IEEE Open Journal of
Intelligent Transportation
Systems

Received 25 January 2024; revised 26 April 2024 and 2 June 2024; accepted 7 June 2024. Date of publication 11 June 2024; date of current version 17 July 2024.

Digital Object Identifier 10.1109/OJITS.2024.3412820

A Big Data Architecture for Digital Twin Creation of Railway Signals Based on Synthetic Data

GIULIO SALIERNO [1], LETIZIA LEONARDI [1], AND GIACOMO CABRI [2] (Senior Member, IEEE)

[1] Department of Engineering "Enzo Ferrari," University of Modena and Reggio Emilia, 41125 Modena, Italy

[2] Department of Physics, Informatics and Mathematics, University of Modena and Reggio Emilia, 41125 Modena, Italy

CORRESPONDING AUTHOR: L. LEONARDI (e-mail: letizia.leonardi@unimore.it)

ABSTRACT Industry 5.0 has introduced new possibilities for defining key features of the factories of the future. This trend has transformed traditional industrial production by exploiting Digital Twin (DT) models as virtual representations of physical manufacturing assets. In the railway industry, Digital Twin models offer significant benefits by enabling anticipation of developments in rail systems and subsystems, providing insight into the future performance of physical assets, and allowing testing and prototyping solutions prior to implementation. This paper presents our approach for creating a Digital Twin model in the railway domain. We particularly emphasize the critical role of Big Data in supporting decision-making for railway companies and the importance of data in creating virtual representations of physical objects in railway systems. Our results show that the Digital Twin model of railway switch points, based on synthetic data, accurately represents the behavior of physical railway switches in terms of data points.

INDEX TERMS Big data, digital twin, machine learning, synthetic data, railway industry, artificial intelligence.

I. INTRODUCTION

THE RAILWAY industry, with its vast legacy systems, generates massive volumes of heterogeneous data from various assets like locomotives, tracks, signals, and critically, railway switch points. Switch points are key components in railway operations, responsible for controlling the route of trains by allowing them to be guided from one track to another. Given their pivotal role, switch points demand continuous predictive maintenance and monitoring to ensure safety and reliability. The inherent complexity of data generated by switch points, arising from a multitude of sensors and control systems produced by different manufacturers over decades, necessitates the development of accurate Digital Twin (DT) models. These digital twins can aptly simulate the conditions and potential failures of railway switch points by leveraging the available data. In this digitized era, a Digital Twin is conceptualized as a sensor-empowered digital representation of a tangible entity.

In this paper, we present our approach to creating a Digital Twin model specifically for railway switch points. The choice of switch points as the focus stems from their critical importance in railway operations and the challenges involved in effectively utilizing the heterogeneous data generated by these assets. Our Digital Twin model aims to provide a virtual representation of physical railway switch points, enabling simulation, monitoring, and predictive maintenance capabilities.

The advent of next-generation sensing capabilities within the factories of the future is paving the way for revolutionary opportunities in the manufacturing sector. The value derived from Big Data, once thought to be far-fetched, is now being harnessed due to the proliferation of both smart technologies and the Internet of Things (IoT). The railway industry, in particular, has discerned the significance of harnessing information from this expansive Big Data to elevate various sectors, encompassing maintenance, safety, and customer satisfaction. The escalating emphasis on sustainable railway transport further underscores the pivotal role of data analytics in the realm of railway operations.

With the challenges at hand, this paper ventures to propose a tailored Big Data architecture. The primary objective is to cater to the unique demands of managing the vast and

The review of this article was arranged by Associate Editor Yihui Wang.

diverse railway data, subsequently facilitating the streamlined development of Digital Twin models that can provide predictive insights. This architecture, in its essence, seeks to:

- facilitate the seamless aggregation and integration of data originating from diverse legacy systems;
- undergo preprocessing procedures aimed at enhancing the quality of sensor data by filtering out noise;
- offer customized data access mechanisms optimized for the advancement of Digital Twin frameworks;
- enable the scalable simulation of railway infrastructure assets via Digital Twin technologies;
- easy the development of predictive maintenance strategies by leveraging asset simulations.

In the grander scheme, the architecture is designed to resonate with the distinctive scale, variety, and analytical requisites of railway systems. It does share an affinity with conventional lambda architectures; however, custom inclusions like the Digital Twin Layer imbue it with advantages peculiar to the railway industry. While the methodologies and techniques elucidated herein have the potential to be extrapolated to other intricate industrial environments with a substantial legacy infrastructure, the primary emphasis remains on optimizing this architecture for the multifaceted world of railway data and the accompanying Digital Twin prerequisites.

The overarching ambition of this endeavor is to conceptualize and actualize a big data architecture that can adeptly ingest, process, and dissect data pertaining to railway switch points. This data will be the cornerstone upon which a Digital Twin model, encapsulating both raw and synthetic data, will be sculpted. Although this case study emphasizes railway switch points, the architectural framework conceived has the latitude to foster the genesis of other Digital Twin models, and by extension, applications rooted in predictive maintenance [1].

II. RELATED WORK

The huge potential of Big Data in the railway sector is confirmed by different works that identify that sector as an optimal domain for the application of Big Data solutions. The literature has identified the main features to cope with Big Data applications in the railway sector. Specifically [2]:

1) in terms of volume of data, hundreds of TB/day by different sources for the European railway system;
2) heterogeneity of the sources of information;
3) peculiarity of predictive algorithms for maintenance planning and its optimization and, in general, for decision-making applications.

These features justify identifying appropriate solutions and tools for the deployment of a Big Data infrastructure in this complex scenario, especially to cope with asset maintenance in the perspective of Industry 5.0.

The survey in [3] demonstrates that the use of Big Data into railway transportation systems can be classified on the basis of its applications. The authors describe that most works were carried out on railway vehicles, tracks, or signaling equipment. The authors' results demonstrate that most of the work focuses on data analysis of vehicle data (53%) while only few works (11%) propose applications for the analysis of signal equipment. Motivated by these results, we investigate the design of a Big Data infrastructure, especially focused on the DT model creation and analytics of objects composing the railway yard.

From Big Data architecture applications, authors in [4] propose a cloud-based Big Data architecture for real-time analysis of data produced by onboard equipment of high-speed trains. A framework for smart railway passenger stations by using Digital Twin technology is proposed by authors of [5] which focus on the importance of aspects related to business needs, architecture, data collection, model development, and potential applications for effective enablement of a DT model for the railway domain.

Recent studies have also highlighted the potential of Digital Twins and advanced technologies in shaping the future of railway systems. The work by [6] presents a survey of experts on the technologies, challenges, and opportunities for the railway system towards 2050, underscoring the importance of integrating cutting-edge technologies like Digital Twins, IoT, and AI. Additionally, researchers have explored the implementation of model-oriented approaches for the safe integration of GNSS-based virtual balises in ERTMS/ETCS Level 3 [7], and the application of control methods for virtual coupling in railway operations [8]. These advancements demonstrate the growing significance of Digital Twin and related technologies in shaping the future of the railway industry.

On the adoption of synthetic data, authors of [9] highlight their importance for condition monitoring of railway Digital Twin HVAC (Heating, Ventilation and Air Conditioning) systems. In this work, the DT of a HVAC system is augmented with synthetic data to overcome the lack of real data. Thus, the state of degradation is inferred by using a DT hybrid-model of the HVAC system composed both of real and synthetic data.

Recent explorations in Digital Twin concepts for railways have highlighted a critical need for enhanced data collection and integration, particularly for the effective implementation of AI and Machine Learning models. This need justifies the proposal for a comprehensive big data architecture, designed to gather, process, and utilize data specifically for railway signals. Such an architecture would be instrumental in fully enabling the creation and optimization of AI models.

The current integration of artificial intelligence in railway Digital Twins is limited, often focusing on applications like predictive maintenance. However, advancements in AI, especially in areas like explainability, causality, and reasoning, present an untapped potential to revolutionize railway Digital Twins. By fully harnessing these AI advancements, Digital Twins can evolve from passive monitoring systems to proactive, collaborative decision-making tools for human operators. This evolution is particularly crucial given that existing models typically cater to basic signaling systems and fall short in integrating with more advanced systems like the European Train Control System (ETCS). This gap is significant as it leads to a misalignment with the comprehensive requirements of railway signaling systems, including adherence to speed restrictions and movement authorities [10].

These identified limitations in scale, integration, and intelligence underline the need for the proposed railway Digital Twin architecture of this work. By leveraging a big data architecture that incorporates cloud-based microservice architectures, Internet-of-Things integration, and physics-informed machine learning, the proposed system aims to address these challenges. This innovative approach is targeted to surpass the limitations of current models, offering a more robust, scalable, and intelligent solution for railway systems, especially in the realm of signal processing and AI model development.

III. RAILWAY INTERLOCKING SYSTEM AS DATA SOURCE

The primary motivation of adopting a Big Data architecture is the huge volume of data produced by Railway Systems as Computer-based Interlocking systems. A Computer-based Interlocking System (CIS) is a complex safety-critical system that guarantees the absence of critical safety conditions, such as trains occupying the same track.

In particular, CIS communicates with sensors that monitor the state of the points in the railway yard via duplex communication. Moreover, CIS sends commands to each smart board, which controls the corresponding physical point on the line and collects data about its status. The collected data is written into CIS data storage as log files in XML format, which contain semi-structured data describing the point's behavior upon an issued command. This paper focuses on the data produced by the commands requested for the railway switch points. As the CIS controls only points under its governance, multiple CIS control a complete railway line that produces different log files according to the points under their governance. According to this complexity, multiple CIS produces large volume of data that must be collected and processed.

In the current version, the system can gather voltage-time and current-time parameters. In a future version further parameters can be considered, such as weather-related ones. The complexity is anyway given by the quantity of data gathered.

The proposed Big Data architecture takes into account the complexity of the railway domain and enables data ingestion from different data sources (i.e., multiple CIS) by defining data flow processes, as described later in Section V. A high-level communication schema between the CIS and the physical level is shown in Figure 1.

Raw data collected from sensor boards connected to railway switch points include different information that can be summarized as follows:

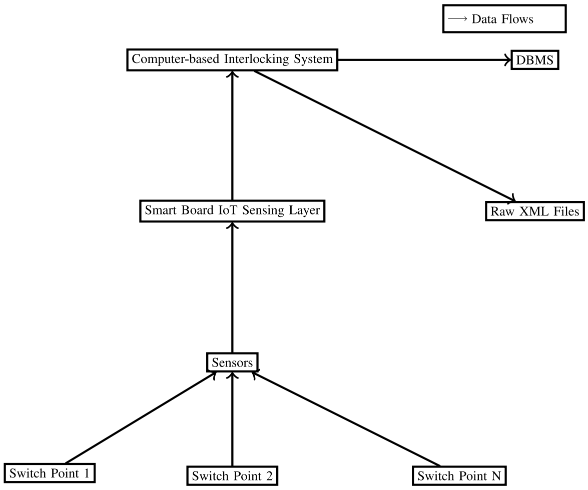
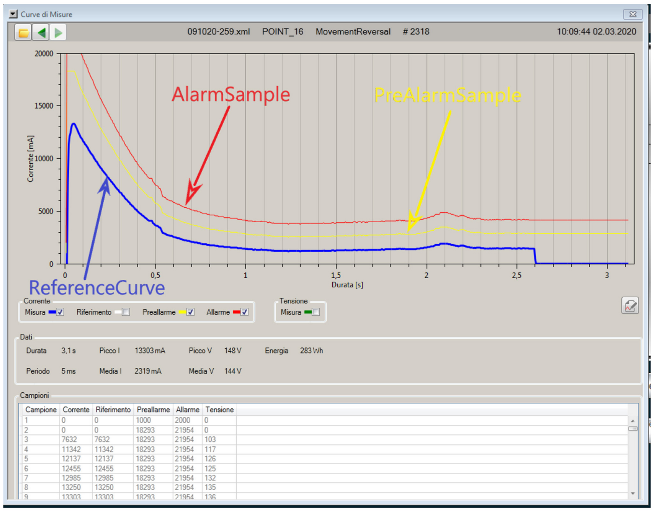


FIGURE 1. High-level representation of communication flows between CIS and railway signal sensors.



Curve di Misure

091020-259.xml POINT_16 MovementReversal # 2318 10:09:44 02.03.2020

Corrente: Misura ☑ Riferimento □ Preallarme ☑ Allarme ☑

Tensione: Misura □

Dati

Durata 3,1 s Picco I 13303 mA Picco V 148 V Energia 283 Wh

Periodo 5 ms Media I 2319 mA Media V 144 V

Campioni

Campione	Corrente	Riferimento	Preallarme	Allarme	Tensione
1	0	0	1000	2000	0
2	0	0	18293	21954	0
3	7632	7632	18293	21954	103
4	11342	11342	18293	21954	117
5	12137	12137	18293	21954	126
6	12455	12455	18293	21954	125
7	12985	12985	18293	21954	132
8	13250	13250	18293	21954	135
9	13303	13303	18293	21954	136

FIGURE 2. Plot of data sample representing a switch point movement.

1) *timestamps* $= \{(t_1, c_1), (t_2, c_2), \ldots, (t_n, c_n)\}$. Each time value t_i represents the operation time of the command c_i assigned by the logging server and recorded by the smartboard;
2) information about the *smartboard* which collects the data. A smartboard S collects the data from a channel number cn using a sampling frequency of f;
3) information about the *action* requested by the interlocking. The interlocking requests an action of type m, which involves a total of n_{tot} movements, and the current movement is n_{cur};
4) a list of raw values representing Power Voltage supplied to the switch points over time is given by $V = [v_1, v_2, \ldots, v_n]$ at times $t = [t_1, t_2, \ldots, t_n]$ to execute the movement.

For example, in Figure 2, we present samples collected from a railway switch point. These samples contain different types of information that produce three distinct curves:

- The *ReferenceCurve*, denoted by $I = [i_1, i_2, \ldots, i_n]$, is a discrete sample curve representing the point's current drain over time, generated upon the command requested by a CIS. This curve is used to derive the other two curves at times $t = [t_1, t_2, \ldots, t_n]$.
- *PreAlarmSample:* a pre-threshold curve that is computed by adding an intermediate threshold value t_{th} to the *ReferenceCurve*, i.e., $PreAlarmSample = [i_1 + t_{th}, i_2 + t_{th}, \ldots, i_n + t_{th}]$
- *AlarmSample:* an alarm curve that is computed by adding a dynamic alarm threshold t_{alarm} to the *PreAlarmSample*, i.e., $AlarmSample = [i_1 + t_{th} + t_{alarm}, i_2 + t_{th} + t_{alarm}, \ldots, i_n + t_{th} + t_{alarm}]$

Therefore, the *PreAlarmSample* and the *AlarmSample* curves are dynamically computed based on the *ReferenceCurve* values. These curves represent threshold values above which the working conditions of the points are potentially anomalous and must be considered indications of potential failures.

As shown in the *ReferenceCurve*, the highest power consumption is typically observed at the beginning, i.e., when starting the engine and releasing the switch lock. During the actual movement of the tongue rails, the electric current consumption remains nearly constant at a medium level. At the end of the switch movement, the electric current decreases, and when the switch tongues are locked back, the motor shuts down, and the measured electric current returns to zero amperes [11].

This XML data, represented in Figure 3, demonstrates the intricate nature of railway sensor data, also in our current version that consider only voltage-time and current-time parameters. The data encapsulates various timestamps marking the operation times of commands logged by the server. Additionally, details about the smartboard–such as channel numbers and sampling frequencies–are outlined. Vital information regarding the interlocking actions, including the type of action and the details about movements, is also encompassed.

Furthermore, the data provides sensor metrics, particularly power voltage and current measurements, both supplied with distinct timestamps. Such diverse and intricate data not only highlights the complexity of railway sensor data but also showcases the variance in types and formats of readings–ranging from timestamps to integers and from commands to floating-point values.

Processing and drawing meaningful conclusions from this extensive array of sensors demand a robust framework. Our proposed big data architecture is meticulously designed to cater to and simplify the handling of such heterogeneous and multifaceted railway sensor data.

```
<dataCollection>
    <!-- Timestamps -->
    <timestamps>
        <timestamp>
            <time>2023-10-05T08:00:00Z</time>
            <command>commandA</command>
        </timestamp>
        <timestamp>
            <time>2023-10-05T09:00:00Z</time>
            <command>commandB</command>
        </timestamp>
        <!-- ... -->
    </timestamps>

    <!-- Smartboard Information -->
    <smartboard>
        <channelNumber>5</channelNumber>
        <samplingFrequency>10Hz</samplingFrequency>
    </smartboard>

    <!-- Interlocking Action Information -->
    <interlockingAction>
        <type>actionType1</type>
        <totalMovements>100</totalMovements>
        <currentMovement>10</currentMovement>
    </interlockingAction>

    <!-- Power Voltage Information -->
    <powerVoltages>
        <voltage time="2023-10-05T08:00:00Z">230</voltage>
        <voltage time="2023-10-05T09:00:00Z">235</voltage>
        <!-- ... -->
    </powerVoltages>

    <!-- Currents in Amperes Information -->
    <currents>
        <current time="2023-10-05T08:00:00Z">5</current>
        <current time="2023-10-05T09:00:00Z">5.2</current>
        <!-- ... -->
    </currents>
</dataCollection>
```

FIGURE 3. Sample XML representation of the data including current samples with values.

IV. A BIG DATA FRAMEWORK FOR RAILWAY SIGNAL DATA

The railway industry relies heavily on various physical assets, including locomotives, tracks, signals, and switch points, to ensure the safe and efficient operation of railway networks. Railway *switch points*, in particular, play a crucial role in directing trains from one track to another, thereby facilitating smooth transitions and route adjustments. As such, the condition and performance of switch points are of paramount importance for ensuring operational reliability and safety within railway systems.

Traditionally, the maintenance and monitoring of railway assets have been carried out through manual inspections and periodic assessments. However, with the advent of advanced sensor technologies and data analytics, there is a growing opportunity to leverage data-driven approaches for predictive maintenance and real-time monitoring of railway infrastructure. By harnessing the vast amounts of data generated by sensors installed on railway assets, it becomes possible to gain valuable insights into asset health, performance trends, and potential failure risks. In this section, we provide a comprehensive overview of the proposed Big Data framework for handling railway signal data. The framework is designed to address the challenges associated with managing the large volume of data generated by multiple switch points in a railway yard while facilitating the development of Digital Twin models for predictive maintenance and optimization.

The proposed Big Data architecture implements fundamental steps for a Big Data pipeline. In particular, this architecture is designed to provide the following capabilities, including the ability to design Digital Twin models:

1) Data ingestion: The architecture is capable of ingesting data from various sources, such as sensors, switches, and other devices, in real-time or batch mode adopting Apache NiFi.
2) Data storage: The architecture is designed to store the massive amount of data generated by the switch points in a distributed file system, such as Hadoop Distributed File System (HDFS).
3) Data processing: The architecture provides tools for processing the data, including batch processing with Hadoop MapReduce.
4) Data analysis: The architecture supports data analysis using machine learning algorithms, statistical models, and other data analytics techniques.
5) Data visualization: The architecture provides visualization tools to enable users to interactively explore and visualize the data.
6) Digital twin modeling: The architecture is also designed to enable the development of Digital Twin models, which are virtual representations of physical assets, systems, or processes. This allows for predictive maintenance, optimization, and other advanced analytics use cases.

The storage Layer is tasked with managing the vast volumes of data generated by switch points and other railway assets. Leveraging distributed file systems such as the Hadoop Distributed File System (HDFS), it offers a robust and fault-tolerant storage platform capable of accommodating diverse data types and formats. Moreover, the Storage Layer preserves the raw, immutable sensor data in its original format, ensuring data integrity and traceability for subsequent analytics tasks. The storage Layer is tasked with managing the vast volumes of data generated by switch points and other railway assets. Leveraging distributed file systems such as the Hadoop Distributed File System (HDFS), it offers a robust and fault-tolerant storage platform capable of accommodating diverse data types and formats. Moreover, the Storage Layer preserves the raw, immutable sensor data in its original format, ensuring data integrity and traceability for subsequent analytics tasks. The architecture shown in Figure 4 takes the Lambda Data architecture [12] as a reference model and is composed of four plus one layers, briefly explained in the following and fully described in Section V.

The **Storage Layer** is responsible for implementing data storage and contains the storage platform, which provides a distributed and fault-tolerant file system. This layer should store data in its original form, and new data will be created from the upper layers.

The storage Layer is tasked with managing the vast volumes of data generated by switch points and other railway assets. Leveraging distributed file systems, such as the Hadoop Distributed File System (HDFS), offers a robust and fault-tolerant storage platform capable of accommodating diverse data types and formats. Moreover, the Storage Layer preserves the raw, immutable sensor data in its original format, ensuring data integrity and traceability for subsequent analytics tasks.

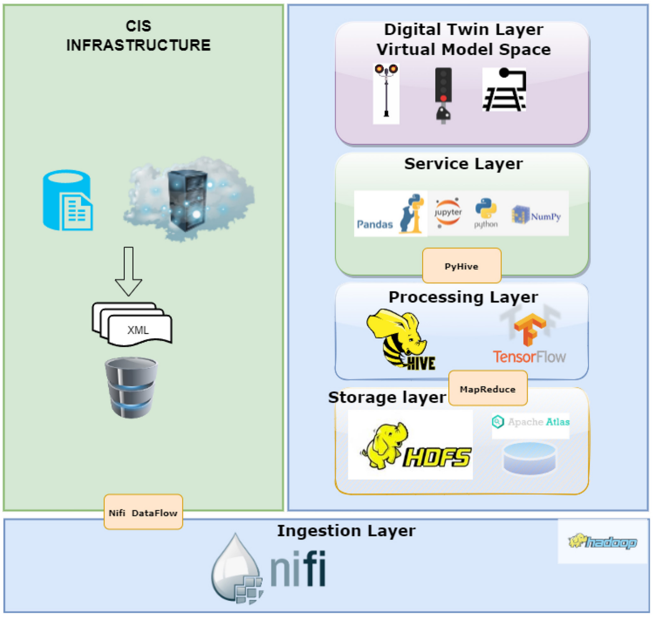


FIGURE 4. Stack for Railway Big Data Management: Architectural Layers and Technologies.

The **Processing Layer** is responsible for executing data manipulation and transformation tasks to prepare the raw sensor data for analysis and modeling. Leveraging technologies, such as Hadoop MapReduce, facilitates batch processing of large datasets, enabling data aggregation, filtering, and feature extraction. By transforming raw data into structured datasets, the Processing Layer lays the groundwork for advanced analytics and modeling tasks further downstream.

The **Service Layer** serves as the interface between end-users and the underlying data and analytics infrastructure. This layer provides a suite of analytics services and tools, including machine learning algorithms, statistical models, and data visualization capabilities, to support various use cases such as predictive maintenance, anomaly detection, and performance optimization. By empowering users to interactively explore and analyze railway signal data, the Service Layer facilitates data-driven decision-making and operational excellence within railway systems.

At the pinnacle of the framework lies the Digital Twin Layer, which enables the creation and management of virtual replicas of physical assets, systems, or processes within railway networks. By synthesizing data from multiple sources and leveraging advanced modeling techniques, the Digital Twin Layer provides a holistic view of asset health, performance, and operational conditions. Through virtual simulations and predictive analytics, Digital Twins empower railway operators to proactively identify maintenance needs, optimize asset utilization, and enhance overall system resilience. The **Digital Twin Layer** is responsible for creating and storing virtual objects based on the data collected, processed, and stored in the Storage and Processing Layers. The enriched model generated from the processed data can be used for running simulations, studying performance issues, generating possible improvements, and creating synthetic data for further analysis. The Digital Twin Layer enables the generation of virtual models by emulating a physical object with appropriate methods or functions, reading object data from the Processing Layer, storing it into internal attributes of the objects, and encapsulating everything into a Python class. A Python object as a class instance can be used in external simulation programs for probing data or for optimizing internal parameters.

In addition to the above described layers, the **Ingestion Layer** acts as an interface between the architecture and the external data source. It implements all tasks for data ingestion from the external data store and is based on ingestion tools, which allow the definition of data flows. A data flow consists of a variable number of processes that transform data by creating flow files that are moved from one process to another through process relations.

The Big Data architecture, described in the following, has the goal of enabling Digital Twin creation even if it shares many similarities with the general Lambda architecture reference model. However, there are some key differences in terms of the specific characteristics and goals of our proposal.

The major difference is that the goal is not just to process and analyze large volumes of data, but also to create and maintain accurate Digital Twin models. Digital Twins are virtual replicas of physical objects, systems, or processes that can be used for simulation, monitoring, and optimization. To achieve this, the Big Data architecture for Digital Twin creation needs to include additional components and processes that are specific for Digital Twin modeling.

Therefore, in terms of achieving interactions between the Digital Twin models and the Big Data architecture, our proposed architecture includes a separate layer called the *Digital Twin Layer*, as we have already shown. This layer is responsible for handling the communication between data stored by the Big Data architecture and the Digital Twin models. It enables the DT models to access and analyze the data from the Big Data architecture and also allows the models to update the data in real-time based on their simulations and predictions.

Compared to the Lambda reference architecture, the addition of the Digital Twin Layer provides a more streamlined and efficient way to create and maintain DT models. It allows for a more seamless integration between the data processing and modeling components of the architecture, and enables more advanced applications such as predictive maintenance and real-time optimization.

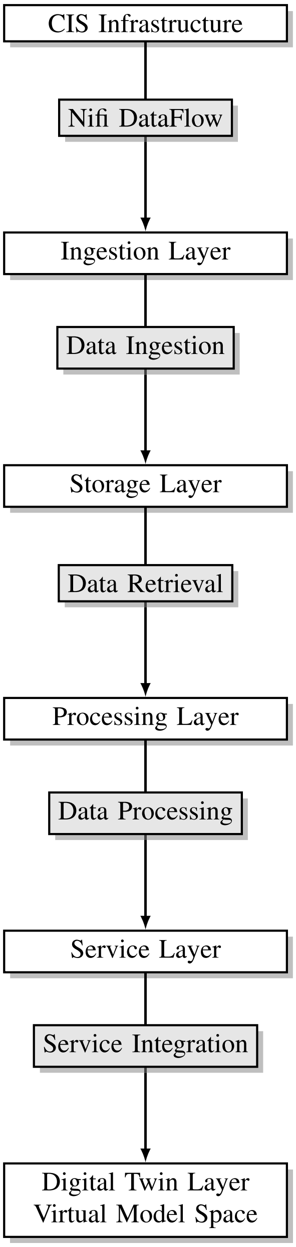


FIGURE 5. Layered architecture with communication flows between the key components.

Figure 5 represents the layered architecture of the system, highlighting the key components and their communication flows. At the base is the CIS Infrastructure, which serves as the underlying data ingestion mechanism using Nifi DataFlow. The Ingestion Layer receives the data from the CIS Infrastructure, and then passes it on to the Storage Layer for persistence.

The data stored in the Storage Layer is then accessed by the Processing Layer, where various data processing tasks are performed, such as with TensorFlow and MapReduce. The results of the data processing are then made available to the Service Layer, which integrates the processed data and provides services to the Digital Twin Layer.

The Digital Twin Layer represents the virtual model space, where the integrated services and data are consumed to enable the Digital Twin functionalities. The arrows between the layers indicate the directional flow of data and communication between the components.

This layered architecture allows for modular design, scalability, and flexibility in the system, enabling efficient data ingestion, processing, and integration to support the digital twin capabilities.

The proposed architecture enables Digital Twin (DT) creation through several key capabilities tailored to the requirements of virtual modeling. Firstly, the Digital Twin Layer provides a dedicated interface for developing and accessing virtual representations of physical assets using processed sensor data. This layer contains specialized application programming interfaces (APIs) and data structures engineered explicitly for DT models. Secondly, the Storage Layer is designed to preserve raw, immutable sensor data in order to construct accurate historical simulations within the DT. It also permits storing multiple copies of datasets at varying aggregation levels to support the needs of DT development.

Additionally, the Processing Layer implements essential data transformations by extracting key features from the raw sensor streams. This vital preprocessing step adapts the data into structured datasets better suited for training and updating DT models. Furthermore, the Service Layer supplies interactive tools such as notebooks that empower iterative development of DT models utilizing the processed data. This layer enables rapid prototyping and refinement of the virtual models. Several additional processes and components are vital to fully support the DT development lifecycle. These include modules for mapping virtual models to physical assets, synchronizing bidirectional data flows to maintain model accuracy, configuring simulations to recreate operational conditions, evaluating model fidelity, and monitoring asset status. Other critical elements are digital thread trackers relating DT events to physical counterparts, physics-based models capturing engineering knowledge for precise DT behavior, and prediction engines forecasting asset maintenance needs.

In summary, the proposed architecture is tailored to DT creation through dedicated layers for preparing, storing, and accessing data, as well as tightly integrating data pipelines with modeling tools. The additional processes and components address the end-to-end requirements for virtual modeling, from physical asset mapping to operational monitoring.

The introduction of a dedicated Digital Twin Layer in the architecture promotes enhanced streamlining and efficiency in constructing and maintaining DT models through several mechanisms. Firstly, the consolidation of all DT-specific logic into a single modular layer enables improved separation from generic data processing components, promoting organized architecture. Secondly, optimizations such as data representations and application programming interfaces custom-engineered for virtual entities accelerate development. Thirdly, the unified interface concurrently servicing data pipelines and modeling tools simplifies integration and maintenance. Fourthly, efficient access to preprocessed data avoids repeated raw transformation, expediting model building. Fifthly, abstraction of physical details enhances model reusability across assets, improving scalability. Sixthly, coordinated bi-directional data flows ensure model accuracy via synchronization to current data. Finally, common interfaces for model management reduce lifecycle complexity. In summary, the dedicated Digital Twin Layer creates an optimized pipeline between data and models via enhancements explicitly targeting efficiency in virtual entity creation, usage, and maintenance.

TABLE 1. URI stack for a single resource.

URI Type	Example Value
RealURI	/mnt/hdfs/smartboard1/ch1/p1.avro
VirtualURI	/smartboard1/ch1/p1.avro

V. ARCHITECTURE IMPLEMENTATION

The architecture was implemented using components of the Hadoop framework,[1] which is a distributed processing system that enables distributed processing of large data sets across clusters of computers using commodity hardware. In the following, we detail all the layers introduced in the above section: i) Storage Layer, ii) Processing Layer, iii) Service Layer, iv) Digital Twin Layer and v) Ingestion Layer.

A. STORAGE LAYER: DATA GOVERNANCE AND LOGICAL ORGANIZATION

The Storage Layer is based on the data lake concept [13]. A critical aspect of a data lake is preserving data in their original format. The Extract, Load, and Transform (ELT) process of a data lake stores data in its raw format and automates the process of extracting new insights from it. This implies that multiple copies of a single object can exist with varying aggregation levels and metadata.

Preserving data in their original format is crucial to avoid scenarios where the data on the platform become unusable due to complexity, size, variety, or a lack of metadata. To achieve this, we adopt a URI abstraction mechanism that simplifies data access and establishes a data governance policy. A Uniform Resource Identifier (URI) is a character sequence that uniquely identifies resources on the platform. For example, at the Storage Layer, a resource's URI is its absolute pathname.

To avoid the need for defining multiple URIs for each resource, which can be used by multiple components at different architectural layers, we abstract URI access for data retrieval. As a result, the *RealURI* points to a resource stored on the distributed file system while abstracting its physical location. A *RealURI* is bound to a single *VirtualURI*, which abstracts the specific pathnames used by a particular

[1] https://hadoop.apache.org

TABLE 2. URI abstractions for storage resources.

URI Type	URI
RealURI	`hdfs://data_path/smart_board_number/channel/point_number`
VirtualURI	`adc://data_path/smart_board_number/channel/point_number`
PresentationURI	`hive:adc://data_path/smart_board_number/channel/point_number`

implementation of distributed file systems. A *VirtualURI* is an optional URI created when a Processing Layer or Service Layer component uses a resource stored on the file system.

For example, the URI stack for a single resource is presented in Table 1, where each resource is identified by: the smartboard identifier, the channel number in which a point is attached, and the point number.

The separation enables modularity and loose coupling between layers. The Storage Layer only handles RealURIs. The mapping between RealURI and VirtualURI is maintained in a lookup registry by the Processing Layer. This registry assigns a unique VirtualURI for every RealURI so other layers do not need to know the physical location details.

The PresentationURI is created on demand by the Service Layer to provide custom views of the VirtualURI data for end users. This on-the-fly creation of PresentationURIs prevents tighter coupling between layers. Only the VirtualURI persists across architecture layers for consistent data access. An analogous URIs stack shown in Table 1, is adopted to provide data access to the applications. As an example, the *VirtualURI* refers to the resource at a platform level, while the *PresentationURI* represents a table view of the data created by the Apache HIVE[2] in the Processing Layer. The full URIs stack adopted by the applications at each architectural layer is reported in Table 2. It is important to note that while resources can be assigned to an unbounded number of *PresentationURI*s depending on the type of components that use the data, the *VirtualURI* is mandatory and refers to a single *RealURI*.

Figure 6 depicts the logical organization of folders and data in the Storage Layer that is structured into three different areas:

1) *Landing Zone*, which is the area where raw data is stored after the ingestion phase.
2) *Gold Zone*, which is the area where cleaned and aggregated data is stored after processing. This area also maintains a view of data and DT model artifacts. Additionally, copies of data at different granularity levels (raw, pre-processed, and results folders) are kept for further analysis or produced as results of queries.
3) *Dev Zone*, which is the area where end-users and analysts access data through the data catalog (where URI resources are stored), train analytical models, and query data.

In the following, we describe the semantics of the folder structure:

- Raw: The initial repository for incoming data, typically raw files emanated from source systems. The SMART_IO directories collect both data inputs and outputs referred to each board on a specific channel represented by their corresponding folders.
- Projects: Diverse directory structures dedicated to orchestrating the distinct workflows of diverse projects. In particular, the Artifacts directory is a repository designed for storing analytical outcomes and relevant outputs primed for dissemination. Data are organized for production usage and analysis via "Hive Tables" stored in the corresponding directory enabling SQL-based data accesses.
- Users: Representing the analysts involved in various projects. Each user has its own corresponding directory (Working Directory) which may be tethered to a specific project. In addition, Data Catalogue is a repository containing metadata describing data belonging to a specific project.

The directional arrows elucidate the data's progression: from its raw ingestion phase, through preprocessing, culminating in the analysis results. The intermediary folders within SMART_IO bolster data isolation, ensuring replicability for each specific analyst or project. The hierarchically-structured directories intuitively segment the lake, organizing it into logical domains and facilitating data flow. This meticulous arrangement empowers a multitude of users to synergistically partake in the analytical process while adhering to governed protocols.

The **Storage Layer** was implemented using the Hadoop File System (HDFS), a fault-tolerant distributed file system that provides high data availability. HDFS stores raw data in its original format as provided by the Ingestion Layer's tasks, as shown in Figure 7. The ingestion tasks extract data and metadata and aggregate them into a specific folder on HDFS that represents the data for a particular point.

Data representing point behavior (see Section III) are stored in their original format as XML files. Therefore, these data must be processed and transformed to create new datasets, which is the task of the Processing Layer.

B. PROCESSING LAYER: DATA TRANSFORMATION AND AGGREGATION

The Processing Layer is a critical component of our proposed architecture, responsible for transforming the raw XML files into a more usable format and aggregating them into relevant datasets. This transformation is accomplished through the coordination of two main components.

The first component is a dataset builder module that processes the raw data and extracts relevant features (see

[2] https://hive.apache.org

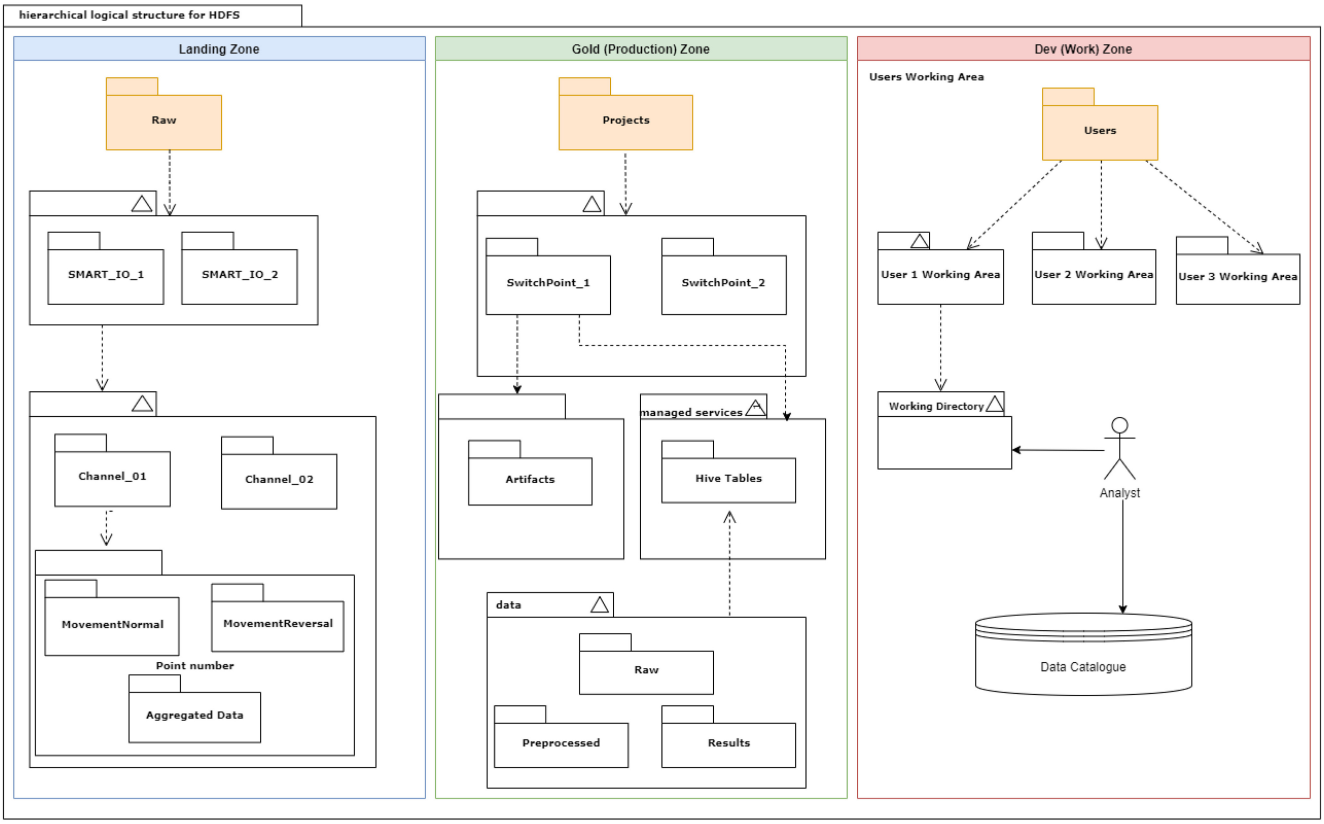


FIGURE 6. Hierarchical Logical Structure of HDFS.

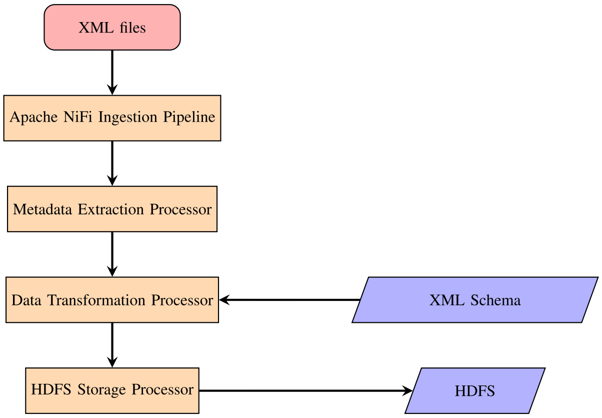

XML files
Apache NiFi Ingestion Pipeline
Metadata Extraction Processor
Data Transformation Processor
XML Schema
HDFS Storage Processor
HDFS

FIGURE 7. Ingestion pipeline to store data and metadata on the Storage Layer.

Figure 8). This module parses the XML files, extracts the attributes, and aggregates them into CSV files using various aggregation functions (*min*, *max*, *avg*). The aggregated data is then written back to the Hadoop Distributed File System (HDFS) in the directory of the original data.

The second component of this layer is responsible for importing the aggregated CSV files into HIVE tables, thereby facilitating subsequent data analysis. HIVE, a data warehousing tool provided by the Hadoop stack, employs a SQL-like language (HIVEQL) for querying data. To import data into HIVE tables, we define a general schema that aligns with the structure of the switch point data. This schema, derived from the aggregated CSV files created by the dataset builder module, represents a generic switch point data in the following structure:

```
smart_board_number_point_number_agg
(RecordSampleTime DATE,
MovTime FLOAT,
current_mA FLOAT,
voltage_V FLOAT)
```

These tables store aggregated data containing extracted features from the raw data. These features include: 1) timestamps indicating when the samples were collected, 2) the estimated time to complete an operation, 3) the average electric current (in *mA*) issued by the point, and 4) the average voltage (in *V*). Features 2), 3), and 4) are obtained by aggregating single measurements contained in the original data.

Furthermore, we define a HIVE table to store raw switch point data in table format, which is structured as follows:

```
smart_board_number_point_number_raw
(RecordSampleTime DATE,
MovementNumber INTEGER,
MovementDestination BOOLEAN
current_mA FLOAT, voltage_V FLOAT)
```

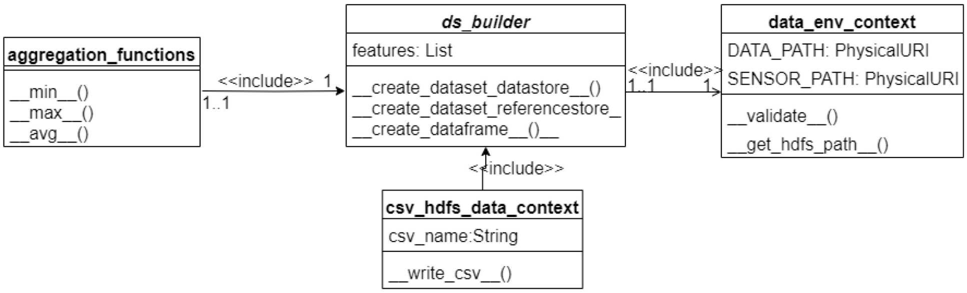


FIGURE 8. Class diagram of the dataset builder module.

This table stores metadata information, including *MovementDestination*, which represents the type of movement executed by the switch point (Normal or Reversal), and *MovementNumber*, representing the total number of movements executed since its initialization. These efficient transformation and organization of data play a vital role in facilitating subsequent stages of model creation and analysis.

The motivation for segregating aggregated data into the Gold Zone while maintaining the Dev Zone for modeling activities follows prudent data governance practices for production-grade analytics pipelines. The Gold Zone contains carefully curated datasets that have undergone transformations and aggregations from raw inputs via Extract-Load-Transform (ELT) processes. This zone houses cleansed and structured data engineered to serve as reliable information assets for downstream analytics and monitoring applications. In contrast, the Dev Zone provides an environment for data scientists and analysts to freely explore data, construct models, execute ad-hoc queries, and conduct various experimentations without risk of compromising the integrity of the data in the Gold Zone. By keeping aggregated datasets within the governance of the Gold Zone, accidental modifications or deletions during exploratory work are avoided. This separation also enables enforcement of appropriate access controls on production data resources. Meanwhile, the flexibility of schema and storage in the Dev Zone empowers users to join, sample, and modify data as required for modeling innovation without the constraints of hardened schemas typical of the Gold Zone. In summary, the proposed segregation balances the critical needs of safeguarding production data quality via the Gold Zone while enabling exploratory creativity in the Dev Zone.

C. SERVICE LAYER: DATA ACCESS AND MODEL CREATION

The Service Layer, also known as the Presentation Layer, encapsulates the tasks required to build analytical models, graphically visualize data, and set up Digital Twin (DT) models. To accomplish these tasks, we leverage the capabilities of Jupyter notebooks,[3] a comprehensive solution designed to support the scientific computing workflow, from interactive exploration to publishing detailed computation records.

Jupyter notebooks organize code into cells, which contain chunks of code that can be individually edited and executed. The output from each cell appears directly below the cell and is stored as part of the document [14]. With support for a variety of programming languages, Jupyter notebooks can integrate numerous open-source tools for data analysis, including but not limited to Numpy,[4] Pandas,[5] and Matplotlib.[6]

These tools provide the capability to parse structured data, perform data manipulation, and visualize data using built-in libraries. Particularly, the Pandas' *DataFrame* data structure is widely adopted as the input format for many analytical models provided by machine learning libraries such as scikit-learn[7] and SciPy.[8]

In the development of the DT model representation of a railway switch point, we employed pure Python. For the generation of synthetic data, we utilized the TensorFlow framework,[9] a robust platform that supports comprehensive machine learning and deep learning workflows. This integrated environment offers a flexible and accessible platform for the creation, training, and refinement of analytical models, making it an integral part of our proposed architecture.

D. DIGITAL TWIN LAYER: VIRTUAL OBJECT DESIGN

To effectively implement the Digital Twin (DT) Layer, we adhered to a methodology composed of the following steps [15], [16]:

- *Defining Virtual Objects:* The initial step in the process involves the definition of the virtual objects, derived from the data available in the Storage and Processing Layers. These virtual objects serve as digital replicas of physical entities or systems in the real world. It is essential that these virtual objects encapsulate all relevant details necessary for simulation or analysis. This includes, but is not limited to, the object's physical

[3] https://jupyter.org

[4] https://numpy.org

[5] https://pandas.pydata.org

[6] https://matplotlib.org

[7] https://scikit-learn.org/stable/index.html

[8] https://scipy.org

[9] https://www.tensorflow.org

properties, its behavior under different conditions, and the interrelations with other entities in the system. The creation of a comprehensive virtual object provides a foundation upon which high-fidelity simulations and analyses can be built.

- *Development of Algorithms or Models:* Once the virtual objects are defined, the next step involves the development of algorithms or models that leverage the processed data to instantiate these virtual objects. The design of these algorithms needs to effectively translate data into meaningful features that contribute to a high-fidelity digital representation of the object or system. This might involve a variety of methods, such as machine learning techniques, statistical analysis, and systems modeling, among others. The accuracy and reliability of the virtual objects greatly depend on the sophistication of these algorithms and models.
- *Storing Virtual Objects:* After the virtual objects are created, they need to be stored in a manner that allows easy and quick access to them by the other layers of the architecture. This storage strategy may involve the creation of dedicated databases or repositories, defined with appropriate APIs or interfaces. These APIs or interfaces should be designed to allow secure and efficient querying and manipulation of the virtual objects. Such an arrangement ensures that the Digital Twins are readily available for further operations and applications, providing the backbone for the subsequent steps.
- *Utilization of Virtual Objects:* The final step in the process involves the application of the virtual objects. They can be used to run simulations that replicate real-world scenarios, study performance under varied conditions, identify potential improvements, and generate synthetic data for further analysis. By leveraging the capabilities of the Service Layer to emulate the behavior of the physical entities and systems, the virtual objects serve as inputs, effectively bridging the gap between the digital and physical worlds. The insights gained from these simulations and analyses can provide valuable inputs for decision-making processes and future development.

E. INGESTION LAYER: DATA COLLECTION AND FLOW MANAGEMENT

The Ingestion Layer is a crucial component of any data pipeline and plays a pivotal role in data collection. This layer has been implemented using Apache NiFi, a powerful and reliable system designed to automate the flow of data between systems. Apache NiFi is based on the principles of flow-based programming, an approach that defines applications as networks of "black box" processes, which exchange data across predefined connections by message passing.

Apache NiFi allows to design complex dataflows graphically and handles failure recovery and data backpressure seamlessly, making it an excellent choice for this layer of the architecture. It offers many built-in Processors for data ingestion, transformation, and distribution. Each Processor has a specific role and can be configured to meet specific needs. In addition, the Controller provides a variety of services to the Processors and components involved in data processing and flow management, such as offering connection services to various data sources like relational databases and FTP servers. In our specific use case, a data flow extracts data from the CIS external sources and stores it on the platform. This flow is not random but follows a predefined route that describes the process of data extraction and storage.

An example of such a data flow is one that combines data from the external file system, as shown in Figure 9. This particular flow reads files from a local file system, unpacks them, and then writes the unpacked data into a specific folder on the HDFS.

Each file that is read and unpacked is converted into a flow file - the basic processing entity in NiFi. These flow files then follow the data flow defined in NiFi and are eventually written to the HDFS. This specific implementation shows the power and flexibility of Apache NiFi in handling varied data sources, different data types, and complex data flow requirements. It also showcases how NiFi can interact with other systems in the big data ecosystem, such as HDFS in this case.

This ingestion mechanism ensures a reliable, efficient, and traceable data collection process, laying a solid foundation for subsequent stages in the data pipeline, such as processing and analysis.

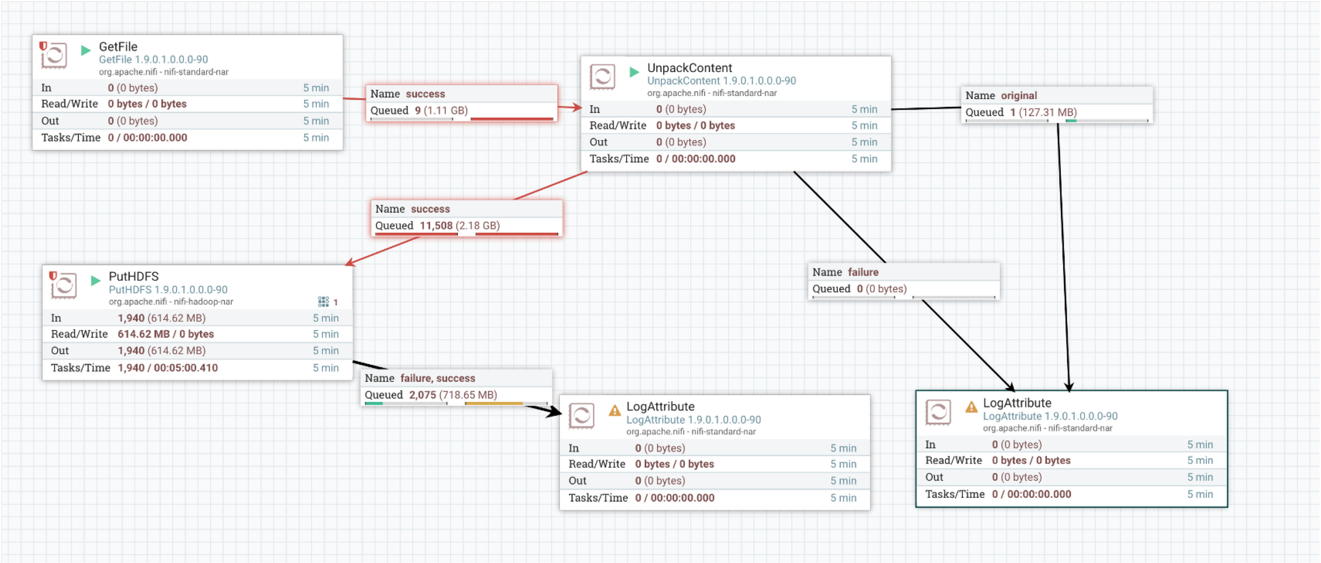
Before data can be employed in the analysis, it must be transformed to fulfill the requirements of analytical models. In this sense, the pre-processing of data is done by the Processing Layer which implements all the tasks required to build datasets from raw data.

VI. DESIGN AND EVALUATION OF A DIGITAL TWIN MODEL FOR RAILWAY SWITCH POINTS

The design of a DT model requires identifying the characteristics to be included in the digital objects, which will determine the complexity and structure of the Digital Twin model. In the proposed case study, we decided to model only the basic characteristics of a railway switch point based on the data detailed in Section III, for the sake of simplicity. Figure 10 presents the DT model pipeline detailing the steps followed during the creation of our Digital Twin model.

The first two steps involve data ingestion and processing, which are detailed in Section V. Step 3 specifies the core step for creating the DT model. A DT model should simulate the behavior of the physical object and predict the behavior of its physical counterpart. The last step is in charge of performing simulations on the DT model created in the previous step.

In Figure 11, we present the diagram illustrating the three primary classes plus one of the DT model. The `DigitalTwinModel` class represents the abstract class of a DT, which defines methods to access a data map, process and update the data. The `HIVE API` class represents



GetFile
GetFile 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name success
Queued 9 (1.11 GB)

UnpackContent
UnpackContent 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

Name original
Queued 1 (127.31 MB)

Name success
Queued 11,508 (2.18 GB)

Name failure
Queued 0 (0 bytes)

PutHDFS
PutHDFS 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-hadoop-nar
1

In	1,940 (614.62 MB)	5 min
Read/Write	614.62 MB / 0 bytes	5 min
Out	1,940 (614.62 MB)	5 min
Tasks/Time	1,940 / 00:05:00.410	5 min

Name failure, success
Queued 2,075 (718.65 MB)

LogAttribute
LogAttribute 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

LogAttribute
LogAttribute 1.9.0.1.0.0.0-90
org.apache.nifi - nifi-standard-nar

In	0 (0 bytes)	5 min
Read/Write	0 bytes / 0 bytes	5 min
Out	0 (0 bytes)	5 min
Tasks/Time	0 / 00:00:00.000	5 min

FIGURE 9. Example of an Apache NiFi dataflow implementing the ingestion pipeline depicted in Figure 7.

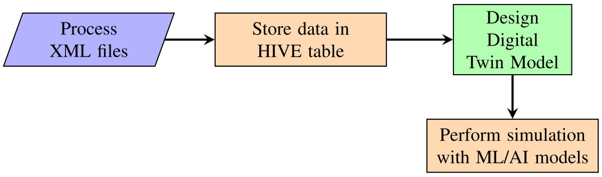

Process XML files
Store data in HIVE table
Design Digital Twin Model
Perform simulation with ML/AI models

FIGURE 10. DT Pipeline Creation.

the interface class that defines methods for accessing the data stored on the HIVE tables described in the previous section. It provides two methods to query and update the data. The `DTSwitchPoint` class implements the physical railway switch point as a `DigitalTwinModel` by defining methods and a data structure that stores switch point data.

We provide essential parameters of the `DTSwitchPoint` class, including electric voltage (V) and current (mA) values over time related to the movement of a switch point, as shown in Figure 12.

To simulate the behavior of switch points in terms of electric current and voltage curves, we have extended the DT model with a `TimeGAN` class by incorporating a TimeGAN model [17]. The TimeGAN model can learn and generate new synthetic data for these features. It is capable of generating fully synthetic data by being trained on a representation of the raw features as a time-series.

A Generative Adversarial Network (GAN) [18] consists of two neural networks: a generator G and a discriminator D. The generator takes a random noise vector z as input and generates synthetic data $x = G(z)$. The discriminator takes a data sample x as input and produces a probability $D(x)$ that indicates whether the input data (generated by the generator) is real or fake.

The two networks are trained adversarially: the generator is trained to produce synthetic data that can deceive the discriminator, while the discriminator is trained to distinguish real data from synthetic data. This is achieved by minimizing the following objective function:

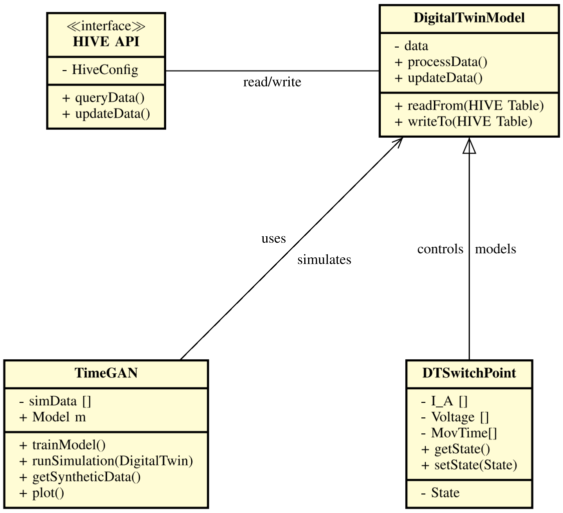


FIGURE 11. UML diagram classes of the railway switch point Digital Twin design.

$$\min_{G}\max_{D}\Big\{\mathbb{E}x \sim p_{data}(x)\big[\log D(x)\big] + \mathbb{E}_{z\sim p_z(z)}\big[\log(1 - D(G(z)))\big]\Big\} \tag{1}$$

where $p_{data}(x)$ is the distribution of the real data and $p_z(z)$ is the distribution of the random noise vector z. The first term in the objective function maximizes the probability of the discriminator correctly identifying real data, while the second term maximizes the probability of the discriminator incorrectly identifying synthetic data as real. The generator

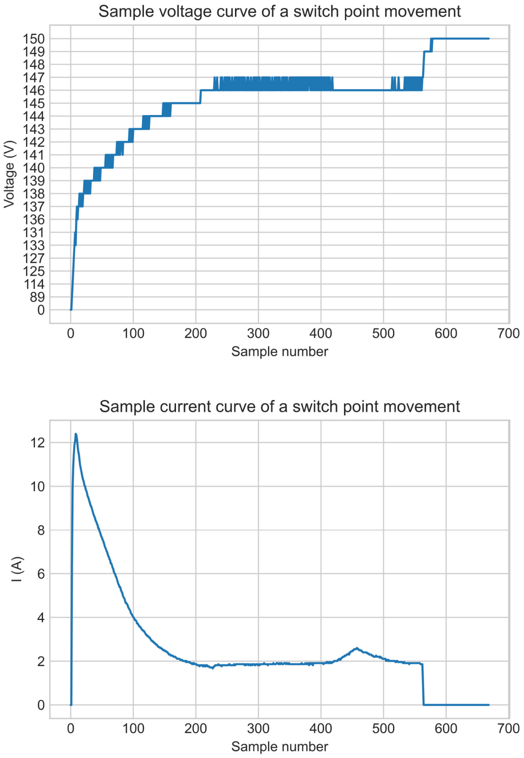


FIGURE 12. **Switch Point features for DT model creation: voltage (up) and supplied Power (down).**

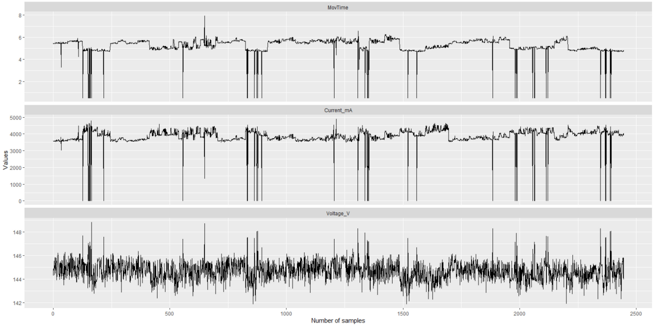


FIGURE 13. **Switch point features of the training dataset.**

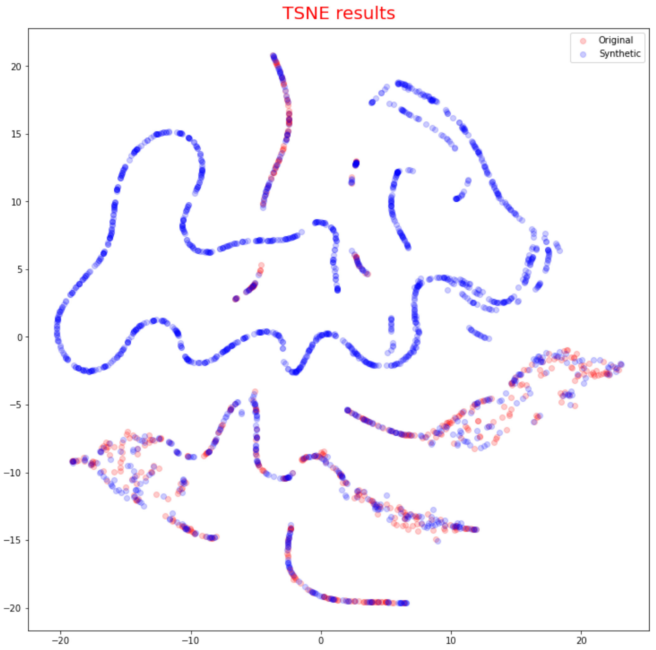


FIGURE 14. **Switch point real and synthetic data comparison.**

tries to minimize the second term by generating synthetic data that can fool the discriminator.

During training, the generator and the discriminator are updated alternately, with the generator updating to maximize the objective function while the discriminator updates to minimize it. The training process continues until the generator is able to produce synthetic data that is indistinguishable from the real data by the discriminator.

We trained a TimeGAN model using raw data from an operational switch point stored in a HIVE table. The training dataset consists of 108,859 samples of the two features collected between January 1st, 2018, and January 1st, 2019. Figure 13 shows an aggregated view of the dataset per day. The data points exhibit various characteristics, such as periodicity, noise level, regularity of time steps, and correlation across time and features.

By performing shift-sampling on our dataset every 669 rows, we obtain 108190 entries, each comprising 669 rows and 2 features. We take a sliding window of size 669 and move it along the rows of the dataset, shifting it by one position at a time to obtain a set of 2D matrices, each with a shape of 669 rows and 2 columns representing the features.

The sampling window size of 669 data points was chosen to match the sampling rate of the sensors that monitor the switch point. Therefore, a sampling window of 669 data points captures a complete movement of the physical object. As a result, we obtained a dataset with dimensions of (108190, (669, 2)), where the 108190 entries have 669 rows (i.e., timesteps) and 3 features. Before model training, we applied data pre-processing to scale the columnar features within a [0, 1] interval using a *MinMax* scaler. We performed model training for 10000 iterations, as suggested by [17].

To evaluate the model's performance, as there are no established objective metrics for evaluating whether a GAN is performing well during training (i.e., reviewing the loss function is not sufficient), we followed an intuitive visual approach [19] to judge the quality in terms of distribution differences between synthetic and real data. To perform a visual comparison between the datasets, we applied the t-distributed Stochastic Neighbor Embedding (t-SNE) algorithm to compare the distribution of the real and synthetic data for visualization in a 2D low-dimensional space, as represented in Figure 14.

The results show an overlap between the real and synthetic data points, suggesting similar behavior between the data collected from a real switch point and their corresponding DT based on synthetic data. Therefore, generating synthetic

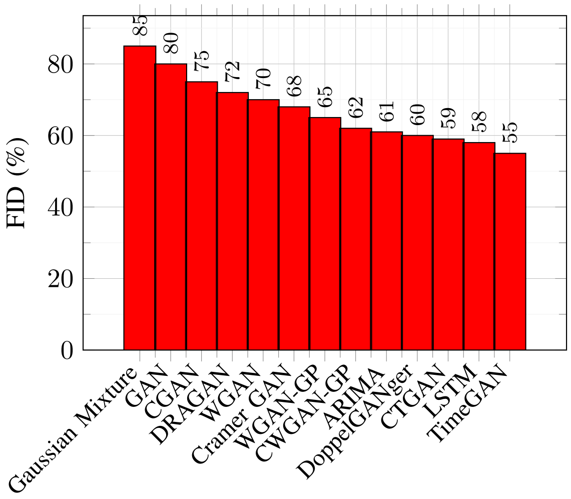


FIGURE 15. FID scores for different models. Lower is better.

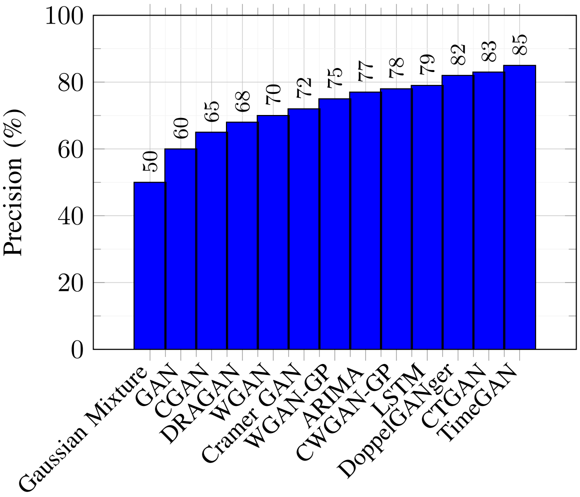


FIGURE 16. Precision scores in percentage. Higher is better.

data to create a Digital Twin model of a physical object is suitable for simulating its behavior as well as enabling analytics in a simulated environment.

Furthermore, we have made a comparative analysis with different models on the same dataset, evaluating ARIMA [20], LSTM [21], GAN, Conditional GAN (CGAN) [22], Wasserstein GAN (WGAN) [23], WGAN with Gradient Penalty (WGAN-GP) [24], DRAGAN [25], Cramer GAN [26], Conditional Wasserstein GAN with Gradient Penalty (CWGAN-GP) [27], Conditional Tabular GAN (CTGAN) [28], Gaussian Mixture Model [29], DoppelGANger [30] and TimeGAN. The evaluation results span several key metrics: Frechet Inception Distance (FID) (see Figure 15), Precision (see Figure 16), Recall (see Figure 16), and Coverage (see Figure 18). These metrics collectively provide insights into the quality, diversity, and fidelity of the generated synthetic data. In details:

1. *FID:* Lower scores are better and indicate that the synthetic data distribution closely matches the real data distribution. TimeGAN, LSTM, CTGAN exhibit the lowest FID scores, making them highly suitable for applications requiring high fidelity, such as railway signal simulation.

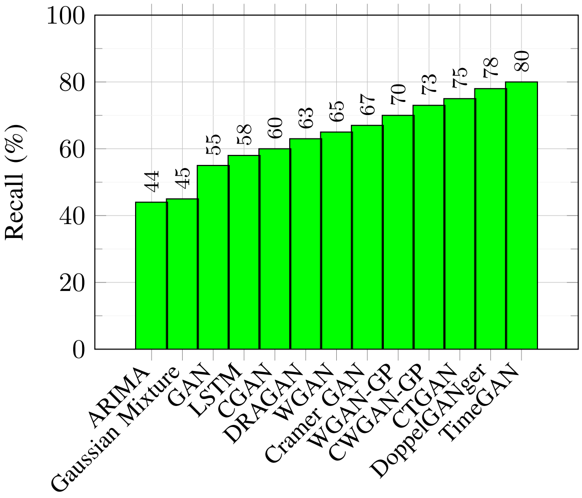


FIGURE 17. Recall scores in percentage. Higher is better.

2. *Precision:* Higher scores indicate a greater ability of the model to generate realistic and varied samples. TimeGAN, CTGAN, and DoppelGANger show the highest precision, making them excellent choices for generating a wide range of realistic signal patterns as in railway simulation.

3. *Recall:* Higher recall suggests that the model can generate a wide variety of data points from the target distribution. Again, TimeGAN and DoppelGANger excel in this aspect which is crucial for capturing the diverse behaviors seen in railway signals.

4. *Coverage:* Higher coverage means the model can generate samples that cover the entire data distribution. CTGAN and CWGAN-GP perform well here, ensuring that rare but critical signal patterns are not missed.

In any case, after the better performances of these two models, TimeGAN model follows.

For the purpose of creating digital twins for railway signal systems, considering the four metrics shown before, TimeGAN surpasses other models for FID, Precision and Recall. For the coverage, CTGAN and CWGAN-GP exhibit a better performance, but they handle categorical variables without taking into account temporal dimension. However, for the goal of this study, we focus on TimeGAN because it focuses on sequential data and it is particularly well-suited for capturing temporal dependencies in railway signal data.

However, based on our experience, training a GAN-based model to generate synthetic data is a time-consuming task. Therefore, the establishment of a Big Data architecture – like the one proposed in this work – is necessary to ingest and process the volume of data and satisfy the constraints of GAN models, which require parallelization to speed up training and simplify DT model creation through deep learning models.

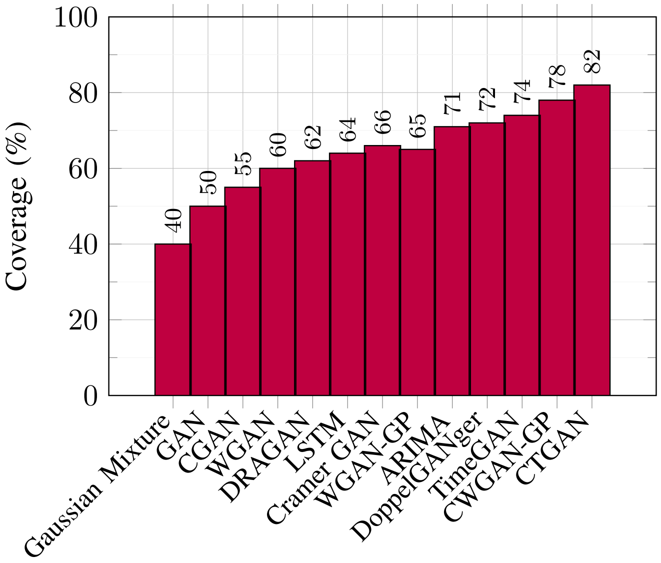


FIGURE 18. Coverage scores in percentage. Higher is better.

Synthetic data generated by a TimeGAN model can be used in DT model creation to complement the raw data and enable the creation of a more comprehensive and accurate model of the physical object. In DT model creation, synthetic data generated by using a TimeGAN model can be used to augment and enrich the available raw data. Even though the raw data may be enough, synthetic data can provide several benefits:

- Increased Data Availability: In some cases, the raw data available may be limited or incomplete. Synthetic data generated using TimeGAN can be used to create a larger dataset that is more representative of the real-world scenario. This can help in better training machine learning models.
- Improved Data Quality: Raw data can contain noise, errors, or missing values, which can affect the performance of machine learning models. Synthetic data generated using TimeGAN can help in filling gaps in the raw data, reducing noise, and improving the overall quality of the data.
- Reduced Data Bias: Raw data can often be biased towards certain patterns or trends. Synthetic data generated using TimeGAN can help in introducing new patterns and trends, reducing the bias in the data.
- Scenario Generation: Synthetic data generated using TimeGAN can be used to create different scenarios or simulations, which can help in testing the robustness and scalability of the DT model.

Therefore, the use of synthetic data generated by using a TimeGAN model can help in creating more robust and accurate DT models, especially when the raw data available is limited, noisy, or biased.

A key novel contribution of this work is the integration of synthetic data generation capabilities within the Digital Twin Layer to enhance representations of real-world physical systems. While the overall procedure of creating virtual representations follows well-established Digital Twin development steps, this work uniquely integrates generative modeling techniques such as Generative Adversarial Networks (GANs) to synthesize simulated data that augments the available real-world data. Specifically, the Digital Twin Layer leverages neural network-based approaches to learn the underlying distributions and relationships in source datasets and generate new representative synthetic data points. This enables a significant expansion of the data available to construct more comprehensive Digital Twin models that incorporate a wider array of potential scenarios (i.e., rare events as failures) beyond the limitations of real historical data. By combining data-driven insights from real system telemetry with the flexibility of artificially generated data, the fidelity and robustness of the resulting Digital Twin models can be improved considerably. The addition of synthetic data generation capabilities demonstrates an innovative way in which simulations of physical systems can become more realistic through prudent blending of real and artificial data within the Digital Twin development process.

In this way, the integration of synthetic data creation using modern generative techniques within the Digital Twin Layer highlights a novel contribution to enhancing virtual representations and generating more informed insights compared to conventional reliance on real-world data alone.

A. DEALING WITH ABNORMAL SCENARIOS

While the TimeGAN model was primarily used to generate synthetic data that complemented the available real-world data, its capabilities may be extended to accurately representing abnormal switch behavior as well. The key benefits of integrating synthetic data generation, as highlighted in the previous section, include increased data availability, improved data quality, and reduced data bias. These characteristics can be particularly valuable when modeling rare or atypical scenarios that may not be well-represented in the historical sensor data.

The paper [17] indicates that the synthetic data generated by the TimeGAN model exhibits similar characteristics to the real-world data, such as periodicity, noise levels, and correlation across time and features. This suggests that the model is able to capture the underlying dynamics and patterns present in the training data, including potential abnormal states of the switch. The visual comparison using t-SNE further reinforces the similarity between the real and synthetic data distributions, implying the model's ability to generate realistic representations of the system's behavior.

By leveraging the synthetic data generation capabilities of the TimeGAN model, the Digital Twin development process can be enhanced to incorporate a broader range of scenarios, including those that may be considered abnormal or rare in the real-world. This can be achieved by ensuring the training data used to fit the TimeGAN model includes examples of abnormal switch behavior, either from historical records or through targeted data collection campaigns. The resulting synthetic data can then be used to augment the real-world data, leading to more comprehensive and robust Digital Twin models that are better equipped to simulate and analyze a wider range of potential operating conditions.

Further investigation into the specific techniques and strategies employed to ensure accurate representation of abnormal switch behavior using the TimeGAN model would be valuable. This could involve quantitative evaluation of the model's ability to generate realistic synthetic data for such scenarios, as well as case studies demonstrating the impact of this approach on the fidelity and predictive capabilities of the resulting Digital Twin models. By addressing this aspect, the integration of synthetic data generation within the Digital Twin development process can be further strengthened, ultimately leading to more accurate and reliable virtual representations of physical systems.

B. CONSIDERATION ON ADOPTION OF SYNTHETIC DATA FOR RAILWAY DIGITAL TWINS

The integration of synthetic data generation using the TimeGAN model within the Digital Twin development process represents a novel contribution to the railway domain. In contrast to conventional approaches that rely solely on real-world sensor data, this method offers several key advantages.

Improved Data Representation: Existing Digital Twin models in railways often face limitations in data availability, quality, and representativeness. Real-world sensor data can be prone to noise, errors, and biases, which can hamper the fidelity of the virtual representations. The ability to generate synthetic data using the TimeGAN model allows for a more comprehensive and nuanced representation of the system's behavior, including rare or abnormal scenarios that may be under represented in the historical data.

Enhanced Flexibility and Scenario Analysis: The synthetic data generated by the TimeGAN model can be used to create a wider range of simulated scenarios, enabling more thorough testing and validation of the Digital Twins. This flexibility is particularly valuable for evaluating the model's performance under various operating conditions, including edge cases and unexpected events, which are crucial for enhancing the robustness and predictive capabilities of the virtual representation.

Reduced Reliance on Scarce Data: In some railway applications, obtaining high-quality, comprehensive real-world data can be challenging due to factors such as sensor failures, environmental conditions, or limited access to operational data. The synthetic data generation capability can help overcome these limitations and reduce the reliance on scarce real-world data, ultimately expanding the range of applications where Digital Twin models can be effectively deployed.

C. POSSIBLE FUTURE INVESTIGATIONS

To further enhance the performance and impact of the proposed approach, the following areas can be examined in future investigations:

Quantitative Evaluation of Synthetic Data Quality: While the visual comparison using t-SNE provides a qualitative assessment of the similarity between real and synthetic data, a more rigorous quantitative evaluation would be beneficial. This could involve statistical metrics, domain-specific performance indicators, or other analytical techniques to objectively assess the fidelity and representational accuracy of the generated synthetic data. Potential metrics to consider include distribution distance measures, feature-wise correlation analysis, and domain-specific validation criteria to ensure the synthetic data accurately captures the essential characteristics of the real-world switch behavior.

Incorporation of Domain-Specific Knowledge: Exploring ways to integrate domain-specific knowledge and expert insights into the synthetic data generation process could further improve the realism and applicability of the generated data. This could involve techniques such as conditional generation, where the TimeGAN model is conditioned on domain-specific parameters or constraints to generate synthetic data that aligns with expert understanding of the system. Additionally, leveraging hybrid approaches that combine data-driven modeling with physics-based models of the railway system may help better capture the underlying dynamics and produce more representative synthetic data.

Exploration of Alternative Generative Models: While this work focuses on the use of the TimeGAN model, investigating the potential of other generative modeling techniques, such as variational autoencoders (VAEs) or normalizing flows, may further enhance the synthetic data generation capabilities and better capture the underlying dynamics of railway systems. These alternative approaches may offer unique advantages in terms of data quality, computational efficiency, or the ability to model complex temporal and multivariate relationships present in railway sensor data.

Integration with Predictive Maintenance and Anomaly Detection: Exploring the synergies between the synthetic data generation approach and predictive maintenance or anomaly detection algorithms could lead to more robust and reliable decision-making capabilities for railway operators. The generated synthetic data could be used to train and validate these algorithms, allowing for more comprehensive testing and evaluation of their performance under a wider range of scenarios, including rare or abnormal events that may be difficult to observe in historical data.

By addressing these areas for improvement and further leveraging the strengths of synthetic data generation within the Digital Twin development process, the presented approach can contribute to more accurate, flexible, and impactful virtual representations of railway systems, ultimately leading to enhanced operational efficiency, safety, and decision-making in the railway domain.

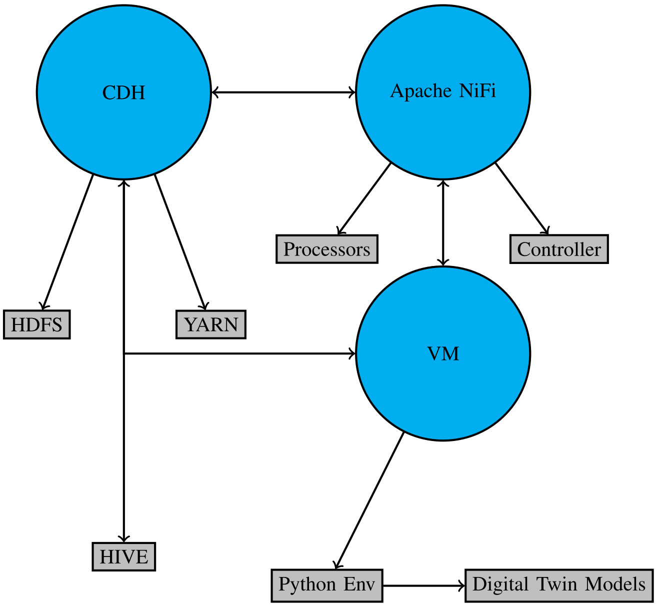


FIGURE 19. Container Architecture with CDH, Apache NiFi, and VM running Python Objects of Digital Twin Models.

VII. ARCHITECTURE DEPLOYMENT

The architecture resulting from our approach described above has been deployed in a test environment using containerization technology (Figure 19).

In particular, we have used two separate containers. In one container, we adopt a Cloudera[10] pre-built image implementing the Hadoop stack (including its fundamental components HDFS, YARN, and HIVE), indicated as CDH (Cloudera Distributed Hadoop) in the Figure 19, which implement the three Storage, Processing and Service Layers, as explained in Section V-A, B, and C. The other container is based on Apache NiFi used to implement the Ingestion Layer together with Processors and the Controller as explained in Section V-E.

These two containers communicate over a virtual network, allowing exchanging data in an isolated environment exposing Web services to access the platforms and perform tasks.

The DT Layer has been implemented by a Python environment that allows the execution of the instances of the classes, described in Section VI, that realized the Digital Twin Models, via a Virtual Machine (VM).

This deployment enables the scalability of the architecture by moving containers on a distributed cluster.

A docker image containing the proposed platform is made available for further testing.[11]

VIII. CONCLUSION

This paper introduces a novel approach to design a Digital Twin (DT) model of a railway switch point. The derived virtual objects are shaped by synthetic data, applying machine learning techniques to raw data collected from physical entities. The proposed Big Data architecture tackles several significant challenges that stem from the sheer volume and complexity of data produced in the railway domain, as well as the requirement to preserve and process raw data from legacy systems.

Through our proposal, we addressed the heterogeneity of data sources in railway systems by enabling the collection of data from diverse objects such as semaphores, switches, signals, and level crossings. Each of these objects produced by various manufacturers presents different data formats and communication protocols, adding complexity to the data acquisition process.

The proposed architecture not only manages the storage of raw data in its unmodified form, a necessity for detailed analysis during system failures, but also handles the transformation of this data for analytics and DT model creation. Dealing with missing or erroneous data, which is not uncommon, is also crucial to ensure the accuracy of the resulting analytical models.

The case study demonstrates that the architecture can efficiently collect, process, and transform raw data into a suitable training dataset for Generative Adversarial Network (GAN)-based DT model creation. The synthetic data generated by the GAN accurately simulate the physical object's behavior, as shown by the overlapping distribution of real and synthetic data as well by models comparison.

It is worth noting that despite existing signal processing techniques for railway systems, DT technology offers unique advantages. Digital Twins provide a virtual representation of the physical asset, thereby allowing improved monitoring, control, and optimization of its performance. Our DT model for railway switch points can enhance existing signal processing techniques, improving overall system performance, and ensuring the safety and reliability of railway systems.

We acknowledge that GAN training is a time-consuming task that might necessitate further research for efficient parallelization and speed optimization, especially in scenarios where real-time data processing is required. Nonetheless, our research presents a promising solution for simulating the behavior of railway switch points, which can also be extended to other domains with similar data characteristics.

REFERENCES

[1] I. Errandonea, S. Beltrán, and S. Arrizabalaga, "Digital twin for maintenance: A literature review," *Comput. Ind.*, vol. 123, Dec. 2020, Art. no. 103316, doi: 10.1016/j.compind.2020.103316.

[2] A. Thaduri, D. Galar, and U. Kumar, "Railway assets: A potential domain for big data analytics," *Procedia Comput. Sci.*, vol. 53, pp. 457–467, 2015, doi: 10.1016/j.procs.2015.07.323.

[3] F. Ghofrani, Q. He, R. M. Goverde, and X. Liu, "Recent applications of big data analytics in railway transportation systems: A survey," *Transp. Res. Part-C, Emerg. Technol.*, vol. 90, pp. 226–246, May 2018, doi: 10.1016/j.trc.2018.03.010.

[4] Q. Xu et al., "A platform for fault diagnosis of high-speed train based on big data," *IFAC-PapersOnLine*, vol. 51, no. 18, pp. 309–314, 2018, doi: 10.1016/j.ifacol.2018.09.318.

[10] https://www.cloudera.com

[11] docker pull julio92sg/data:cloudera-hadoop-nifi

[5] S. Kaewunruen and N. Xu, "Digital twin for sustainability evaluation of railway station buildings," *Front. Built Environ.*, vol. 4, p. 77, Dec. 2018, doi: 10.3389/fbuil.2018.00077.

[6] M. Nold and F. Corman, "How will the railway look like in 2050? A survey of experts on technologies, challenges and opportunities for the railway system," *IEEE Open J. Intell. Transp. Syst.*, vol. 5, pp. 85–102, 2024, doi: 10.1109/OJITS.2023.3346534.

[7] O. Himrane, J. Beugin, and M. Ghazel, "Implementation of a model-oriented approach for supporting safe integration of GNSS-based virtual Balises in ERTMS/ETCS level 3," *IEEE Open J. Intell. Transp. Syst.*, vol. 4, pp. 294–310, 2023, doi: 10.1109/OJITS.2023.3267142.

[8] J. Xun, Y. Li, R. Liu, Y. Li, and Y. Liu, "A survey on control methods for virtual coupling in railway operation," *IEEE Open J. Intell. Transp. Syst.*, vol. 3, pp. 838–855, 2022, doi: 10.1109/OJITS.2022.3228077.

[9] A. Gálvez et al., "Hybrid models and digital twins for condition monitoring: HVAC system for railway," in *Proc. 10th EUROSIM Congr. Model. Simul.*, 2019, p. 52.

[10] R. Tang et al., "A literature review of artificial intelligence applications in railway systems," *Transp. Res. Part-C, Emerg. Technol.*, vol. 140, Jul. 2022, Art. no. 103679, doi: 10.1016/j.trc.2022.103679.

[11] D. Fässler, "Time series models for predicting failure-relevant characteristics of railway switches," Ph.D. dissertation, Dept. Math. Econ., Ulm Univ., Ulm Germany, Jul. 2020.

[12] N. Marz, *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*. Sebastopol, CA, USA: O'Reilly Media, 2013.

[13] H. Fang, "Managing data lakes in big data era: What's a data lake and why has it became popular in data management ecosystem," in *Proc. IEEE Int. Conf. Cyber Technol. Autom., Control, Intell. Syst. (CYBER)*, 2015, pp. 820–824, doi: 10.1109/CYBER.2015.7288049.

[14] T. Kluyver et al., "Jupyter notebooks-a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, F. Loizides and B. Scmidt Eds. Amsterdam, The Netherlands: IOS Press, 2016, pp. 87–90.

[15] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE Access*, vol. 8, pp. 108952–108971, 2020, doi: 10.1109/ACCESS.2020.2998358.

[16] M. Grieves and J. Vickers, "Digital twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems," *Transdisciplinary Perspectives on Complex Systems*. Cham, Switzerland: Springer, 2017, pp. 85–113.

[17] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Advances in Neural Information Processing Systems*, vol. 32, H. Wallach, H. Larochelle, A. Beygelzimer, F. d Alché-Buc, E. Fox, and R. Garnett, Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2019.

[18] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," 2016, *arXiv:1701.00160*.

[19] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen, "Improved techniques for training GANs," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NIPS'16)*, 2016, pp. 2234–2242.

[20] J. D. Hamilton, "Front Matter," in *Time Series Analysis*. Princeton, NJ, USA: Univ. Press, 1994, pp. i–iv. [Online]. Available: http://www.jstor.org/stable/j.ctv14jx6sm.1

[21] S. Hochreiter, J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997, doi: 10.1162/neco.1997.9.8.1735. [Online]. Available: https://doi.org/10.1162/neco.1997.9.8.1735

[22] M. Mirza and S. Osindero, "Conditional generative adversarial nets," 2014, *arXiv:1411.1784*.

[23] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," 2017, *arXiv:1701.07875*.

[24] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville, "Improved training of Wasserstein GANs," in *Advances in Neural Information Processing Systems*, vol. 30, I. Guyon et al., Eds. Red Hook, NY, USA: Curran Assoc., Inc., 2017.

[25] N. Kodali, J. Hays, J. D. Abernethy, and Z. Kira, "On convergence and stability of GANs," 2017, *arXiv:1705.07215*.

[26] M. G. Bellemare et al., "The Cramer distance as a solution to biased Wasserstein gradients," 2017, *arXiv:1705.10743*.

[27] C. Fabbri. "Conditional Wasserstein generative adversarial networks." 2017. [Online]. Available: https://cameronfabbri.github.io/papers/conditionalWGAN.pdf

[28] L. Xu, M. Skoularidou, A. Cuesta-Infante, and K. Veeramachaneni, "Modeling tabular data using conditional GAN," in *Proc. 33rd Conf. Neural Inf. Process. Syst.*, 2019, pp. 1–11.

[29] D. Reynolds, "Gaussian mixture models," in *Encyclopedia of Biometrics*. Boston, MA, USA: Springer, 2009, pp. 659–663, doi: 10.1007/978-0-387-73003-5_196. [Online]. Available: https://doi.org/10.1007/978-0-387-73003-5_196

[30] Z. Lin, A. Jain, C. Wang, G. Fanti, and V. Sekar, "Using GANs for sharing networked time series data: Challenges, initial promise, and open questions," in *Proc. ACM Internet Meas. Conf.*, New York, NY, USA, 2020, pp. 464–483, doi: 10.1145/3419394.3423643. [Online]. Available: https://doi.org/10.1145/3419394.3423643

Open Access funding provided by 'Università degli Studi di Modena e Reggio Emilia' within the CRUI CARE Agreement