and amount of electrical energy produced, see *Energy module* in Figure 4-1. AI can be used in a form of predictive model based on the analysis of sequential data, i.e. so-called time series, to enable more effective forecasting in the field of demand-supply. Such information would improve the way the network is managed by energy system operators, improve network security and enable the transformation of the network to a flexible model, where it would be possible to introduce new prices for energy based on

the knowledge of what is happening in the network. Both wind and photovoltaics will require separate models.

The input data (weather forecast) is already available and produced on a daily basis, though access to real data for verification is required. The process of obtaining this data is ongoing.

The output data is historical data regarding energy generation. For initial work, synthetic data will be used. As of real data, PSNC is finalising the installation of a large PV system (almost 1MWp) and will start collecting and storing the data for further processing. For wind farms, the data is planned to be obtained through cooperation with one of the largest DSOs in Poland.

The described AI applications have not been implemented yet, as initial examples of the provided data were only available within the month of the deliverable's submission. Performing EDA is the first next step to be performed in order to facilitate the process of realizing the proposed pipelines.

4.2 Urban Air Project (UAP)

The HiDALGO2 Urban Air Project (UAP) application provides computational solutions for modelling air pollution concentrations and wind-related indicators for urban planning at very high resolution (e.g. 1-meter spatial resolution at street level). For a detailed description of UAP see the HiDALGO2 D5.3 document. Due to the large scale of the problem, high-performance simulations and evaluations are requested, thus we need HPC for simulations and HPDA methods for the evaluation of the simulation results.

Uncertainty quantification, data assimilation of observation and/or external simulations data, and real-time processing for emergencies (e.g. when the pollutant is a poisonous gas emitted from accidents or intentional harm and urgent response is needed) request a significantly faster approximate simulation model for the airflow and the pollutant dispersion than the simulation with HPC maybe on the price of reducing some accuracy but retaining the most important physical properties. The original computational model is often called the full order model (FOM) while the faster simulation is called the reduced order model (ROM).

In this document, we introduce the UAP pilot from the perspectives of creating ROM methods by using AI methods. Fast AI-emulators for the HPC simulations and HPDA methods for evaluations of the computational results, either from the HPC simulations or from the AI emulators. Also, we shall give the specifications of the AI emulators.

4.2.1 Pilot Description

The core of the current UAP simulation consists of computing the urban airflow. The other features and user-requested target parameters like air pollution concentration or wind-comfort parameters are computed from the urban airflow. In this document we focus on the urban airflow computations.

For the airflow computations the unsteady Navier-Stokes equations are solved. Several solvers are used, namely the OpenFOAM-based pimleFOAM, the RedSIM and the xyst. Each solver requests the subdivision of the urban air domain into small subdomains called cells, e.g. tetrahedral with edge size of 1-2 metres at urban ground level and coarser elsewhere. In total, the airflow domain is subdivided into 1 million to 100 million cells. For each cell all necessary physical variables (e.g. pressure, velocity; the concrete list of relevant physical variables depends on the applied physical model and its approximating numerical method) are computed in each time-step, sometimes the variables assigned to the entire cell, or the edge of the cells, or the vertices.

Thus, for each time step, we compute a vector of *N* physics-related variables by FOM, called state-variables in internal formats. The vectorization of physical scalars is done in a fixed way, e.g. in the order of the cell numbers with a given order of the physical variables. Under a certain strategy, we sample *M* state-variables from the FOM simulations and store them in the so-called snapshot matrix *X*. The physical time points belonging to the sampled variables are stored in the vector *time*, in the same order as the state-variables are stored in *X*. The concrete set of input parameters for the FOM are saved in a configuration file and some parameter files; the input parameters in the configuration file do not change in the construction of the ROM while the values in the parameter files are subject to change. For example,

- in the configuration file *conf* we fix the city geometry, the city mesh containing the cell information, parameters of the applied numerical scheme under the FOM, and the weather data that we do not change,
- in the parameter file *parBC* we store the boundary conditions for the FOM that are subject to change for the training of the AI-emulator, typically the wind boundary conditions. The wind boundary conditions are stored in a CSV-file: wind velocity coordinates at reference boundary points are saved.
- In the parameter file *parIC* the state vector belonging to the initial time is stored. In the header of the file, the physical time to which the initial state vector belongs is stored.
- Optionally, *parOBS* may store observational values belonging to the FOM if any. In the current version of UAP we do not use this

When saving the results of more than one FOM simulations, the number of simulations is saved to each file, thus *conf, X_1, time_1, parBC_1, parIC_1, X_2, …* are used.

4.2.2 HPDA application

According to EU directives, the pollution load in a city is quantified by yearly average values of hourly measurement data. This assessment is also possible using modelling and simulation. Long-term pollution spread simulation data, however, is extremely large. Even for moderate-size (a few million cells) meshes, the data volume is about

100GB for one month and 1 TB for one year if sampled per hour. While on-the-fly post-processing makes the most sense, as analysis at a more refined time scale would make sense and the sampling method would not require that kind of storage space, the need arises for analysing the data post-simulation.

4.2.2.1 Methods of HPDA analysis

The post-simulation analysis will encompass various methods to measure and assess data across specified locations and dimensions. Initially, point sampling will be conducted at predefined locations, such as monitoring stations. This will be complemented by surface sampling, which will be executed on both flat and uneven surfaces within designated areas, for instance, at an inlet or a specified height above the ground. Additionally, volumetric sampling will measure the total amount of pollution within the city, and line sampling will quantify pollution along pedestrian pathways.

The results of these analyses will be presented primarily in the form of time series data. This will include time series for specific values, such as pollutant concentration behaviour at monitoring stations, and for weighted values, such as the outflux of pollutants from city areas. Moreover, the data will be analysed to determine averages, minimums, maximums, and spreads, such as the daily average concentration of pollutants at two meters above ground level. Further analysis will involve calculating moving averages, as well as ongoing minimum and maximum values.

Regarding the methodology of sampling, the analysis will employ various approaches to ensure accuracy and comprehensiveness. This will include using the nearest neighbour or an average of nearest neighbours for point-specific data, averaging from points within a given radius for area-focused data, and range-dependent weighted averaging to take into account the distance effects on sampling points. Additionally, the analysis will calculate the minimum, maximum, and spread within each range to provide detailed statistical insights.

4.2.2.2 The HPDA analysis workflow

While it is generally good to have a brute force analysis tool, it is preferred to have a reasonable method to avoid reading all the data and getting an accurate enough result by looking at a fraction of the input. Moreover, it is reasonable to check if climate and weather conditions relate to the relevant time sample window. Additionally, the creation of a proper data analytic workflow would be beneficial:

1. Given yearly weather and climate data within a city, which time intervals are to be simulated to get results within a specific accuracy?
2. Given results with a specific accuracy, what new time intervals are to be simulated to get results within an improved accuracy?

3. Given a new city geometry and long-term climate data, what are the appropriate analytic steps for preparing analysis (1) and (2)?

4.2.2.3 Input format for HPDA

The current data is available in Ensight Gold format (see section 11.1 in [33]). Data is stored in binary per node. One month is stored in one case directory. Time is indexed in seconds, beginning at the start of the specific month. Velocity (**U**), pressure (**p**) and the pollutant NOx (**nox**) are stored. Time stamps are shifted with the number of iterations for calculating the steady-state initial condition (600).

4.2.2.4 Output format for HPDA

To fully leverage the capabilities of data analytics, it is advantageous to employ both value-based and plot-based results. For value-based results, presenting data in a dataframe format is preferred as it facilitates further analysis and interpretation. This is particularly useful for handling large datasets where efficient manipulation and analysis are required.

For plot-based results, which are essential for visualising patterns and trends, different types of plots are recommended depending on the analysis type. Time series analysis of derived data is best represented through appropriate plots that highlight changes over time. Heat maps or scatter plots are suitable for representing statistics of surface sampling, providing a visual representation of data variability and density across a defined space. Additionally, plot series are effective for detailed time series analysis of surface samples, allowing for a dynamic exploration of temporal changes in the data collected.

To fully benefit from data analytics, both value-based and plot-based results are preferred:

1. Dataframe format is preferred for value-based results to facilitate further analysis.
2. Appropriate plots are preferred for time series analysis of derived data.
3. Heat maps or scatterplots are preferred for statistics of surface sampling.
4. Plot series are preferred for time series of surface samples.

4.2.3 AI application

As introduced in Section 4.2.1 above, the UAP FOM simulations are described by a set of snapshot series *S_conf = [X_1, time_1, parBC_1, parIC_1, …]*.

The task for an AI application is to train such an emulator based on *S_conf* that enables to predict X from a suitable *time*, *parBC* and *parIC* parameters.

The construction of such emulator is the task of the next periods of the project. A set of concrete *conf* and *S_conf* has been generated. Also, a test-set generator has been implemented, too.

One special case of this general AI-problem has been implemented for RedSIM, namely the POD and the POD-DEIM methods. In POD, the Proper Orthogonal Decomposition method, a projection is defined based on S_conf, and then the FOM is applied in the projected space. Here the projection is defined by the truncated *U_r* matrix from the left-singular vectors belonging to the singular value decomposition of the *[X_1, X_2, …]* matrix as the projector. Alternatively to the AI-task above, we may train an emulator in the projected space. However, this is a special case of the above defined AI-task when we set up an autoencoder for the first layers of a neural network used to train the emulator.

We shall report the results of the AI-task in subsequent reports of the projects.

4.3 Urban Building (UB)

4.3.1 Pilot description

The Urban Buildings (UB) pilot's objective is, as stated in D5.3, to provide tools to accurately predict energy consumption, thermal comfort, and indoor air quality both at the building scale and at the urban scale.

Currently, the UB application can generate data concerning the heat flux and temperature of the building's surfaces in both Level of Detail (LOD)-0 and LOD-1 geometries. As solar masks are computed and used to calculate such outputs, they can also be returned by the simulation. Additionally, a discretisation of the city's geometry is available in conjunction with its geographic information in *msh* [34] and *GeoJSON* [35] file formats, respectively. The full documentation of the produced dataset is provided in Annexes: UB Simulation Output Data Description.

4.3.2 HPDA application

The simulation at the city scale needs to be analysed at various scales (district and city) and described with proper statistical data that are collected in an automatically generated comprehensive report. These data could also be visualised meaningfully using info-visualization [36]. Solar masking analysis generally involves assessing the shading created by elements of the urban landscape, such as buildings, trees, structures and other physical features.

An initial approach to this pilot was attempted using a sample dataset containing Solar Masks of buildings in Faches-Thumesnil, France. The HPDA development team conducted a round of exploratory analysis with a particular focus on the implementation

of the Density-Based Spatial Clustering of Applications with Noise (DBScan) clustering algorithm. The primary objective was to evaluate the algorithm's performance and its ability to reveal meaningful insights within the dataset.

Data transformation

The sample dataset includes solar masks for around 1000 buildings in Faches-Thumesnil, France. It comprises 5375 CSV files, each corresponding to a building face or roof of a building, with each file containing 72 rows and 10 columns. We flattened the contents of these files to a vector of 720 features. The final dataset, which was used in the methods described next, was obtained by piling the vectors for all building faces. It consists of 5375 rows x 720 features. In the final dataset, each row refers to a specific building face.

Design and Implementation

Our first thoughts regarding possible applications that could make use of the sample data were focused on trying to classify the various building faces. Our initial expectation was that the application of a classification algorithm would reveal some patterns or information hidden in the data. To test this hypothesis, we decided to utilise the DBSCAN [37] algorithm, which has the ability to identify clusters of data-points and assign each datapoint to either one of the clusters or a “noisy” set, which is treated as a group of outliers.

Density-Based clustering was applied using the scikit-learn library in a single-node Python environment. A series of experiments were performed that varied the parameters eps (minimum distance between two samples for neighbourhood consideration) and $minPts$ (minimum number of neighbours within eps radius). These experiments aimed to determine optimal clustering configurations as evidenced by the number of clusters, quantity of noise points (rows not belonging to any cluster), and Silhouette Coefficient (ranging from -1 to 1, where values close to 1 indicate well-defined clusters).

Findings and next steps

Upon careful analysis of the clustering experiments using the DBSCAN algorithm, it was deduced that the outcomes predominantly illuminated the primary orientations of buildings in the Faches-Thumesnil district. This revelation, while insightful, was not aligned with the practical objectives of the project. Recognizing the limited applicability of these findings, the team convened to reassess and refine the project's goals. This collaborative effort led to a redefined task with a clearer focus.

We identified the need for a more detailed dataset, which would be augmented with more features – including geographical information about the relative locations of

buildings in the urban environment. Our goal is to make it possible to identify the areas that benefit from maximum sunlight and those that are likely to be shaded at different times of the day and year. This identification can be proven fruitful for the following use cases.

1. A notable use case is identifying overexposed buildings and neighbourhoods to be used as a target for placing solar panels. Not only will this enable the orientation and tilt of solar panels to be optimized to maximise their exposure to the sun for improved energy production, but it will also enable precise town planning decisions.
2. On the same topic, identifying underexposed building groups can grant major insights into preventing health issues or other sorts of impact on the occupants. Additionally, targeting underexposed areas may be useful for optimising energy efficiency and predicting possible deterioration of the structures.
3. Finally, clustering buildings based on sun exposure and temperature can contribute to identifying possible unoptimised structures. It can also help in detecting zones that require more attention concerning indoor comfort, as well as energy optimisation. This use case can serve as a good indicator for urban planning strategies to identify the more suitable locations for development.

4.3.3 AI application

Description of the use case

Shading from surrounding buildings impacts the amount of solar energy that hits a target construction, which leads to modifications of temperature, humidity or incident light. The purpose of this use case is to quantify the variation, caused by a new building, on the solar intake and solar masks of the surrounding area.

Status of the collaboration/ Methods

The problem is tackled using supervised learning. Presently, a graph is associated with the city landscape: nodes correspond to buildings and edges correspond to the relation “building 1 shades building 2”. The objective of the training is to predict which edges are activated, hence which buildings are shaded by a given one.

The training of the neural network is performed over a custom dataset that UNISTRA has provided. This dataset contains one folder per building, where the solar masks that are stored are computed in the absence of this specific building. We stress that not all the masks of all the buildings are stored in each folder, only those whose values are modified by the absence/presence of the building under examination. Hence, it is necessary to provide also the “full” dataset where all the masks are computed in the presence of all the buildings.

NOTE: The code for the generation of the dataset one-building-out is to be found in the solar-shading repository, on branch 40-compute-shading-masks-when-a-building-is-absent. The generation of the "full" dataset can be obtained using the code in the master branch. However, the AI analysis should now use the new solar mask generation.

It is crucial to recognise that each building presents a unique solar mask for every surface it comprises. The central aim of our endeavour revolves around the determination of how solar masks evolve following the construction of a new building. While this scenario is relatively uncommon in bustling city centres, where new constructions are infrequent, it becomes a more prevalent occurrence in suburban landscapes. As of now, our analysis does not encompass the influence of factors such as vegetation or other minor parameters that might have the potential to impact the values of solar masks. These considerations may come into play as we delve deeper into understanding the dynamics of solar masks and their changes in response to evolving urban environments.

Given the dataset at hand, our attention is directed towards visualising the topological arrangement of the buildings. These structures are interconnected with neighbouring constructions via various relationships, such as their proximity and the shading they provide for one another. Therefore, representing them as a graph emerges as the most logical approach. However, the creation of this graph necessitates careful consideration of which relationships should be depicted as edges and which attributes of the buildings should serve as node features.

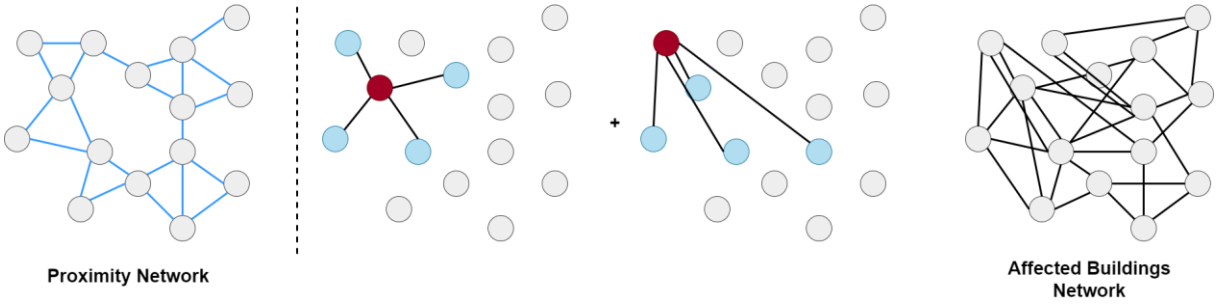

Figure 4-2 Affected Buildings Network creation. A simple Proximity Network versus an Affected Buildings Network. The process of iteratively adding edges to create the Affected Buildings Network is described on the right side of the figure.

By leveraging building affectedness as edge information, where buildings are connected only if the deletion of one affects the solar mask of the other, we can construct an affected buildings network (Figure 4-2). This network exhibits high irregularity in terms of edge distribution and may be either directed, with directions pointing to the surrounding affected buildings, or undirected.

An initial examination of this network reveals its density, with some edges existing between buildings that are not nearby. This unexpected observation stems from the

consideration of even minute changes in solar masks during dataset creation. To provide a more realistic depiction of the network, we propose establishing a threshold for solar mask difference before and after building deletion (Figure 4-3). Following experimentation, the threshold value was determined to be greater than or equal to *0.01* of the mean squared error of the pixel values of the original and modified solar masks. The depicted graph reflects the notion of expected proximity between co-affected buildings and is also not overly dense.

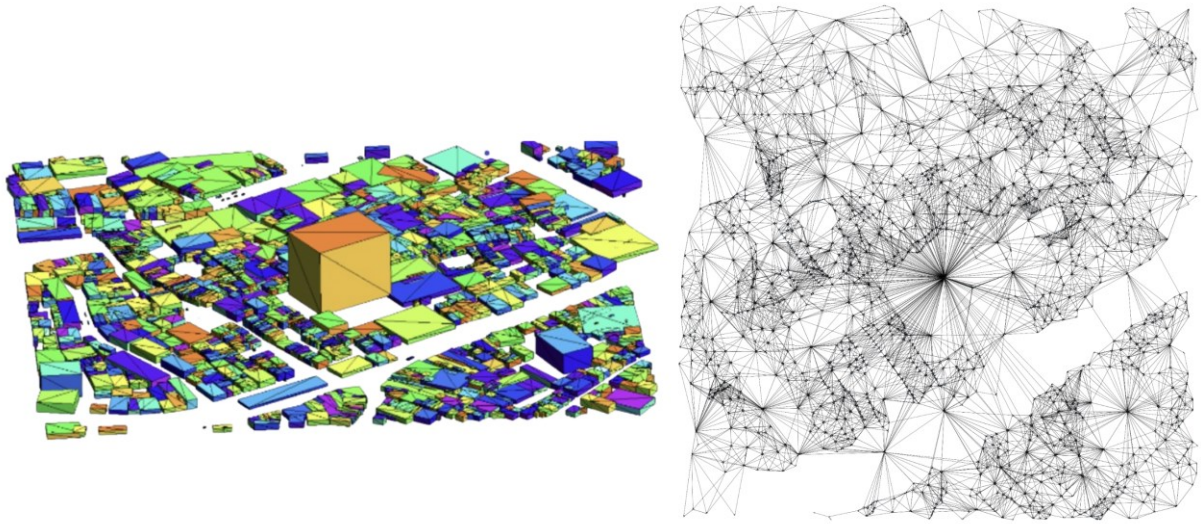

Figure 4-3 Depiction of the Affected Buildings Network with Threshold

After the creation of the affected buildings network, the graph representation is utilised to train a neural network for predicting edges between buildings, a task known as Link Prediction. This process involves training a Graph Neural Network (GNN) to learn node embeddings and predict the existence of edges between nodes. GNNs are the most commonplace choice for tasks on graphs, with link prediction being one of the tasks they tackle the most. GNNs are fairly lightweight networks; they are not based on computationally intensive techniques like attention-based transformers and are generally shallow yet effective.

The methodology employed can be outlined as follows:

- Data Preparation:
  - A portion of existing edges are removed, keeping all present nodes (buildings).
  - The remaining graph is fed through a Graph Neural Network (GNN) for training.
- Model Architecture:
  - A two-layer Graph Convolutional Network (GCN) [38] is employed to encode the graph's nodes through message passing [39].

  - The decoder part of the network, responsible for binary classification of the existence of an edge between two specific nodes, is treated as a hyperparameter.
- Experimental Settings:
  - Explored settings include:
  - Structure of the graph: directed vs undirected
  - Classifier type: Simple Dot Product vs MLP [40]
  - Node features: Building location vs building height
  - Threshold on solar mask difference

Table 4-1 presents the Area Under the Curve (AUC) of transductive Link Prediction on the affected buildings network for initial experiments, providing a reliable measure of the model's performance often used in relevant literature [41]. Results are provided for both the configuration that leverages a threshold for the solar masks, as well as not, for direct comparison. Higher ROC AUC scores denote better performance. Percentages highlighted in bold indicate best results per column.

Table 4-1 Initial test AUC scores on affected buildings prediction, on graph with edge threshold versus no threshold.

	Test AUC		Test AUC (with threshold)	
	Height Information	Topological Information	Height Information	Topological Information
Undirected Graph + Simple Classifier	**79.2**	**80.9**	**71.3**	71.6
Directed Graph + Simple Classifier	77.0	74.4	69.8	64.9
Undirected Graph + MLP Classifier	74.6	75.1	65.9	**78.4**
Directed Graph + MLP Classifier	71.6	74.9	55.6	77.5

We also provide training loss diagrams for all reported experiments (Figure 4-4). The decent of the loss function, which in all cases reaches a value very close to 0, combined with the elevated test set ROC AUC scores prove that the models were able to fit the data.

To explain and elaborate upon the provided results:

- The majority of configurations for GNN models yield high AUC scores (exceeding 70%), even in preliminary experiments that have not undergone full optimisation yet.
- Initial results indicate trends:
  - Building location as a feature is more informative.
  - Undirected graphs tend to yield better performance.
- Despite lower AUC results for thresholded cases, the predicted affected buildings correlate better with proximity, indicating promising results.

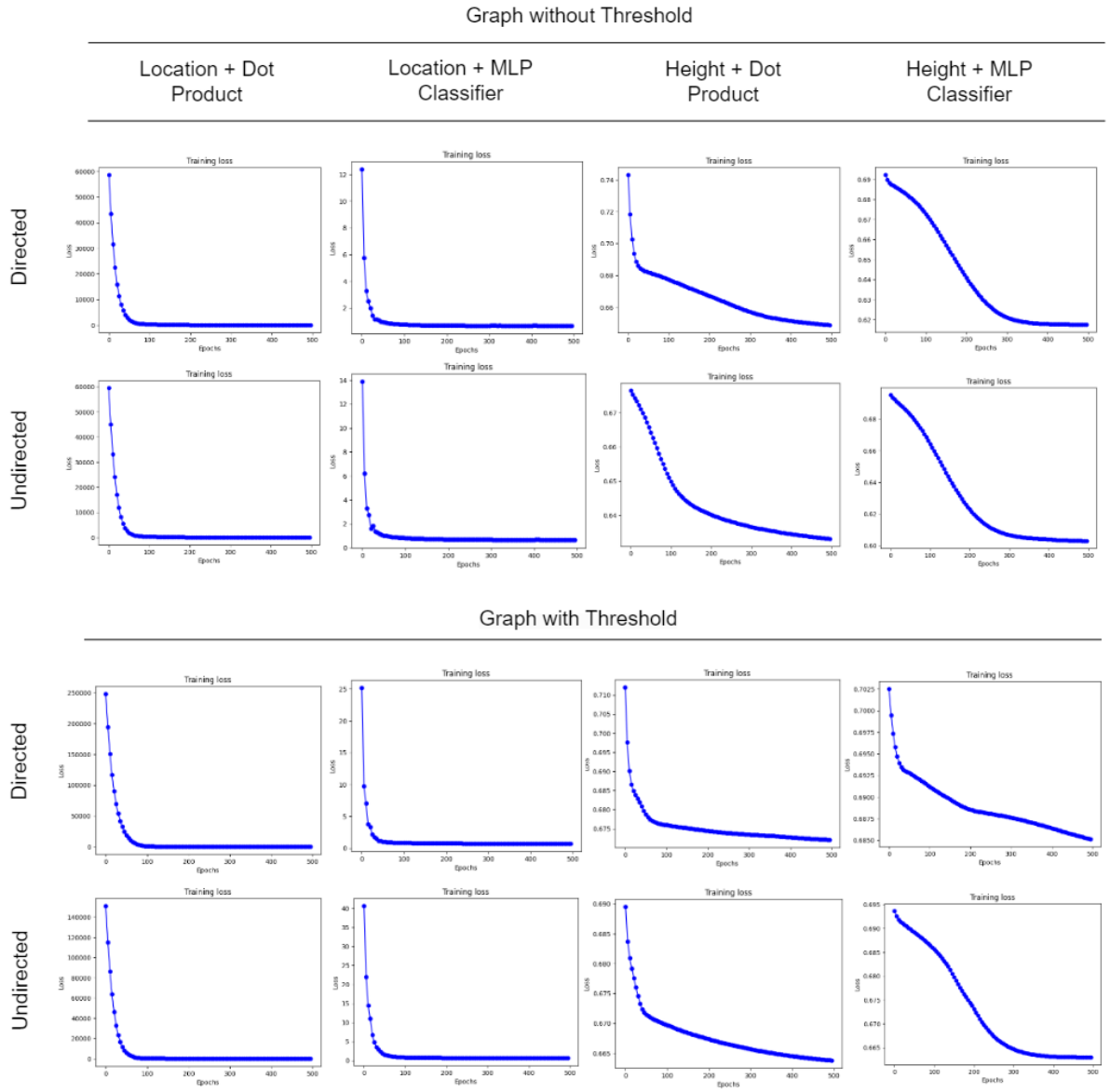

Figure 4-4 Training losses on affected buildings prediction, on graph with edge threshold versus no threshold.

Further optimizations on the GNNs will be conducted in the future. However, the initial findings support the hypothesis that building location features and undirected graphs

contribute to better performance in transductive Link Prediction tasks on affected building networks.

Analysing graphs and employing GNN models offer an additional advantage. During training, the generation of node embeddings to improve link prediction reflects the similarity between nodes/buildings. Consequently, our current efforts can be expanded to accommodate inductive scenarios in the future, such as predicting connections for buildings absent in the original graph. Moreover, comparing graph embeddings is pertinent for explainability techniques, a crucial aspect for contemporary AI applications where models are frequently opaque [19]. These efforts could serve as a blueprint for enhancing the explainability of our approach in the future.

4.4 Wildfires (WF)

4.4.1 Pilot description

The Hidalgo2 project aims to explore the use of HPC facilities to address environmental challenges resulting from climate change. Among these challenges, forest fires are a growing concern for administrations, emergency response agencies (fire-fighters, civil protection), and the population. Recent events in Greece (2023), the United States (2023), and Chile (2024) highlight the need to characterize the territory regarding its sensitivity to these highly destructive and deadly events.

The fire-atmosphere interaction is increasingly relevant in a type of fire that goes beyond fire-fighting capabilities and poses a risk to both responders and the population. This interaction is complex, dynamic, and occurs in three-dimensional spaces that are challenging to model with numerical solutions. Given the importance of the issue, a large number of researchers have focused on trying to characterise the dynamics of ignition, consolidation, and propagation of fire fronts and their interaction with the atmospheric boundary layer—essentially, the lower layers affected by these powerful disturbances. Indeed, more energetic forest fires inject energy and a significant amount of gasses, particles, and water vapour, altering the local circulation of the atmosphere. In the most dramatic cases, these disturbances create convection cells that induce winds dragging the fire itself, thus creating feedback phenomena that are challenging to control. It is also relevant to consider the production and dispersion of smoke, which often affects urban areas that are sometimes located far from the main front.

The numerical simulation of these phenomena is complex and computationally expensive. Computational Fluid Dynamics (CFD) models rely on three-dimensional grids of small cells for which the equations governing the movement are calculated. The good news is that these computational methods are highly parallelisable and scalable, making them perfect candidates for application on HPC architectures. One of these solutions, WRF-SFIRE, has been developed and extensively tested over the

last decade and adopted for the pilot study of forest fires in Hidalgo2. Installed and tested on various EuroHPC JU facilities (Vega, LUMI, MeluXina), MeteoGrid has conducted simulations of some of the most relevant fires in the central region of Spain, for which there is sufficient documentation.

Despite all these advantages, applying fire simulation for operational purposes on HPC facilities is a significant challenge: data capture, pre-processing, and final computation can take several hours, even with the most powerful facilities. Therefore, it is necessary to work with pre-computations of all possible scenarios in indexed ensemble sets, which can be accessed on demand in the operational phase in just a few seconds. This approach requires the precise characterisation of all possible meteorological scenarios in which forest fires can occur in an area, including wind speed and direction and the moisture conditions of forest fuels. It is also necessary to simulate fire spread from all possible points of origin (initial ignition points) in the study area. This involves, through the combination of factors, several tens or hundreds of thousands of simulations that must be stored efficiently and accessed practically instantaneously.

These simulation ensembles serve not only as massive datasets for training AI engines to find relationships between real and simulated fires but also enable High-Performance Data Analytics (HPDA) for studying landscape sensitivity to factors and processes governing forest fires. One of the classic HPDA approaches involves clustering all simulations obtained into ensemble sets and calculating statistics to obtain, for example, the Burn Probability (BP) at each point on the territory. In the HIDALGO2 project, similar analyses are proposed for the probability of smoke passage due to forest fires, namely the Smoke Probability (SP).

The fire simulation ensembles comprise sets referring to various changing factors, such as wind speed or direction (atmospheric configuration) or the position of the fire's origin on the territory. Typically, ensemble simulations are conducted by integrating a wrapper responsible for varying the conditions and launching the simulations to the calculation engine it includes. The results are written in separate files, and, as a complement, statistics are generated for data analysis.

In the pilot study of forest fires, ensemble sets of simulations have been developed to establish a database for subsequent use in AI inference engines through similarity and to conduct high-performance data analysis (HPDA) of landscape sensitivity to forest fires. For this purpose, a study area near Barcelona, Spain, has been selected, which has been analysed on previous occasions and for which there is a good set of baseline data (Annexes: WF Simulation Output Data Description).

4.4.2 HPDA application

A typical application of data analysis for characterising the sensitivity of the territory to forest fires is the so-called **Burn Probability** (BP), whose expression is:

$$BP = 100 \cdot \frac{NF}{NS}$$

Where:

BP Burn Probability, expressed in %
NF Number of times fire passes through a point in the territory
NS Total number of ensemble simulations

In HiDALGO2, it is suggested to use this BP index coupled with the presence of buildings, roads, and other potentially vulnerable infrastructures associated with the population in the landscape. This way, it is possible to calculate the probability of undesired impacts and refer it to a risk map.

Furthermore, it is suggested, in a novel manner, to apply the same concept to ground-level smoke intrusion resulting from all possible simulated forest fires in the ensemble, given a threshold value of the traced pollutants (especially suspended particulate matter PM10 and PM2.5) set according to health impact criteria for the population. The resulting value will again be combined with the distribution of buildings, road networks, and other population-related infrastructures, as well as the average population density in the study area. This could serve as an indicator of risk due to the probability of pollutant presence in populated areas. A sample of a possible expected output, visualised on a map, is displayed in Figure 4-5.

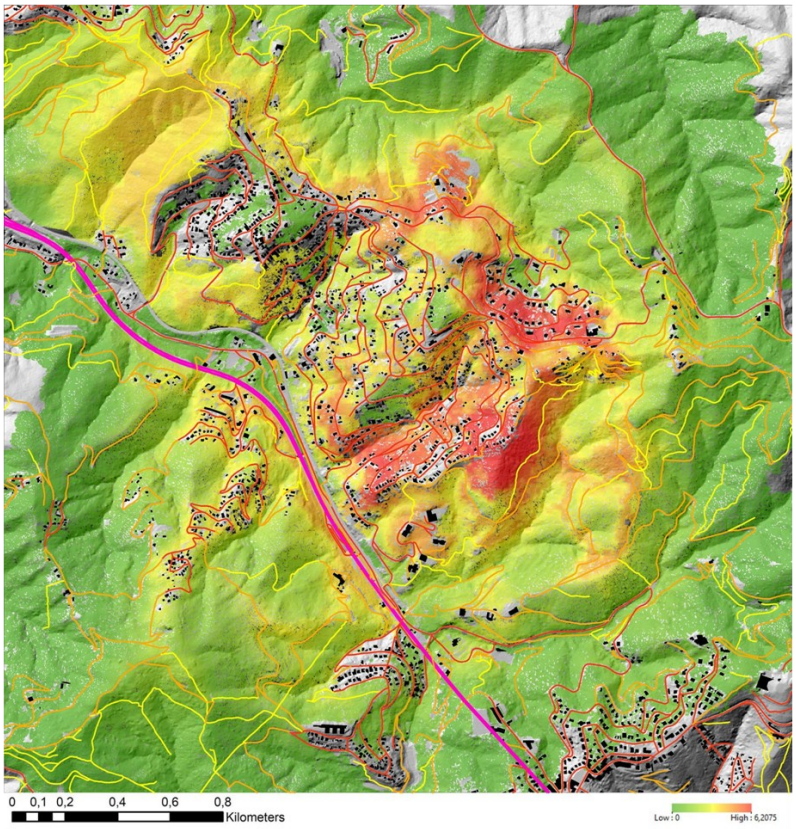

Figure 4-5 An example of a typical HPDA analysis using Burn Probability (%), resulting from the overlap of all simulations performed in the mentioned Rectoret and Les Planes study case. The areas in red indicate a higher likelihood of fire spread.

4.4.3 AI application

Performing a large number of simulations in advance allows, on one hand, to train an AI engine to find patterns and thus quickly characterize the fires that occur later in the real world. On the other hand, it is possible to establish a database of indexed simulations according to the factors that govern them to subsequently use them for operational purposes.

The strategy for using these simulations in the operational phase is conceptually straightforward: the progress of a real-world fire is detected, and its perimeter is captured through satellite or aerial photography. This perimeter, along with the point of origin, is used in an AI engine to, through a shape similarity analysis and based on actual meteorological conditions, search for and extract the pre-calculated simulations that most closely resemble the observed fire. This process requires a detailed description of the fire shape using shape indicators, which undergo a similarity analysis. If no reasonably similar simulation is found, the new situation is computed and added to the simulation database for subsequent use. As can be seen, it is

necessary to train an AI engine with thousands of simulations to extract patterns about the shapes that can then be easily identified in the case of real fires.

Specifically, the proposed procedure for the application of AI engines for the search and discovery of the closest simulations is as follows:

- A real fire is developing and the fire front position is captured at a certain time (i.e. using satellite-borne sensors MODIS, SUOMI etc.)
- The shape of the current fire front is extracted from the active (burning fires)
- The descriptors of the shape are calculated for the given time
- Other variables are also considered for the analysis: wind speed, wind direction and coordinates of the point of origin (if known)
- A search and discover algorithm is applied to a large database of simulations, in which the same shape descriptors and the other variables are the associated indexes. This applies similarity routines to extract the simulations that are closest to what has been observed at the given time.
- The extracted pre-calculated simulations, provide a frame for the projection of the expected fire behaviour, which is rendered instantly.
- If the similarity analysis fails to find a simulation, the system suggests to proceed with the simulation and include it in the database for future use.

Wildfire shape descriptors

To proceed with the similarity analysis, it is the proposed method for obtaining basic descriptors of fire spread shapes, namely:

- From the simulation 2D grid storing the fire access time, extract the desired contour corresponding at a certain time of simulation.
- Calculate the centre of gravity of the resulting shape
- Calculate the oriented minimum bounding box (use rotating callipers method)
- Extract major and minor axes (named as Length and Width)
- Extract orientation of major axis (it will be the general main propagation direction)
- Derive eccentricity (length-to-width ratio)
- Calculate the moment of inertia of the shape referred to as the major axis

The proposed shape descriptors used for similarity analysis are:

- Total area enclosed by the shape

- Total length of shape perimeter
- Eccentricity
- Orientation angle
- Moment of inertia

The initial stage in developing the AI similarity model involves conducting an Exploratory Data Analysis (EDA) to gain a comprehensive understanding of the dataset. To achieve this, we begin by extracting the relevant features for each time step of every fire. Specifically, Figure 4-7 and **Error! Reference source not found.** illustrate the evolution of these features for a single fire, with the objective of gaining insight into the ability of the handcrafted features to properly describe the evolution of a fire. The evolution of the fire, along with the evolution of the handcrafted features for a set of fires, is also exported in GIF format to a dedicated data repository. An instance of these GIF files, featuring the forest fire, along with the shape descriptors, is depicted in Figure 4-6.

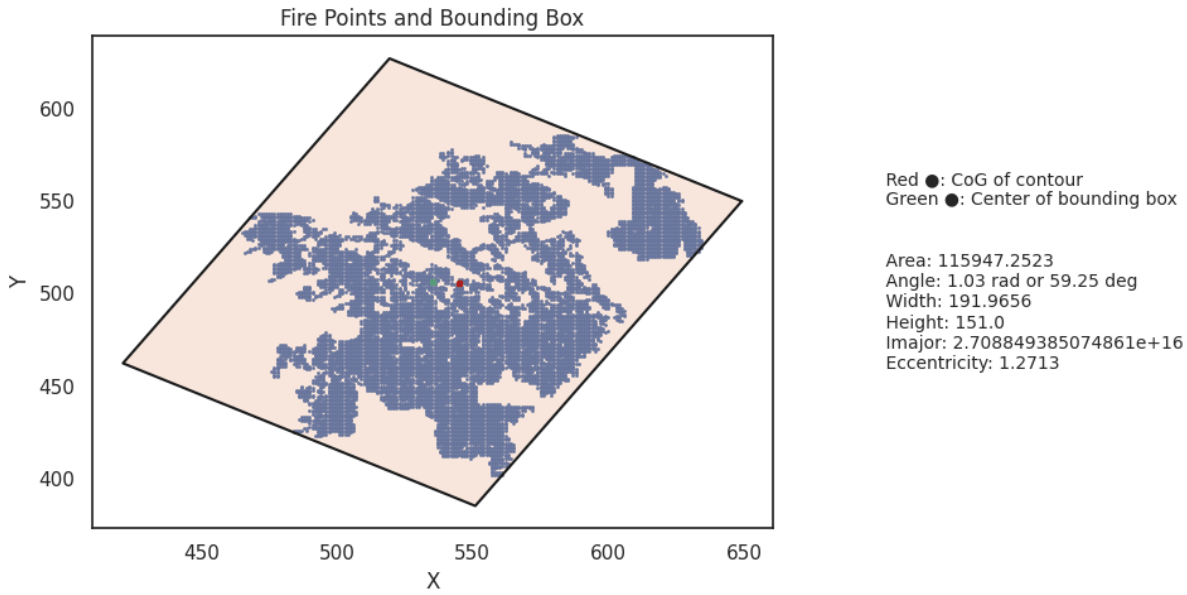

Figure 4-6 An example of forest fire spread shape descriptors extraction, as applied to the simulations ensemble of Rectoret area.

Studying Figure 4-6, we can make several observations. First, the centre of gravity of the contour, indicated by a red dot, differs from the centre of the bounding box, marked by a green dot. These two features represent distinct characteristics of the fire. Additionally, as mentioned in section 3.2, the definition domains of these features vary significantly. For example, the total area of the bounding box is approximately 115,000 square meters, while the angle is only 59 degrees. To prevent the similarity algorithm from overly focusing on the area – due to its large values – normalisation is crucial. In

Figure 4-7 and **Error! Reference source not found.**, the width and height of the bounding box are measured not in actual meters but in pixel counts. This approach was chosen to enhance our understanding of the differences between the features when they are counted in the image of the actual fire. All calculations will utilise the features of the actual fire.

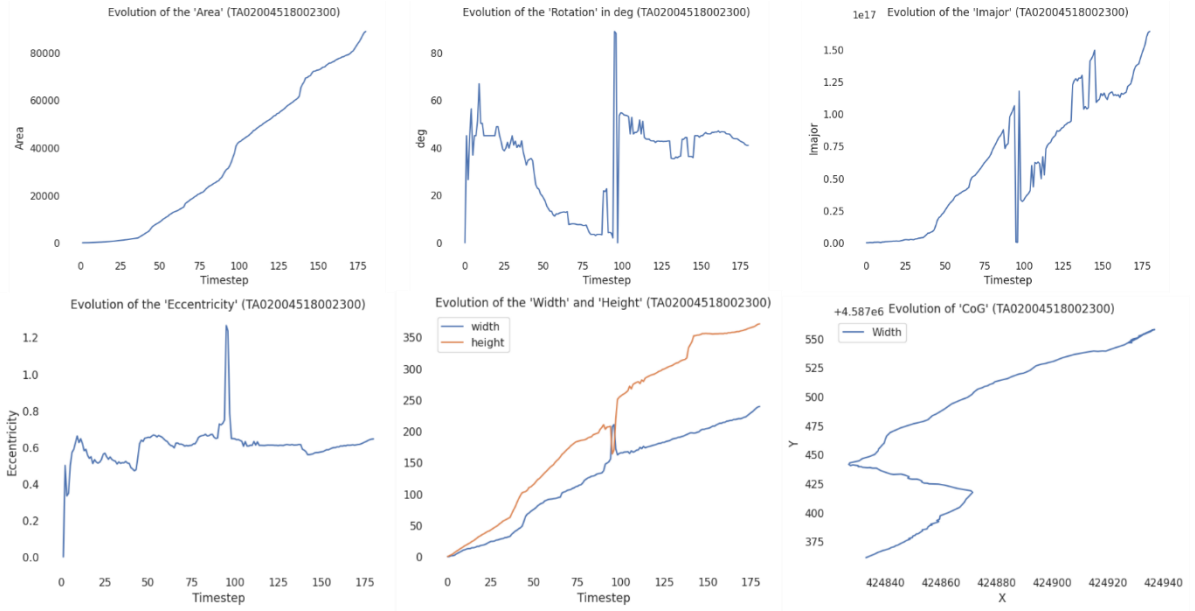

Figure 4-7 illustrates the evolution of the six features: “Area,” “Rotation,” “Imajor,” “Eccentricity,” “Width,” “Height,” and “Center of Gravity of the contour” for each timestep.

Figure 4-7 illustrates the development of features in a fire simulation. Initially, as expected, the “*Area*” increases consistently over time. However, this is not always the case. For instance, if the fire is temporarily contained, the area will decrease before increasing again. Additionally, the steepness of the line indicates the growth rate of the fire. Also, the behaviour of the “*Rotation*” in the fire simulation does not align with expectations. For example, at timestep 100, the noticeable spike in rotation is not due to an actual change in the fire itself, but rather because the “*Height*” and “*Width*” of the fire altered, as shown in Figure 4-7 titled "Evolution of 'Width' and *'Height'*." Despite this anomaly, the rotation generally exhibits a smooth transition over time. This consistency is also observed in the "I_{major}" and "*Eccentricity*" features of the simulation. Similarly, the “*Width*” and “*Height*” of the fire, like the “Area”, have shown a steady increase. However, the “*Height*” increases more rapidly than the “Width”, leading to swapping of values in these features around the 100th timestep. Lastly, the “*Center of Gravity*” proves to be a valuable feature, as it correlates with the overall direction of the fire. Initially, the fire progresses linearly in both **x** and **y** coordinates, but then it shifts direction primarily in the x-axis, repeating this pattern later on.

Error! Reference source not found. illustrates comparable metrics for an alternate fire simulation, demonstrating consistent behaviour within the fire's area, marked by a steady increase. Notably, around the 100th and 150th timesteps, there is a rapid expansion of the fire. This quick growth is clearly represented in the "*Area*" graph by the steepness of the line. Also, the parameters "*Rotation*", "I_{major}", and "*Eccentricity*" exhibit numerous spikes. These fluctuations are primarily attributed to changes in the "*Height*" and "*Width*" characteristics, which define specific features of the fire's behaviour. Additionally, by analysing the "*Center of Gravity*", the progression of the fire can be predicted, initially rising along the y-axis before descending.

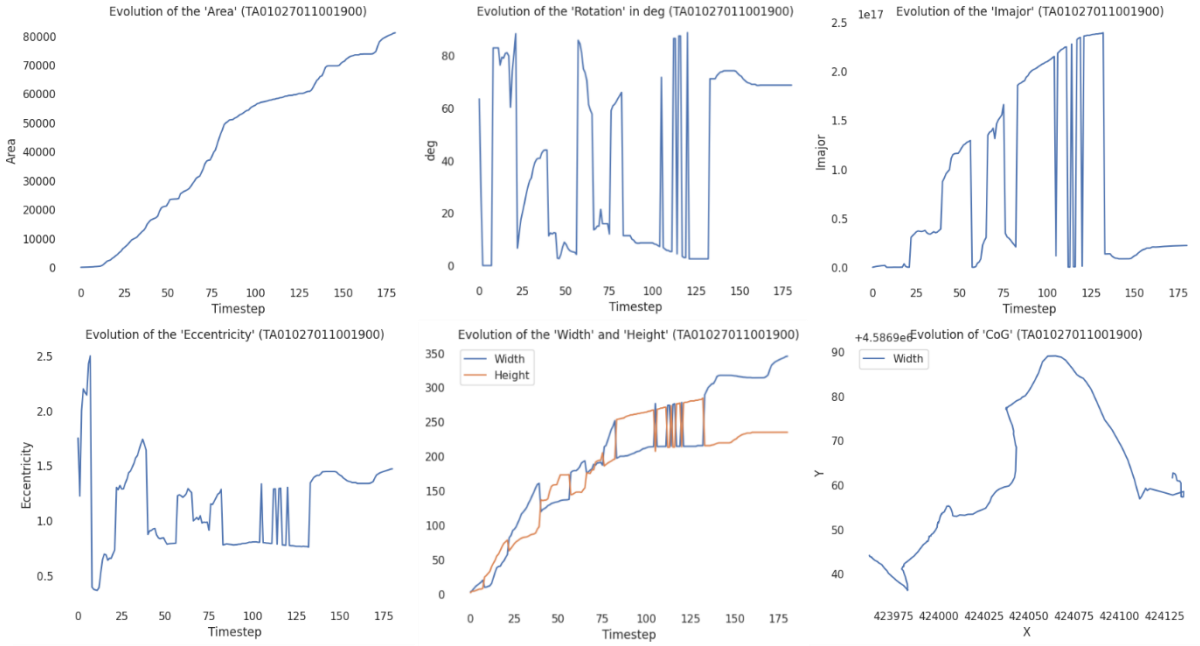

Figure 4-8 illustrates the evolution of the six features: "Area," "Rotation," "Imajor," "Eccentricity," "Width," "Height," and "Center of Gravity of the contour" for each timestep.

Figure 4-9 illustrates the evolution of the feature "Area" across 10 different fire simulations. By analysing the development of the areas affected by the ten fires, it is possible to observe variations in how these fires evolved. Some fires exhibited rapid growth, while others spread much more slowly. This comparison highlights the diverse behaviours of wildfires in different conditions.

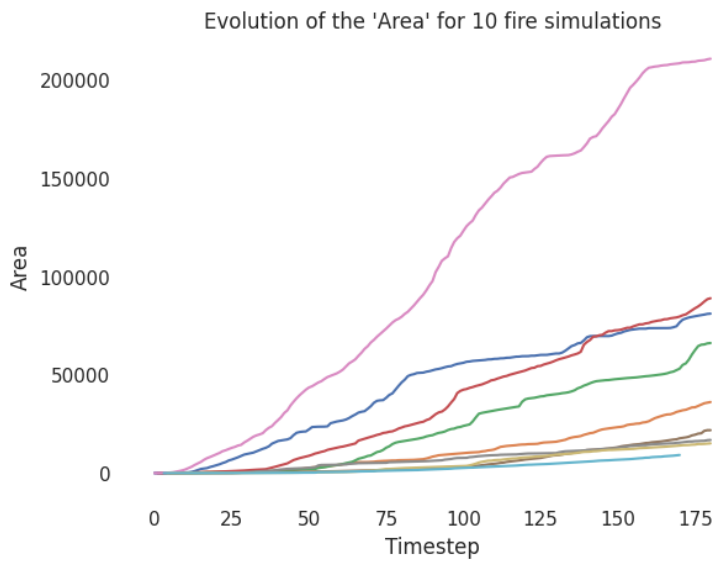

Figure 4-9 The evolution of the "Area" across ten different forest fires.

Observing the diverse trajectories of handcrafted shape descriptors among these forest fires, we notice that each fire follows a unique path. This uniqueness can be advantageous for identifying the fire's evolution based on these features. However, it also poses a challenge when attempting to find similar fire simulations for a new fire.

To delve deeper into the distinctiveness of each fire, we employed Principal Component Analysis (PCA) to reduce the fire's dimensionality for visualization. Figure 4-10 displays the outcomes of the PCA analysis. The left panel shows PCA features color-coded by the fire simulation filenames, whereas the right panel color-codes them according to the timestep from which each vector was extracted. Notably, the fire data consistently aligns along a linear trajectory, signifying close proximity among the features extracted at each timestep. Interestingly, the lower and central parts of the diagram show a distinct cluster of points, as indicated in the right figure where blue points represent the positions at timestep zero. Further analysis reveals that these points precisely correspond to the features extracted during the initial timestep of each fire simulation.

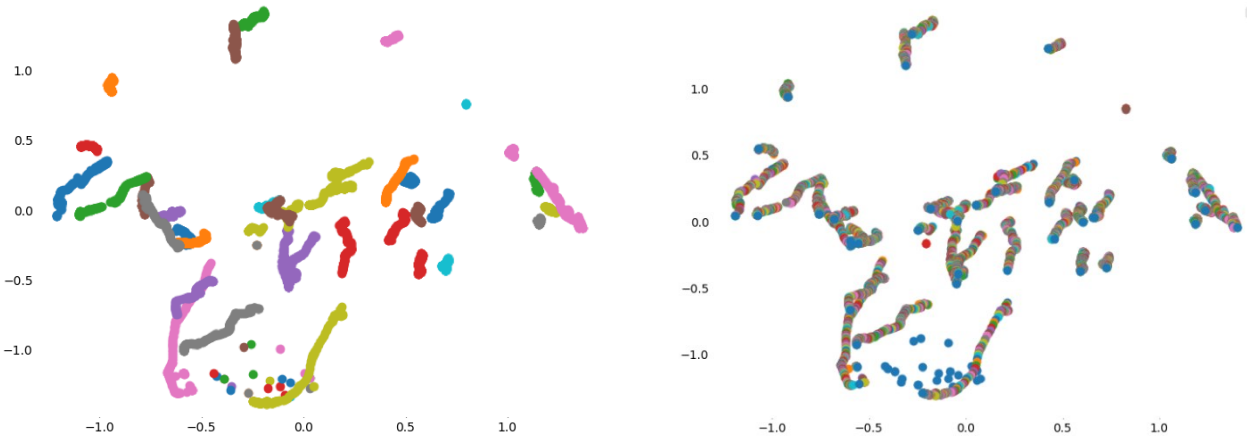

Figure 4-10 The results of PCA for each timestep of fire simulation.

Next Steps

Our initial analysis using handcrafted features provided valuable insights. While it confirmed the potential for fire identification based on these features, the observed uniqueness of each fire challenged finding similar simulations. PCA revealed strong correlations between features, suggesting redundancy and a distinct cluster for initial stages, highlighting their unique characteristics.

The next steps to enhance the similarity algorithm consist of:

1. **Trainable Similarity Measure**: Move beyond handcrafted features to trainable similarity measures that learn optimal feature weightings based on a pre-defined evaluation metric, such as the accuracy of predicting fire spread. This dynamic approach allows the model to prioritise features most relevant to specific fire behaviour comparisons. By constantly refining these weights, the model can achieve more accurate and context-specific fire simulation comparisons.
2. **Feature Extraction using classical Computer Vision techniques**: Use computer vision techniques to extract additional features capturing visual aspects like flame shape, smoke patterns, and fire intensity variations. These features can provide a richer picture of fire behaviour that goes beyond the data captured by the current set. By incorporating these additional features, the model can develop a more nuanced understanding of fire dynamics and how they evolve over time.
3. **Terrain Information Integration**: Fire spread is heavily influenced by the surrounding environment. Integrating information extracted from the terrain, such as slope, vegetation cover, and fuel load, could significantly enhance the model's ability to predict fire behaviour. Imagine a scenario where a fire starts

on a steep slope with dry brush – the model, considering this terrain data, could predict a faster and more aggressive fire spread compared to a scenario on flat terrain with moist vegetation. Considering these factors alongside the fire data itself, the model can build a more comprehensive picture of how a fire might evolve within a specific environment. This can lead to more accurate and informative fire simulations, ultimately allowing for better fire management strategies, resource allocation, and potentially even earlier intervention during real-world fire events.

4. **Integration of Multimodal Information**: The incorporation of multimodal data into the search algorithm represents a potential next step. This data, which combines handcrafted information with images of the fire and terrain, may be utilized. Through this approach, information concealed within the terrain and the fire, along with the handcrafted features, could be integrated into a single pipeline. This integration aims to extract higher-level features of the fire by leveraging all available information.
5. **Integration of Generative Models**: Another possible advancement involves the employment of generative models to optimize a hidden representation of a wildfire. For instance, by utilizing a generative model that mimics the fire's evolution, a hidden representation of the model for an individual fire can be exported and potentially used in the search algorithm. Additionally, the use of Large Vision Language Models (LVLMs) could be explored to generate a semantic description of a fire in natural language, which might also be incorporated into the search algorithm.

By implementing these proposed next steps, we aim to develop a more robust and informative AI similarity model for fire simulations. This enhanced model will leverage the power of trainable similarity measures, additional features extracted using computer vision techniques, and terrain data to provide highly accurate and insightful comparisons between fire simulations. This will ultimately contribute to improved fire prediction, prevention, and control efforts, leading to a safer future for communities and ecosystems.

4.5 Material Transport in Water

Environmental processes often involve complex interactions between fluids, solids, and gases. One example is sediment transport in rivers that is also impacted by pollutant transport and temperature effects.

4.5.1 Pilot description

For the multi-physics simulations, we use the waLBerla framework (https://walberla.net/) that is based on the Lattice Boltzmann Method for computational fluid dynamics and applies the MESA-PD module for particle simulations based on the discrete element method. As finite element based software we apply HyTeG (https://hyteg.pages.i10git.cs.fau.de/hyteg/index.html). Pollution is modelled by a concentration field and treated numerically by either an explicit or implicit solver. All physical models are fully coupled. In Hidalgo2 the main goals are to improve performance portability and scalability of the simulations and to investigate different possibilities for coupling the different models and software packages.

4.5.2 HPDA application

Usually direct numerical simulations (DNS) on a small volume fraction of the riverbed are performed. Large-scale runs may contain 10^11 or more particles. As input data e.g. the shapes, initial positions, and velocities of the particles are required. Output data is stored either as raw data or VTK files. It contains for example time-dependent fluid velocities and density, particle positions and velocities, and resolved concentrations.

There will be only a limited number of large-scale runs due to the huge memory and compute time requirements. We perform in-situ data analysis and data compression during the simulation. A proper HPDA solution is required to analyse the huge amount of data in the most efficient way to provide, e.g. mean results or probability distribution.

4.5.3 AI application

Similar to the RES pilot, we envision to investigate the accuracy of AI based surrogate models to approximate the DNS for larger simulations. This could reduce the required runtimes drastically. The idea is to predict future timesteps by AI models based e.g. on vision transformers trained on simulation data to be able skip or enlarge the timesteps during the simulation.

5 Adaptation to HiDALGO2 resources

The initial phase of the HiDALGO2 project involved conducting various preliminary experiments. At the time of preparing this deliverable, these experiments were performed using computational resources available to ICCS. This decision was primarily due to the convenience it offered during the initial stages of data exploration and algorithm development. Utilising these resources at this stage allowed for greater flexibility and a faster iteration process as the team navigated through the complexities of the data and refined the algorithms used in these experiments.

It is important to note that the data exploration algorithms employed in this phase are generally lightweight and do not require substantial computational power. Additionally, the GNNs used are also designed to be efficient.

Despite the initial use of more accessible computational resources, the project is poised to scale significantly. The next step for the AI/HPDA team is to transition the experiments to the HiDALGO2 project's dedicated resources. This shift will provide access to advanced supercomputing facilities, including the MeluXina and LUMI supercomputers. Leveraging these powerful systems will not only accelerate the experimentation process but also enhance the capability to handle more complex and computation-intensive algorithms.

The use of supercomputers like MeluXina and LUMI in the AI/HPDA part of the HiDALGO2 project is essential for several reasons. These high-performance computing environments offer vastly greater processing power and memory, which are crucial for training more sophisticated models and handling larger datasets effectively. Supercomputers facilitate significantly faster data processing speeds, enabling real-time analytics and the ability to conduct a higher volume of experiments concurrently. This capacity is particularly vital as the project advances to tackle more intricate challenges that demand extensive computational resources and more advanced AI and HPDA applications. Thus, the integration of supercomputing resources is not just an upgrade; it is a critical component in scaling the project's ambitions and achieving breakthroughs in complex data-driven problems.

6 Conclusions

During the initial phase of the HiDALGO2 project, the development team dedicated considerable efforts to accurately define various problems and pinpoint the challenges that needed to be addressed from an AI/High-Performance Data Analytics (HPDA) perspective. They specifically focused on identifying where and how AI/HPDA could support the pilot workflows most effectively. We additionally tried to document the requirements for the various use cases and determine the tools and frameworks upon which our implementations will depend, so that they are added to the available HiDALGO2 services. In terms of data handling, the primary emphasis was on analysing the key aspects of each dataset. The aim was to thoroughly explore these datasets in order to align them effectively with the established methodologies.

Several experiments were conducted, particularly with the UNISTRA and MTG pilots, which provided valuable insights into both the problems at hand and the associated data. Regarding UNISTRA, the initial experimental results offer several key insights into the efficacy of using graph neural networks (GNNs) for transductive link prediction tasks, particularly in networks of affected buildings. The findings suggest that building location is a significantly informative feature and that utilising undirected graphs tends to enhance performance. Interestingly, even though cases with thresholded results showed lower Area Under the Curve (AUC), the predicted affected buildings exhibited a stronger correlation with their proximity, pointing towards the effectiveness of the approach. The study also highlights the advantages of analysing graphs through GNNs, notably in generating node embeddings during training to improve link predictions. Additionally, the process of comparing graph embeddings is becoming increasingly relevant for developing explainability techniques in AI. These techniques are vital in an era where AI models often lack transparency. The current research and methodologies could thus pave the way for improving the explainability of AI applications in the predictive modelling of building networks. In the context of HPDA, our focus has been to identify classes of buildings or building faces depending on their exposure to sunlight. Our initial attempts failed to yield the expected results. However, they have helped us determine that a more detailed data set (i.e., including more features) is needed for this kind of analysis.

For the MTG pilot, the analysis of handcrafted shape descriptors in forest fire simulations revealed that each fire exhibits a unique trajectory, which is useful for tracking fire evolution but challenging for finding comparable simulations. Principal Component Analysis (PCA) was used to reduce the complexity of the fire data, which showed strong correlations among features and highlighted a distinct cluster corresponding to the initial stages of the fires. This insight underscores the potential and limitations of using such descriptors. The next steps include adopting trainable similarity measures to dynamically prioritise relevant features, integrating computer vision techniques to capture additional visual fire characteristics, and incorporating

terrain data to enhance predictive accuracy. These advancements aim to refine the AI model for more precise and informative fire simulation comparisons, thereby improving fire management and prevention strategies.

References

[1] S. Radack, “The system development life cycle (sdlc),” National Institute of Standards and Technology, 2009. Accessed: Apr. 16, 2024. [Online]. Available: https://csrc.nist.gov/csrc/media/publications/shared/documents/itl-bulletin/itlbul2009-04.pdf

[2] K. Shvachko, H. Kuang, S. Radia, and R. Chansler, “The hadoop distributed file system,” in *2010 IEEE 26th symposium on mass storage systems and technologies (MSST)*, Ieee, 2010, pp. 1–10. Accessed: Apr. 16, 2024. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/5496972/

[3] T. White, *Hadoop: The definitive guide*. O’Reilly Media, Inc., 2012. Accessed: Apr. 16, 2024. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=drbI_aro20oC&oi=fnd&pg=PR5&dq=+hadoop+&ots=t1xhtgg-f1&sig=Fd4wzHAsAqC-yUOravMSK1tphx8

[4] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, “Spark: Cluster computing with working sets.,” *HotCloud*, vol. 10, no. 10–10, p. 95, 2010.

[5] J. Dean and S. Ghemawat, “MapReduce: simplified data processing on large clusters,” *Commun. ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008, doi: 10.1145/1327452.1327492.

[6] F. Pezoa, J. L. Reutter, F. Suarez, M. Ugarte, and D. Vrgoč, “Foundations of JSON Schema,” in *Proceedings of the 25th International Conference on World Wide Web*, Montréal Québec Canada: International World Wide Web Conferences Steering Committee, Apr. 2016, pp. 263–273. doi: 10.1145/2872427.2883029.

[7] Y. Shafranovich, “Common format and MIME type for comma-separated values (CSV) files,” 2005. Accessed: Apr. 16, 2024. [Online]. Available: https://www.rfc-editor.org/rfc/rfc4180

[8] D. Vohra, “Apache Parquet,” in *Practical Hadoop Ecosystem*, Berkeley, CA: Apress, 2016, pp. 325–335. doi: 10.1007/978-1-4842-2199-0_8.

[9] M. Frampton, “Apache Mesos,” in *Complete Guide to Open Source Big Data Stack*, Berkeley, CA: Apress, 2018, pp. 97–137. doi: 10.1007/978-1-4842-2149-5_4.

[10] V. K. Vavilapalli *et al.*, “Apache Hadoop YARN: yet another resource negotiator,” in *Proceedings of the 4th annual Symposium on Cloud Computing*, Santa Clara California: ACM, Oct. 2013, pp. 1–16. doi: 10.1145/2523616.2523633.

[11] P. Carbone, A. Katsifodimos, S. Ewen, V. Markl, S. Haridi, and K. Tzoumas, “Apache flink: Stream and batch processing in a single engine,” *Bulletin of the IEEE Computer Society Technical Committee on Data Engineering*, vol. 36, no. 4, 2015.

[12] H. Karau and B. Lublinsky, *Scaling Python with Ray*. O’Reilly Media, Inc., 2022. Accessed: Apr. 16, 2024. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=I4meEAAAQBAJ&oi=fnd&pg=PP1&dq=ray+python&ots=KApUnmDl4C&sig=NbDA-A3WKVuRyFpoqyYjMgfdwJk

[13] A. T. Jebb, S. Parrigon, and S. E. Woo, “Exploratory data analysis as a foundation of inductive research,” *Human Resource Management Review*, vol. 27, no. 2, pp. 265–276, 2017.

[14] J. E. Hilton, C. Miller, and A. L. Sullivan, “A power series formulation for two-dimensional wildfire shapes,” *Int. J. Wildland Fire*, vol. 25, no. 9, pp. 970–979, Jul. 2016, doi: 10.1071/WF15191.

[15] R. Zebari, A. Abdulazeez, D. Zeebaree, D. Zebari, and J. Saeed, “A Comprehensive Review of Dimensionality Reduction Techniques for Feature Selection and Feature Extraction,” *Journal of Applied Science and Technology Trends*, vol. 1, no. 1, Art. no. 1, May 2020, doi: 10.38094/jastt1224.

[16] M. Ebden, “Gaussian Processes: A Quick Introduction.” arXiv, Aug. 29, 2015. doi: 10.48550/arXiv.1505.02965.

[17] D. Kerrigan, J. Hullman, and E. Bertini, “A survey of domain knowledge elicitation in applied machine learning,” *Multimodal Technologies and Interaction*, vol. 5, no. 12, p. 73, 2021.

[18] A. Erasmus, T. D. P. Brunet, and E. Fisher, “What is Interpretability?,” *Philos. Technol.*, vol. 34, no. 4, pp. 833–862, Dec. 2021, doi: 10.1007/s13347-020-00435-2.

[19] A. Dimitriou, M. Lymperaiou, G. Filandrianos, K. Thomas, and G. Stamou, “Structure Your Data: Towards Semantic Graph Counterfactuals.” arXiv, Mar. 11, 2024. Accessed: Apr. 16, 2024. [Online]. Available: http://arxiv.org/abs/2403.06514

[20] E. Dervakos, K. Thomas, G. Filandrianos, and G. Stamou, “Choose your Data Wisely: A Framework for Semantic Counterfactuals.” arXiv, May 28, 2023. Accessed: Apr. 16, 2024. [Online]. Available: http://arxiv.org/abs/2305.17667

[21] G. Filandrianos, K. Thomas, E. Dervakos, and G. Stamou, “Conceptual Edits as Counterfactual Explanations.,” in *AAAI Spring Symposium: MAKE*, 2022. Accessed: Apr. 16, 2024. [Online]. Available: https://www.ails.ece.ntua.gr/sites/default/files/publications/files/conceptual-edits-as-counterfactual-explanations.pdf

[22] S. Van Der Walt, S. C. Colbert, and G. Varoquaux, “The NumPy array: a structure for efficient numerical computation,” *Computing in science & engineering*, vol. 13, no. 2, pp. 22–30, 2011.

[23] W. McKinney, “pandas: a foundational Python library for data analysis and statistics,” *Python for high performance and scientific computing*, vol. 14, no. 9, pp. 1–9, 2011.

[24] F. Nelli, *Python data analytics: Data analysis and science using PANDAs, Matplotlib and the Python Programming Language*. Apress, 2015. Accessed: Apr. 16, 2024. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=f1F1CgAAQBAJ&oi=fnd&pg=PR3&dq=matplotlib+python&ots=2S9YBHyPPh&sig=GfPEW-Zrh4RfUIxQPlrsZY7MhdU

[25] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” *Advances in neural information processing systems*, vol. 32, 2019, Accessed: Apr. 16, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html

[26] M. Fey and J. E. Lenssen, “Fast Graph Representation Learning with PyTorch Geometric.” arXiv, Apr. 25, 2019. Accessed: Apr. 16, 2024. [Online]. Available: http://arxiv.org/abs/1903.02428

[27] A. Hagberg, P. Swart, and D. S Chult, “Exploring network structure, dynamics, and function using NetworkX,” Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008. Accessed: Apr. 16, 2024. [Online]. Available: https://www.osti.gov/biblio/960616

[28] T. Wolf *et al.*, “HuggingFace’s Transformers: State-of-the-art Natural Language Processing.” arXiv, Jul. 13, 2020. Accessed: Apr. 16, 2024. [Online]. Available: http://arxiv.org/abs/1910.03771

[29] W. C. Skamarock *et al.*, “A description of the advanced research WRF version 4,” *NCAR tech. note ncar/tn-556+ str*, vol. 145, 2019, Accessed: Apr. 16, 2024. [Online]. Available: http://pfigshare-u-files.s3.amazonaws.com/14057147/WRF_TechNote_Jan2019.pdf

[30] J. M. Prusa, P. K. Smolarkiewicz, and A. A. Wyszogrodzki, “EULAG, a computational model for multiscale flows,” *Computers & Fluids*, vol. 37, no. 9, pp. 1193–1207, 2008.

[31] R. Rew, E. Hartnett, and J. Caron, “NetCDF-4: Software implementing an enhanced data model for the geosciences,” in *22nd International Conference on Interactive Information Processing Systems for Meteorology, Oceanograph, and Hydrology*, 2006. Accessed: Apr. 16, 2024. [Online]. Available: https://ams.confex.com/ams/Annual2006/techprogram/paper_104931.htm

[32] W. Edeling, “wedeling/EasySurrogate.” Jan. 14, 2024. Accessed: Apr. 16, 2024. [Online]. Available: https://github.com/wedeling/EasySurrogate

[33] “EnSight User Manual for Version 7.6.” National Energy Research Scientific Computing Center (NERSC), 2003. Accessed: Apr. 16, 2024. [Online]. Available: https://dav.lbl.gov/archive/NERSC/Software/ensight/doc/Manuals/UserManual.pdf

[34] “Gmsh 4.12.2.” Accessed: Apr. 16, 2024. [Online]. Available: https://gmsh.info/doc/texinfo/gmsh.html

[35] H. Butler, M. Daly, A. Doyle, S. Gillies, T. Schaub, and S. Hagen, “The GeoJSON Format,” Internet Engineering Task Force, Request for Comments RFC 7946, Aug. 2016. doi: 10.17487/RFC7946.

[36] “Data and information visualization,” *Wikipedia*. Apr. 01, 2024. Accessed: Apr. 18, 2024. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Data_and_information_visualization&oldid=1216672204

[37] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, “A density-based algorithm for discovering clusters in large spatial databases with noise,” in *kdd*, 1996, pp. 226–231. Accessed: Apr. 18, 2024. [Online]. Available: https://cdn.aaai.org/KDD/1996/KDD96-037.pdf?source=post_page----------

[38] T. N. Kipf and M. Welling, “Semi-Supervised Classification with Graph Convolutional Networks.” arXiv, Feb. 22, 2017. Accessed: Apr. 16, 2024. [Online]. Available: http://arxiv.org/abs/1609.02907

[39] J. Gilmer, S. S. Schoenholz, P. F. Riley, O. Vinyals, and G. E. Dahl, “Message Passing Neural Networks,” in *Machine Learning Meets Quantum Physics*, vol. 968, K. T. Schütt, S. Chmiela, O. A. Von Lilienfeld, A. Tkatchenko, K. Tsuda, and K.-R. Müller, Eds., in Lecture Notes in Physics, vol. 968. , Cham: Springer International Publishing, 2020, pp. 199–214. doi: 10.1007/978-3-030-40245-7_10.

[40] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016. Accessed: Apr. 16, 2024. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=omivDQAAQBAJ&oi=fnd&pg=PR5&dq=Goodfellow,+Ian,+Yoshua+Bengio,+and+Aaron+Courville.+Deep+learning.+MIT+press,+2016.&ots=MON1cstARV&sig=f_Ef746ZEnmr0Smq7TUQOaUR-U8

[41] M. Zhang and Y. Chen, “Link prediction based on graph neural networks,” *Advances in neural information processing systems*, vol. 31, 2018, Accessed: May 10, 2024. [Online]. Available: https://proceedings.neurips.cc/paper/2018/hash/53f0d7c537d99b3824f0f99d62ea2428-Abstract.html

[42] J. Bennett, *OpenStreetMap*. Packt Publishing Ltd, 2010. Accessed: Sep. 28, 2023. [Online]. Available: https://books.google.com/books?hl=en&lr=&id=SZfqRcPXApoC&oi=fnd&pg=PT9&dq=openstreetmap&ots=G4_GlEoXE4&sig=sJTp3De523T3lWjvGy7evrxENqg

[43] S. Koranne, “Hierarchical Data Format 5 : HDF5,” in *Handbook of Open Source Tools*, Boston, MA: Springer US, 2011, pp. 191–200. doi: 10.1007/978-1-4419-7719-9_10.

Annexes

UB Simulation Output Data Description

The UB application produces, assuming that the simulation is named “strasbourg_M0” and is done using 24 partitions, a folder with the following structure.

- /gis/
  - *strasbourg_M0.json*
  - *strasbourg_M0_lod0.msh*
  - *strasbourg_M0_lod1.msh*
  - /partitioning/
    - *building_ids_partitioning.json*
    - *building_ids.h5*
    - *mesh.lod0.np24.h5*
    - *mesh.lod1.np24.h5*
    - *mesh.lod0.np24.json*
    - *mesh.lod1.np24.json*
- /instance/np24/
  - building_metadata.h5
  - /outputs/
  - /lod0/exports/
  - /lod1/exports/
- /simulators/
- setup.json

Each folder and file are described in detail below.

/gis/strasbourg_M0.json: The file contains the geometric information associated with the simulated set of buildings. It is stored in GeoJSON [35] format using MultiPolygon geometries. This information is requested from OpenStreetMap[2] [42]. The produced file schema is defined as follows.

```
{
 “building”: (array) [
          {
          “height” : (float) Height of the building, from the ground to the top of the roof
          “min_height” : (float) Elevation of the bottom part of the building relative to the
                                                    ground.
          “id” : (int) OpenStreetMaps identifier of the building
          “type” (str) : Type of building (e.g. “apartments”, “hospital”, “university”, …)
          “bounding_box”:(Geometry object) Coordinates of the building’s bounding box
          “outline” : (Geometry object) Coordinates of the building’s footprint.
          “parts” : (array) [
          {
```

2 https://wiki.openstreetmap.org/

```
                    "geometry" (Geometry object) Footprint of the building part,
                    "height" : (float) Height relative to the ground of the building part.
                    "minHeight": (float) Elevation of the bottom of the building part from the
                                                    ground.
                    "id": (int) OpenStreetMaps identifier of the building part
          }]
          },
{...}
],
"referenceCoordinates": (array(2)) [ (float) latitude, (float) longitude ]
}
```

The Geometry Object is described below:

```
"geometry" : {
          "coordinates" : (MultiPolygon) Coordinates in Geographic Coordinate System
                                                    (degrees)
          "coordinates_projection": (MultiPolygon) Projected cartesian coordinates using Mercator
                                                    projection (meters)
          "type" : (str) Type of the geometry ( in most cases "MultiPolygon" )
}
```

/gis/strasbourg_M0_lod0.msh, /gis/strasbourg_M0_lod1.msh: These two files contain the meshes for the LOD-0 and LOD-1 geometries in the Gmsh *MSH* format [34]. The mesh vertices X and Y coordinates are projected as in the GIS file. To obtain the coordinates in degrees, one must use the *referenceCoordinates* element of the object contained in strasbourg_M0.json.

/gis/partitioning/building_ids_partitioning.json: Describes how the building's output computation is partitioned.

/gis/partitioning/building_ids.h5: Contains one HDF5 [43] dataset named "*building_metadata".* The dataset consists of an ordered list of the building's OpenStreetMap identifiers. The order of the list is used to access the model outputs in */instance/outputs/*.

/gis/partitioning/mesh.lod0.np24.h5, /gis/partitioning/mesh.lod1.np24.h5: Both of these files contain the following datasets.

- "`elements`": connectivity of the elements
- "`elements_ghosts`": ghosts cells
- "`marked_subentities`": entities associated with markers
- "`point_coords`": point coordinates of the vertices of the elements
- "`point_ids`": identifier of the points
- "`stats`": some statistics of the partitioned mesh

/gis/partitioning/mesh.lod0.np24.json, /gis/partitioning/mesh.lod1.np24.json: The JSON file contains various metadata about the partitioned mesh among which the location of the hdf5 file.

/instance/np24/building_metadata.h5: The *building_metadata.h5* file contains a single dataset named "building_metadata", which has a list of length *3*N*, with *N* being the number of buildings. Each N^{th} value corresponds to the building's ID, and the value that follows each ID (N^{th}+1) corresponds to the model used. In this case, it signifies the number of storeys of the building, where 0 means that the building has one storey. Each N^{th}+2 value of the dataset indicates where to find the corresponding outputs on the any */building_*.h5* files. In order to find the number of outputs for each building, the model associated with the building should be explored in the */setup.json* file.

/instance/np24/outputs/: The folder contains as many files as timesteps of the simulation. Each file is named as *building_**t**.h5*, where **t** is the simulation step index and starts at 0. The files are in HDF5 format.

/instance/np24/outputs/building_t.h5: A timestep file contains the integrity of the output associated with a given timestamp, for all buildings. In order to identify the building and quantity to which a given value corresponds, one must look through the */instance/np24/building_metadata.h5* file and the *setup.json* file to find the building ID and output quantity name, respectively.

/instance/np24/lod0/exports/, /instance/np24/lod1/exports/: Both of these folders contain time series data useful for visualisation using ParaView. The data is in EnSight Case Gold format [33], and carries the following information:

- Meshing of the city
- Process identifiers (PID) for which buildings are associated with
- Solar shading coefficients associated with each mesh vertex
- Exterior temperature of the buildings surfaces
- Interior temperature of each building storey
- Ambient temperature

/simulators/: The */simulators/* folder holds the *.fmu* files of the models used in the simulation, as well as their descriptions in JSON format.

/setup.json: This file contains a JSON object with the models used in the simulation. Currently, there is one model per number of storeys of a building, with a maximum of ten. This means that the first element of the *building_models* array represents a one-storey building, and the 9th building represents a ten-storey building. The *output* element of each model holds a reference to a calculated quantity.

WF Simulation Output Data Description

Study Area and Initial Data

The selected study area for these initial exercises is a 3x3 km² zone in a district of the municipality of Barcelona. It corresponds to the urbanisations of Rectoret, Les Planes, Mas Guimbau, and Can Sauró, which are classified as Wildland-Urban Interface (WUI) areas due to their pattern of buildings and vegetation mix. This area has been chosen because sufficient information is available regarding initial data, fire statistics, population, and firefighting and civil protection resources. The boundaries of the study area are (ETRS89 UTM31N):

West	423030	South	4585060
East	426030	North	4588060

The input data for the simulations has been obtained from various sources, primarily from the Cartographic and Geologic Institute of Catalonia (ICGC) in its most recent versions. The information layers used are:

- **Digital Terrain Model**: Extracted from a LiDAR point cloud with a 2m resolution. Slope and orientation at each point have been calculated from this layer, with the same resolution.
- **Fuel Models**: These are idealisations of vegetation structures based on the widely used Anderson-BEHAVE catalogue worldwide. These models refer to surface vegetation, and the characterisation of tree canopy with other parameters is necessary. Since crown fires have not been simulated in this exercise, these parameters are not included.

Wind

According to the area's statistics, the most frequent and adverse winds for wildfires are those with a South and West component (SE, S, SW, and W). However, it has been preferred to perform an ensemble of simulations considering the entire compass rose, i.e., all 8 possible directions (N, NE, E, SE, S, SW, W, and NW) every 45º. The criterion for the description of wind direction angle is as follows: the angle of the wind vector with the north, in a clockwise direction (geographical criterion). The three wind speeds are 5, 10, and 20 km/h at 10m above the surface.

Wind Sheltering Factor (WSF) has been applied to reduce the wind at mid-flame, with some modifications to the original tables (BEHAVE), which better reflect the reality of the local vegetation effect, namely:

Wind Sheltering Factors												
M01	M02	M03	M04	M05	M06	M07	M08	M09	M10	M11	M12	M13
As proposed in BEHAVE:												
0.36	0.36	0.44	0.55	0.42	0.44	0.44	0.28	0.28	0.36	0.36	0.34	1.00
As used in the simulations:												
0.69	0.75	0.44	0.55	0.42	0.44	0.44	0.28	0.28	0.36	0.36	0.34	1.00

Fuel Moisture

The initial moisture values of each component of the forest fuel have been considered the same for all simulations, namely:

1HR = 4 1-hour dead fine fuel moisture, in %
10HR = 5 10-hour dead medium fuel moisture, in %
100HR = 6 100-hour dead large fuel moisture, in %
LH = 100 Live herbaceous fuel moisture, in %
LW = 110 Live woody fuel moisture, in %

These values have been obtained from statistics of average adverse meteorological conditions corresponding to the summer months and for the vegetation in the area.

Initial Ignition Points

Fire Ignition points have been systematically placed in a grid with 100m spacing in both the X and Y axes along the simulation area. However, a buffer zone of 500 m has been created to mitigate edge effects. Therefore, the number of initial ignition points I to be considered in the simulation ensemble is:

$$I = \left(1 + \frac{(426030 - 500) - (423030 + 500)}{100}\right) \cdot \left(1 + \frac{(4588060 - 500) - (4585060 + 500)}{100}\right) = 441$$

Therefore, the obtained number of simulations NS are:

$$NS = 441 \cdot 3 \cdot 8 = 10584$$

Simulations Ensemble Results

The simulations ensemble has been conducted for the mentioned Rectoret area, considering that some of the initial ignition points have fallen on non-combustible points and, therefore, have not undergone simulation. This aspect can be corrected in future exercises by randomly generating positions for the initial ignition points or as proposed in other studies, linking the origins of fires to infrastructure such as buildings, installations, or the road network.

It can be observed that the roads and streets themselves can act as barriers to the spread of fire when they have sufficient width. It is also necessary to emphasise that the generation and transport of embers (firebrands) and the creation of secondary ignition points (spot fires) have not been taken into account, which would result in a very different simulation. These are aspects that will be improved in future work.