Automatic Verification and Validation of Automatically Generated Simulation-Based Digital Twins for Discrete Material Flow Systems

---

Author: Daniel Fischer

Supervisor: Prof. Christian Schwede

Program: Research Master Data Science

Institution: Hochschule Bielefeld (HSBI)

Submission Date: April 21, 2025

Abstract

This thesis addresses the validation of automatically generated Simulation-Based Digital Twins within discrete material flow systems. While learning SBDTs from data promises reduced creation and updating efforts, these benefits are negated if Verification, Validation, and Uncertainty Quantification (VVUQ) require manual expert involvement. To overcome this, a multi-layered, automated VVUQ framework is developed and empirically validated. This framework integrates data processing using Object-Centric Event Logs, twin synchronization, and ML-based validation using a novel supervised classification approach. Results from an IoT Factory case study demonstrate that a Bidirectional Long Short-Term Memory (BiLSTM)-based classifier effectively identifies statistically significant differences between real and simulated process data. Permutation testing confirms these findings across multiple SBDT components (process flow, resource allocation, time models), revealing specific areas where the current SBDT configuration lacked fidelity in the case study. Sanity checks support the framework's validity. The framework enables continuous, automated validation, offering a scalable and objective alternative to periodic manual checks and providing feedback for targeted SBDT improvement. While requiring initial infrastructure setup efforts, the methodology contributes a practical approach for enhancing SBDT reliability and trustworthiness in manufacturing.

Keywords: Simulation-Based Digital Twins (SBDT), Automated Verification, Validation, and Uncertainty Quantification (VVUQ), Data-Driven Validation, Object-Centric Event Log (OCEL), Machine Learning, Supervised Classification, Bidirectional Long Short-Term Memory (BiLSTM), Permutation Testing, Discrete Event Simulation (DES), Discrete Material Flow Systems (DMFS), Manufacturing Systems, Process Mining, Data Quality, Industry 4.0, Internet of Things (IoT).

Contents

List of Tables

List of Figures

List of Abbreviations

4R Representation, Replication, Reality, Relational. 1

Adam Adaptive Moment Estimation. 1
AE Autoencoder. 1
AFS Adaptive Feature Selection. 1
AGDT Automatically Generated Digital Twin. 1
AGV Automated Guided Vehicle. 1
API Application Programming Interface. 1
ASMG Automatic Simulation Model Generation. 1
AUC Area Under the Curve. 1
AutoML Automated Machine Learning. 1

Bi-LSTM Bidirectional Long Short-Term Memory. 1
BNN Bayesian Neural Network. 1
BPM Business Process modelling. 1
BPMN Business Process Model and Notation. 1
BPTT Backpropagation Through Time. 1

CAD Computer-Aided Design. 1
CI Confidence Interval. 1
CI/CD Continuous Integration/Continuous Deployment. 1
CIP Continuous Improvement Process. 1
CNN Convolutional Neural Network. 1
Cosine Cosine Similarity. 1
CPS Cyber-Physical System. 1
CRM Customer Relationship Management. 1

DBSCAN Density-Based Spatial Clustering of Applications with Noise. 1
DDDT Data-Driven Digital Twin. 1
DE Deep Ensembles. 1
DES Discrete-Event Simulation. 1
DL Data Layer. 1
DM Digital Model. 1
DMFS Discrete Material Flow Systems. 1
DNN Deep Neural Network. 1
DS Digital Shadow. 1
DSL Decision Support Layer. 1
DT Digital Twin. 1
DTree Decision Tree. 1

E2O Event-to-Object Relation. 1
EDA Exploratory Data Analysis. 1
EDF Entity Data Format. 1

FN False Negative. 1
FP False Positive. 1
FPR False Positive Rate. 1
FR Functional Requirement. 1

GP Gaussian Processes. 1

HDT Hybrid Digital Twin. 1
HMM Hidden Markov Model. 1

ID Identifier. 1
IF Isolation Forest. 1
IoT Internet of Things. 1
IT/OT Information Technology/Operational Technology. 1

JSON JavaScript Object Notation. 1

KNN K-Nearest Neighbors. 1
KPI Key Performance Indicator. 1

LayerNorm Layer Normalization. 1
LCL Lower Control Limit. 1
LIME Local Interpretable Model-agnostic Explanations. 1
LOF Local Outlier Factor. 1
LSTM Long Short-Term Memory. 1

MAE Mean Absolute Error. 1
MAPE Mean Absolute Percentage Error. 1
MCD Monte Carlo Dropout. 1
ML Machine Learning. 1
MLOps Machine Learning Operations. 1
MQTT Message Queuing Telemetry Transport. 1
MSE Mean Squared Error. 1

NFR Non-Functional Requirement. 1

O2O Object-to-Object Relation. 1
OCED Object-Centric Event Data. 1
OCEL Object-Centric Event Log. 1
OFacT Open Factory Twin. 1
OPC Open Platform Communications. 1
OPC UA Open Platform Communications Unified Architecture. 1
OR Operational Requirement. 1

P-Chart Proportion Chart. 1
PCA Principal Component Analysis. 1
PCB Printed Circuit Board. 1
PlantUML Plant UML. 1
PM Process Mining. 1
PPC Production Planning and Control. 1
PPS Production Planning System. 1

RASS Robotic Assembly Station. 1
ReLU Rectified Linear Unit. 1
ResNet Residual Network. 1
REST Representational State Transfer. 1
RNN Recurrent Neural Network. 1
ROC Receiver Operating Characteristic. 1
RQ Research Question. 1
RR Rejection Rate. 1

SBDT Simulation-Based Digital Twin. 1
SGD Stochastic Gradient Descent. 1
SHAP SHapley Additive exPlanations. 1
sigmoid Sigmoid Function. 1
SLR Systematic Literature Review. 1
softmax Softmax Function. 1
SVM Support Vector Machine. 1

tanh Hyperbolic Tangent. 1
TCP Transmission Control Protocol. 1
TL Twin Layer. 1
TN True Negative. 1
TP True Positive. 1
TPR True Positive Rate. 1
TR Technical Requirement. 1

UCL Upper Control Limit. 1
UI User Interface. 1
UML Unified Modeling Language. 1
UUID Universally Unique Identifier. 1

V&V Verification and Validation. 1
VAP Value Added Process. 1
VBVVQ Value-Based VVUQ. 1
VVUQ Verification, Validation, and Uncertainty Quantification. 1
VVUQL VVUQ Layer. 1

XAI eXplainable Artificial Intelligence. 1
XES eXtensible Event Stream. 1

Chapter 1: Introduction

1.1 Initial Situation

Digital Twins (DT) are a key technology at the forefront of the fourth industrial revolution, often referred to as Industry 4.0. The latter term is characterized by the integration of cyber-physical systems (CPS), the Internet of Things (IoT), and cloud computing to create smart factories aimed at automation and efficiency (**Oztemel2020**). Companies pursue this vision by trying to remain competitive through the adoption of innovative technologies that promise enhanced productivity and reduced operational costs. One such technology that supports this transformation is the DT. It can be defined as a virtual representation of physical assets enabling real-time monitoring and optimization (**Tao2018ijamt**). The DT bridges the connection between the two entities with a bidirectional data flow to exchange information and to influence the behaviour of the physical asset (**grieves2014digital**). This technology is central to Industry 4.0, facilitating the physical and digital worlds through real-time data integration, simulation, and optimization (**judijanto2024trends**).

Although this discipline is rapidly evolving, a unified definition of DT has yet to be established due to the diverse requirements and perspectives across different fields. In engineering, the focus might be on the real-time interaction between physical systems and their digital counterparts, whereas in computer science, the emphasis is often on data integration and simulation capabilities. These varying priorities result in multiple interpretations and applications of the term DT. The concept was first introduced by Michael Grieves in 2002, who defined it as a digital representation of a physical object or system (**grieves2014digital**). However, the concept has evolved since, encompassing a broader range of applications and technologies. In the literature, three terms are used to describe similar characteristics of DT: Digital Model (DM), Digital Shadow (DS), and Digital Twin (DT), see **??** (**jones2020characterising**; **Zhang2021jmsy**).

The Digital Model (DM) represents the most basic form. It involves manual data connections between physical and digital entities. These connections can be temporarily shifted or even disconnected. There is no direct control of the digital object over the physical entity. It is primarily a simple or complex model *describing* the physical object. The data flow must be manually triggered by the modeler, who also interprets the results and controls the DM. The Digital Shadow (DS) is a more advanced version of the DM. It is a digital representation of the physical object that is continuously updated with real-time data, allowing for monitoring, analysis and simulation. While it can predict future states of the physical object based on the current state and historical data, it is not able to influence the physical object without human intervention. The control is, similar to the DM, still in the hands of the modeller. A DS is frequently used for simulation purposes and is sometimes misclassified as a DT in the literature (**kritzinger2018digital**; **sepasgozar2021differentiating**). The Digital Twin (DT)

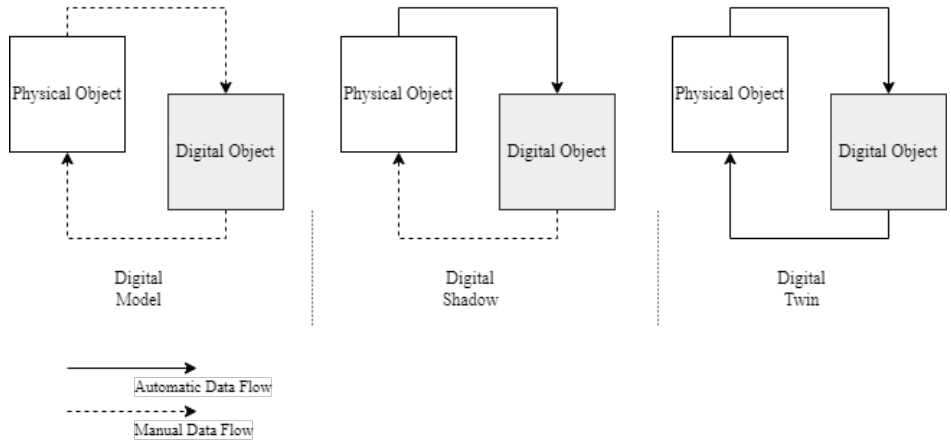

Figure 1.1: Comparison of Digital Shadow (DS), Digital Model (DM) and Digital Twin (DT) as presented by Kritzinger (2018). This distinction is crucial for understanding validation requirements across different digital representation types.

Own illustration based on **kritzinger2018digital<empty citation>** and **Zhang2021jmsy<empty citation>**

is the most advanced version of the three, offering a digital representation of the physical object, which is also continuously updated with real-time data. The DT can be used for monitoring, analysis, and *control* purposes. It can predict the future states of the physical object based on the current state and historical data. The DT can also influence the physical object by sending control signals to it. The control is partially or completely in the hands of the DT. The DT thus *can* serve more purposes than modelling or simulating the physical object. It may serve as an autonomous system, updating itself or with minimal human intervention (**kritzinger2018digital**).

DTs are applied across various sectors, including manufacturing, defence, automotive, service, finance and healthcare (**Tao2018ijamt**). Manufacturing is particularly notable due to its high potential for process optimization and automation. This thesis focuses on the latter, particularly discrete material flow systems (DMFS). These systems process discrete objects (parts) moving along transportation routes or conveyor lines at regular or irregular intervals, integrating both production and logistics operations (**arnold2005materialfluss**; **schwede2024learning**). A key simplification in their modelling is the abstraction of material flow as a sequence of discrete events, following the principles of discrete-event simulation (DES) (**kovacs2016mathematical**; **robinson2014simulation**). DES is well-suited for analysing complex systems where state changes occur at discrete points in time, such as arrivals, departures, and processing steps (**robinson2014simulation**).

Historically, DM played a crucial role in the design, planning, and control of DMFS, primarily through applications like material flow simulations, logistic assistance systems, and digital factory implementations (**Thiede2013**). However, advancements in both DS and DT have enabled a shift from isolated, use-case-specific models toward complete digital representations that span the entire lifecycle of DMFS (**Abdoune2023**). This transition is largely driven by the growing demand for predictive capabilities by stakeholders and automated decision support in manufacturing systems, reflecting the core principles of Industry 4.0

(**frank2019industry**). A second driver of DT innovation lies in the widely available data from IoT devices and sensors, which enhances model training and real-time adaptation of DTs (**Tao2018ijamt**).

In practice, the automated data transfer between the digital model and the physical system is not always critical for DMFS management. Unlike in time-sensitive applications, human decision-makers often remain integral to the control loop, meaning that real-time automation is not always necessary (**schwede2024learning**). Therefore, for this thesis, DS and DTs will be treated as equivalent concepts.

Beyond replicating the current state and managing historical data, DTs are essential for predicting system behaviour and evaluating potential modifications. The widespread use of DES within digital twins highlights the central role of simulation-based DTs (SBDTs) in DMFS (**Lugaresi2021aifac**). As **schwede2024learning<empty citation>** emphasize, SBDTs provide decision support for optimizing costs and performance in highly competitive manufacturing environments. While current SBDTs are primarily developed and updated manually by domain experts, emerging research explores how machine learning (ML) can enhance predictive accuracy and automate model updates by automatically learning model characteristics, reducing costs and development time.

Thus, the progression from digital models to simulation-based DTs reflects an ongoing shift toward data-driven, predictive, and increasingly automated representations of DMFS, enabling more informed decision-making throughout the system's lifecycle (**boschert2016digital**; **lim2020state**).

1.2 Problem

Despite the transformative potential of DTs, their implementation can be challenging. Creating and maintaining accurate DTs require substantial investments in technology and domain knowledge. This investment is wasted if the resulting model fails to accurately represent the physical entity or produces incorrect results. While automatic generation may seem like an elegant solution, it carries risks such as overfitting or biased predictions (**gemanbias**). Manufacturing data for training must be rigorously cleaned and preprocessed. Automatically generated DTs must also undergo automatic Validation, Verification, and Uncertainty Quantification (VVUQ) to preserve their cost and time advantages. Manual VVUQ, which relies on humans in the loop, hinders scalability, automatic synchronization with the physical entity, and depends on costly domain knowledge often provided by experts (**Bitencourt2023**). These hurdles are significant barriers to automatic learning (**ribeiro2016should**; **zhao2024data**). As industries integrate DT into their production processes, establishing trust becomes fundamental as well (**trauer2022digital**; **arrieta2020explainable**). For widespread acceptance among co-workers, stakeholders, and investors, automatic DT creation and VVUQ must demonstrate clear advantages over manual creation and expert-led VVUQ.

Even when DT learning is successfully performed, questions about its correctness, precision, and robustness persist. These concerns are addressed by validation, verification, and uncer-

tainty quantification frameworks (VVUQ) (**sel2025survey**). Ensuring the validity, reliability, and accuracy of a DT is critical, yet traditional VVUQ approaches rely heavily on manual expert involvement and case-specific reference values (**Bitencourt2023**; **hua2022validation**). This leads to inefficiencies, particularly in the context of automated DT generation, where such manual processes undermine the goal of reducing development effort. **hua2022validation<empty citat** even argue that there are no robust and standardized verification and validation (V&V) methods for DTs. As **sel2025survey<empty citation>** point out, uncertainty quantification is often overlooked, but addresses an important aspect of assessing low noise in explanations. One hurdle to standardized VVUQ frameworks is the lack of a clear definitions for validity and verification in the context of DTs (**Bitencourt2023**).

For DMFS, these challenges are even more pressing due to their procedural nature and inherent stochasticity. Rigorous VVUQ is essential to address the risk of manufacturing process failures caused by anomalies, resource constraints, software faults, or human error. This necessity arises because such failures can disrupt the intricate workflows and unpredictable dynamics inherent in DMFS, making reliable performance prediction a priority. When DTs for these systems are generated automatically, traditional validation methods become problematic, as they negate much of the efficiency gains through automation. This creates a fundamental conflict: while automated DT generation reduces initial development and updating efforts, it simultaneously increases the complexity of validation and verification, potentially counteracting its intended efficiency gains.

1.3 Objective

This thesis addresses this conflict by developing a data-driven framework for automated VVUQ of automatically generated, simulation-based DTs that have been learned from data. The focus lies on DMFS due to their practical relevance and dynamical, procedural nature. The research can further be specified by the following research questions (RQ):

- **RQ1:** How can automated validation and verification processes for DTs be efficiently implemented to maintain accuracy?
- **RQ2:** Which data-driven approaches are best suited to identify discrepancies between simulated behaviour and real operational data in discrete material flow systems?
- **RQ3:** To what extent does the developed framework improve the quality and reliability of DTs compared to traditional V&V methods?

This thesis proposes that object-centric event logs—commonly used to generate DTs in manufacturing—can also serve as the foundation for an automated, use-case-independent validation and verification framework. Such an approach would preserve the efficiency benefits of automated generation while ensuring that the resulting DTs meet necessary standards. A key aspect of this approach is the development and monitoring of generic, statistically grounded reference values, which must be quantifiable and have an underlying distribution. The framework will be evaluated using a case study from the discrete material flow domain, providing empirical evidence of its effectiveness in improving model accuracy and efficiency.

1.4 Structure and Methodology

This thesis is organized as follows: **??** establishes the theoretical background on DMFS, SBDTs, Process Mining, and VVUQ. **??** details the development methodology for the automated VVUQ framework, including requirements and the ML-based validation strategy. **??** describes the technical implementation and system architecture. **??** presents the empirical validation using the IoT Factory case study, including main results and sanity checks. Finally, **??** discusses the findings and their implications, while **??** summarizes the key contributions and outlines future research directions.

The thesis follows a Design Science Research approach (DSR). This approach is characterized by the development of artifacts to solve practical problems (**hevner2004design**; **peffers2007design**). Artifacts in the sense of DSR are created objects or constructs which address the given problem and contribute to both theory and practice. The artifacts are evaluated in a real-world context to demonstrate their effectiveness. The thesis applies the cyclical DSR model, see **??**.

The research paradigm of the thesis is deductive-theory critical (**eberhard1987einfuhrung**). A conceptual VVUQ framework is developed based on existing theoretical foundations, while deriving new requirements through a requirements analysis. The framework is then applied in a case study to evaluate its effectiveness. The research is critical in that it aims to improve the efficiency and effectiveness of VVUQ for automatically generated DTs. Elements of empirical research are included through the case study and the data-driven approach.

Chapter 2: Theoretical Foundation

The following chapter provides a theoretical foundation for the research conducted in this thesis. It introduces the basic concepts of material flow planning and simulation, digital twins, process mining, and verification, validation, and uncertainty quantification (VVUQ). The relevance of these concepts in the context of simulation-based digital twins and their application in corporate practice will also be discussed.

2.1 Discrete Material Flow Systems and Simulation

This section begins with an introduction of the underlying concepts of DMFS and Simulation Based Digital Twins (SBDT).

2.1.1 Basic Concepts

DMFS cannot be fully understood without first clarifying the principles of Discrete Event Simulation (DES) for Discrete Event Systems. In DES, a system changes its state through *events* that occur at specific, discrete time instances; it is assumed that no changes occur between two successive events. Consequently, the state of the system is completely defined by the values of its descriptive variables at each event occurrence (**varga2001discrete**). The time at which an event occurs is typically marked by a timestamp, and the scientific observation of such systems is conducted by analysing the discrete *sequence* of events over time (**robinson2014simulation**).

Simulation, in this context, refers to the process of imitating the operation of a Discrete Event System over time, often through multiple event sequences. This imitation is captured in a model, and the core activities in a simulation involve constructing and experimenting with this model. A high-quality simulation abstracts the essential features of the system, which requires the modeller to have a sound *a priori* understanding of what "essential" means in the given context. Although the model can later be refined, its quality is primarily measured by its ability to predict outcomes and offer a diverse range of scenarios (**maria1997introduction**).

In the context of DMFS, their simulation describes the imitation of material flow systems by breaking down continuous flows into discrete events. Such material flow systems can be characterized as "systems processing discrete objects (parts) that move at regular or irregular intervals along transportation routes or conveyor lines, comprising production and logistic systems" (**Arnold2006**; **schwede2024learning**). These systems form the backbone of material flow planning and control structures. The central idea of material flow planning and control is to ensure that material requirements—both in terms of quantity and timing—are met during transportation and storage across the various stages of the supply chain (**Gehr2007**). Importantly, the time horizon of interest spans from order placement up to delivery.

To summarize, DMFS are often simulated using DES, which abstracts the continuous flow

of materials into discrete events. The simulation is carried out using a model. The simulation and modeller are embedded in the context of material flow planning and control, which aims to ensure that material requirements are met across the supply chain. Successfully performed material flow planning and control induce high quality data for simulation and modelling purposes.

2.1.2 Comparing DMFS

Because the simulation of DMFS often involves (discrete) event simulation, events in DMFS need to be further differentiated to be comparable. **Arnold2006<empty citation>** propose to differentiate DMFS into static and dynamic components.

Static components describe the possible states of the system. Possible states can be the set of possible processes given a part or resource, for example. Dynamic components define the concrete material flow for a certain part or order. Static components include parts, resources and processes (**schwede2024learning**). Parts are transformed by processes using resources, sometimes based on orders. Transformation can have an impact on physical properties of the parts (transformation model), spatial position (transition model), the quality of the parts (quality model) and takes time (time model) and uses resources (resource model). Resources have a capacity of handling parts in parallel (resource capacity model) and processes have a predecessor-successors relationship (process model). Dynamic components are used to define the concrete dynamic material flow within the DMFS. There are four components: Order generation, order control, resource control and supply control. Order generation defines the load the system must process. Order control defines how parts are processed, sometimes referred to as routing rules (**mildeautomated**). Resource control defines how resources decide to handle processing requests, also sometimes referred to as priority rules. Supply control describes how supply parts are provided (**mildeautomated**; **schwede2024learning**). See the latter source for a more detailed description of the components.

2.1.3 Production Planning and Control

Successful companies use production planning and control frameworks to describe and optimize their DMFS. After establishing a theoretical foundation and simulation approaches for DMFS, this section thus focuseson Production Planning and Control (PPC) as a critical factor influencing the quality and quantity of data generated by Discrete Event Simulation. PPC is the structured approach to planning, scheduling, controlling and managing all aspects of the manufacturing process. It involves the coordination of resources, processes, and orders to meet production goals. PPC is essential for optimizing production processes, reducing costs, and improving quality. The main functions of PPC include production planning, production scheduling, and production control. Production planning involves determining the production capacity, production goals, and production processes. Production scheduling involves creating a detailed schedule for production activities (**kasper2024designing**). Production control involves monitoring and controlling production activities to ensure that production goals are met (**kiran2019production**). Scheduling is usually the last step performed before execution of the plan (**pinedo2012design**).

The integration of PPC with simulation models is crucial because it directly affects the data quality used in DES of DMFS. Effective PPC processes anticipate anomalies in the production cycle, allowing for adjustments that maintain system efficiency and reliability. If successful, these adjustments yield high-quality data that enhance the accuracy of simulation outcomes. (**kiran2019production**).

2.1.4 Key Performance Indicators

Up to this point, DES for SBDT of DMFS has been introduced, outlining the key factors that contribute to a robust simulation. A model differentiation framework proposed by **schwede2024learning<en** has been briefly presented to facilitate comparison of SBDT. Furthermore, the critical role of PPC in generating high-quality data for simulation has been discussed. These discussions ignored up until now that, even when SBDT are integrated within well-functioning PPC processes, various SBDT models remain prone to errors and inherent trade-offs that must be addressed by the modeller (**Tao2018ijamt**).

The goal conflict of the modeller when developing SBDT can be described by the following conflict triangle in **??** (**robinson2014simulation**; **balci2012life**).

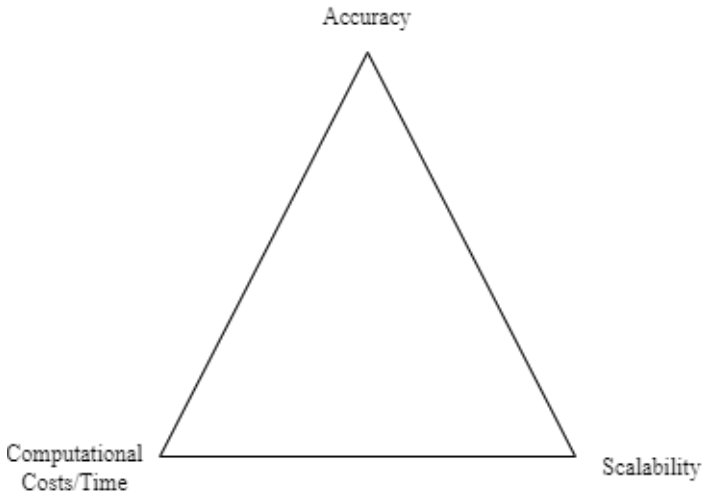

Figure 2.1: The goal conflict of the modeller when developing SBDT. Aiming for higher accuracy often leads to higher computational costs and reduced scalability. Reduced computational cost often leads to reduced accuracy and scalability. Aiming for higher scalability often leads to reduced accuracy and higher costs.

Own illustration based on (**robinson2014simulation**; **balci2012life**).

Focusing one of the three dimensions often leads to trade-offs in the other two dimensions. Oftentimes the data itself is not sufficient to make a decision on which trade-off to make. Limited data points may hinder the modeller from reaching high validity. System architecture may block the system from reaching good scalability. Hardware limitations may hinder the modeller from reaching high efficiency. At other times, corporate management may have a preference for one of the dimensions.

One solution to balance and quantify these goals can be achieved by defining a set of KPIs. Some may already be available through PPC, some may be calculated from DES data or the DES itself. Optimally, the data warehouse provides relevant views (**cui2020manufacturing**).

Because the SBDT in theory mirrors the DMFS, the KPIs gathered from PPC and the DES should yield identical values. Deviations between the KPIs of the SBDT and the DMFS may indicate errors in the SBDT or anomalies in the DFMS. The following KPIs are relevant for the evaluation of SBDT:

$$\text{Throughput} = \frac{\text{Number of units produced}}{\text{Time period}} \tag{2.1}$$

$$\text{Lead Time} = \text{End time of the process} - \text{Start time of the process} \tag{2.2}$$

$$\text{Cycle Time} = \frac{\text{Total processing time}}{\text{Number of units produced}} \tag{2.3}$$

$$\text{Total Setup Time} = \sum_{i=1}^{n} \text{Setup Time}_i \tag{2.4}$$

The PPC related KPIs may be provided by the above-mentioned data warehouse, because they are highly relevant in the context of production scheduling and control. *Throughput*, as defined in **??**, measures the number of produced parts at the last station in a specified period and is an indicator for the productivity of the manufacturing system **hopp2011factory**; **imseitif2019throughput**. *Lead time*, formulated in **??**, is the cumulative time a part travels through the system from start to finish and serves as an indicator for the efficiency of the manufacturing system **slack2010operations**; **pfeiffer2016manufacturing**. *Cycle time*, shown in **??**, measures the same duration as lead time but focuses only on the active production process, excluding transports and waiting times **goldratt2004goal**; **griffin1993metrics**. *Setup time*, as expressed in **??**, measures the time needed to prepare a machine for a new task and is an indicator for the flexibility of the manufacturing system **allahverdi1999review**; **allahverdi2008significance**. In the given use case, the thesis aggregates the setup time for all setup processes. All KPIs presented so far can be calculated dynamically when new data has been sent. Later on, they may serve as an alert system for the modeller to detect deviations between the SBDT and the DMFS, see **??**.

2.2 Digital Twin: Definition and Concepts

The latter section gave a short introduction into DFMS, DES, its metrics and the corporate processes accompanying the SBDT. Now, the thesis sheds light on the DT itself. For a short introduction to the topic, see **??**.

Like introduced in the preceding chapter, DT inherit the highest order of modelling fidelity compared to DM or DS. There are different definitions of DT present in the literature (**Negri2017promfg**; **zheng2019application**; **glaessgen2012digital**; **Demkovich2018def**; **boschert2016digital**; **grieves2014digital**; **kritzinger2018digital**; **Tao2018ijamt**; **zehnder2018representing**). Each of them highlights different aspects of the DT. This thesis utilizes the definition by **grieves2014digital<empty**

which highlights the conceptual elements of the twin and its lifecycle focus:

> The digital twin concept (...) contains three main parts: A) Physical products in real space, (B) virtual products in virtual space and (C) the two-way connections of data and information that tie the virtual and real products together. **grieves2014digital<empty citation>**

The physical product is the entity which will be modelled. The virtual product is the DT itself, but also its infrastructure, for example data services making the real-time data flow possible (**Tao2018ijamt**). The two-way connection is the data flow between the physical and the virtual product. The data flow is bidirectional. **zehnder2018representing<empty citation>** add that the data flow may contain meta data "describing the data source and its context'.

Types of Digital Twins

Now that a unified understanding of DT has been established, this section focuses on how DT may be learned from different sources of information. The following list includes the most relevant types of DT:

- Simulation-based DT (SBDT) (**Lugaresi2021aifac**; **martinez2018automatic**)
- Data-driven DT (DDDT) (**he2019data**; **Friederich2022**)
- Hybrid Digital Twins (HDT) (**luo2020hybrid**; **huang2023hybrid**)

SBDTs (**Lugaresi2021aifac**; **martinez2018automatic**; **boschert2016digital**) are based on DES. They utilize discrete event simulation (see **??**) to create a dynamic representation of the physical system (**schluse2016simulation**; **pantelides2013online**). To incorporate a SBDT into workflows and processes, suitable data structures must be in place beforehand (**boschert2016digital**). DES may improve the predictive capabilities of the model compared to manual twin creation. DES is able to model causal relationships between the events (**francis2021towards**). In contrast, the development of a realistic simulation model requires experts and time (**Charpentier2014**). If the simulation model fails to capture recent behaviour of the physical entity, a recalibration is mandatory (**Friederich2022**). SBDTs are a step forward to speed up the creation and updating processes of DTs.

DDDT rely on the utilization of data to model the physical entity. The data may be gathered from sensors, data warehouses or other sources (later on developed framework summarizes this under the term data sources, see **??**). The data is used to train a model which represents the physical entity. The model may be a neural network, a decision tree or another machine learning model. The model is then used to predict future states of the physical entity. The model may be updated with new data to increase its accuracy (**he2019data**; **Friederich2022**). For a more detailed description of DDDT including its up- and downsides, see **??**.

HDT combine different sources of information to create a more accurate model of the physical entity. The sources may be simulation models (see **??**), data-driven models (see **??**) or physics-based models. Physics-based models contain information about the physical properties and behaviours of the entity. They do not have to learn these characteristics from the data because this information is made available to the model *a priori* (**kapteyn2022data**;

aivaliotis2019methodology). The simulation based models accompanying the physics-based one obeys characteristics of SBDT, see above. The combination of different sources may make the HDT more robust and a faster learner. HDT unite the advantages of SBDT with the knowledge advantage physics based models have. Unfortunately, they also inherit the disadvantages of SBDT through their simulation character. Physics-based models may also involve heavy computational costs and domain expertise (**kapteyn2022data**).

Data-Driven Digital Twins

While SBDTs and HDT possess significant computational costs and require domain expertise, DDDT are able to learn from data without the need for a hand-written simulation model. The DDDT *learns* the model. Learning in the context of DDDT is not trivial, several approaches have been proposed in the literature (**he2019data**; **Friederich2022**; **francis2021towards**). Oftentimes Data Science methods come to work. The learning process may be supervised or unsupervised. Supervised learning uses labelled data to train the model (**cunningham2008supervised**). The label can symbolize different values of interest. Unsupervised learning uses unlabelled data to train the model (**barlow1989unsupervised**). Oftentimes, the task at hand is to group the data into different categories, see **??** (**Biesinger2019**). The learning process may be online or offline. Offline learning uses the data *once* for training, validation and testing, while online learning continuously updates the model with new data to adapt to changes in the physical system. Online learning is thus able to capture new trends in the data and to foresee concept drift (**tsymbal2004problem**). DDDT have to be differentiated from data-driven simulation (**Charpentier2014**), which involves human intervention to create highly individual solutions for the physical entity. The key difference is that every characteristic has to be explicitly described in the model by the expert, there are no efforts to let an intelligent algorithm learn these by itself. DDDT may be able to update themselves to new trends in the data by online learning, termed *synchronization* (**reinhardt2019survey**). Latter has to be differentiated from *updating*, which is a manual process to take corrective action in the logic of the twin itself (**schwede2024learning**). An example for updating a DDDT may be the addition of a new feature to the model. An example for synchronization may be the adaption of the model to new trends in the data. The latter may be done by the model itself, the former has to be done by the modeller. DDDT thus rely less on domain expertise and manual model creation. A suitable model may be able to capture relevant trends in the data and to predict outcomes which describe most of the characteristics of the physical entity. **francis2021towards<empty citation>** propose several process steps a DDDT must undergo to be termed *data-driven*:

1. **Data Collection:** The relevant entities to be modelled have to be identified. This activity involves data gathering of the identified entities and ensuring a steady data stream to a database. The data may be gathered from sensors, data warehouses or other sources.
2. **Data Validation:** This step involves cleaning and preprocessing the data. The data may contain missing values, outliers or other errors. The data has to be cleaned and preprocessed to ensure a high quality of the model. Plausibility checks may be per-

formed to ensure the data is correct.

3. **Knowledge Extraction:** After the data has been collected and cleaned, events have to be detected. **francis2021towards** utilize PM terms in this context, such as event detection and process discovery. The main goal in this step is to find a common ground on which events are of interest. The thesis later dives deeper into PM techniques applied here, see **??**.
4. **(Semi-)automatic Simulation Modelling:** The data is used to train a model. The model is then used to predict future states of the physical entity. The model may be updated with new data to increase its accuracy.
5. **Continuous Model Validation:** Interestingly, **francis2021towards<empty citation>** propose a continuous model validation. In the online learning case, they recommend to use the steady data stream to apply validation techniques continuously, see **??**. The validation may be performed by comparing the model predictions with the real data. If the model deviates from the real data, the model may be recalibrated.

DDDTs go one step further than SBDT and minimize the influence of the human in the loop (**francis2021towards**; **Friederich2022**). Faster model development and updating activities are the result. The third reason to automate DT endeavours elaborated by **schwede2024learning<empty cita** increasing prediction quality, rises and falls with the data quality, thus the gathering and preprocessing efforts of the modeller. Extrinsic factors like the number of data points available also play into the equation. If the number of features is greater than the number of samples, the curse of dimensionality hinders a good modelling performance (**koppen2000curse**). DDDT should avoid biased or noisy predictions at all costs. The identification of *relevant* events poses the risk of introducing a selection bias, rather a confirmation bias. The modeller may have the tendency to select events which complement his hypothesis. Random sampling may be a solution to this problem, but can destroy sequential information patterns in event sequences.

Overall DDDT are a promising approach to model the physical entity. If the right balance between human involvement and automated learning is found, it may be an efficient solution (**francis2021towards**). Thinking one step ahead, employing data-based VVUQ approaches may also be a step forward. This topic will be discussed in Section **??**.

One last discipline, automatic simulation model generation (ASMG), is worth mentioning. ASMG has to be differentiated from DDDT by the effort to automatically generate models, not twins. DM and DS are the goal here, achieved with tools of DES. Automatic DT generation is not necessarily the goal. It aims to automate the model generation process and tries to eliminate the human in the loop, (**reinhardt2019survey**; **lechevalier2018methodology**). Automation is achieved by taking into account a diverse range of data sources, including Computer Aided Design data, PPS data, production manuals, process data and programming code, thus reaching a high data variability. The gathered data has to be processed online or offline as well through suitable frameworks or human intervention. Challenges lay in incomplete data (**bergmann2014automatische**), although the same problems of DDDT also apply here. If the gained data is not mined thoroughly, human intervention is needed again,

mitigating automation efforts.

To conclude this section about DT, the thesis summarizes that there are different types of DT differentiated by their source of information retrieval. A lot of work has been done to make the DT creation, updating and prediction process more efficient. By the help of simulation, data and automated model generation, the DT may be created with less time and resources than manually.

2.3 Process Mining and Event Logs

After introducing the corporate embedding of DTs and their types, the thesis now focuseson process mining (PM) and event logs. PM is a discipline which aims to extract knowledge from event logs. Event logs are the data basis for PM. The following section introduces the basic concepts of PM and event logs.

2.3.1 Core Concepts

PM is a discipline established 1999 which is interdisciplinary rooted in the field of Data Science and Process Science (**van2016data**). Data Science can be considered a process agnostic discipline (**van2016data**) while process science uses models not covering hidden trends in the data. The bridge between both approaches is PM. The goal of PM is to use event data to identify and extract process information (**vanderAalst2012**). This information is used to discover (process discovery), monitor (conformance checking) and improve processes (process enhancement) (**vanderAalst2012**) by using event logs. Such logs must contain a case ID, an activity name and a timestamp. Additional information like resource information, order information or other context information may be added to the log (**vanderAalst2012**). Such logs assume that the process can be captured fully and sequentially.

?? illustrates the PM concepts. The case ID groups unique events which are identified by an event ID to one group, a trace. The timestamp refers to the time of event occurrence, while the activity describes the event. In this example, additional information like resource name and cost are given as well. Cases containing the same events identified by unique event IDs will have different case IDs (**van2016data**). Process discovery may try to produce a process model from the event log. The model may be a Petri net, a BPMN model or another process model. The challenge lies not in the recording of every trace present in the event log, rather in finding a generic representation of the most occurring traces. The process model must be generic enough to describe most traces, but specific enough to not get invalidated by future traces which may contain completely different events. Another major building block of this process model is accounting for trace concurrency. When several events may be identified to happen in parallel during the same time window, the model must recognize this. It can be spoken of a classical bias-variance trade-off lend from data science (**briscoe2011conceptual**). The process must contain the most frequent traces but has to filter out traces which contain anomalies. Such anomalies like longer time per event due to a fire alarm have to be accounted for. Conformance checking may compare a given process model against a given event log. They are specialized in detecting aforementioned anomalies. A key insight in conformance

Table 2.1: A fragment of a manufacturing event log: Each line corresponds to an event. The case ID groups unique events which are identified by an event ID to one group, a trace. The timestamp refers to the time of event occurrence, while the activity describes the event. In this example, additional information like resource name and cost are given as well. Case 1, for example, consists of five events, involving a warehouse, two inspectors, and one machine.

Case id	Event id	Timestamp	Activity	Resource	Cost
1	101	10-01-2025:08.00	receive raw material	Warehouse A	500
	102	10-01-2025:08.30	initial quality check	Inspector Stefan	300
	103	10-01-2025:09.00	cutting process	Machine X	800
	104	10-01-2025:09.45	assembly	Worker Paul	600
	105	10-01-2025:10.30	final inspection	Inspector Eva	400
2	201	11-01-2025:07.45	receive raw material	Warehouse B	500
	202	11-01-2025:08.15	cutting process	Machine Y	800
	203	11-01-2025:09.00	welding	Robot Arm Z	700
	204	11-01-2025:09.45	quality assurance	Inspector David	400
3	301	12-01-2025:06.30	receive raw material	Warehouse C	500
	302	12-01-2025:07.00	initial quality check	Inspector Claudius	300
	303	12-01-2025:07.30	CNC machining	Machine W	900
	304	12-01-2025:08.15	painting	Worker Daniel	500
	305	12-01-2025:09.00	packaging	Worker Johannes	350

Own illustration based on (**van2016data**).

checking lies in two possible deviations from reality: The given model does not capture the real behaviour (A) or reality differs from the model (B). In the first case, the model is not working as intended. In the second case, the event log is corrupt. The third view on event logs, process enhancement, enables the modeller to use the generated process model to identify bottlenecks. Anomalies identified serve as a good starting point because they reveal errors in the process sequence. The given table offers costs associated to each event ID. This information may be used to create an event benchmark to further optimize the desired 'ideal' trace. The first goal of course is to ensure that no mistakes happen during a process.

More generally, PM empowers the modeller to perform VVUQ of an process model, event log or described trace. This concept is captured by the terms *Play-In*, *Play-Out* and *Replay* (**damm2001lscs**), see **??**.

Play-In refers to the creation of a process model out of an event log. Play-Out may be called 'sampling' in data science as it generates traces out of the model. DES uses Play-Out to generate new exemplary behaviour, see **??**. Here, a process model may be used to play-out (simulate) several traces of the desired events and then use bagging techniques like averaging the durations per event to gain robust KPIs (**??**) and to reduce variance. A biased process model may still generate biased KPIs. Replay uses event log and process models together to check conformance, enriching the model with information captured in the log or to transform a descriptive process model into a prescriptive one (**van2016data**).

PM uses different formats to display process models like petri nets and BPMN models, among

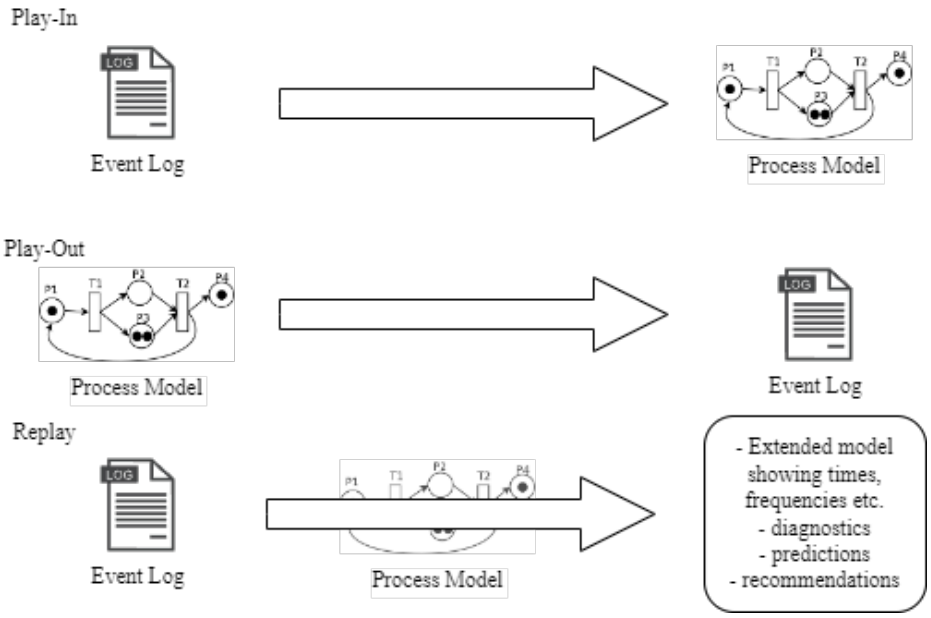

Figure 2.2: The Play-In, Play-Out and Replay concept in the context of PM. The Play-In phase involves the creation of a process model from an event log. The Play-Out phase involves the creation of an event log from a process model. The Replay phase involves the modification of the process model thorough information gained from the event log.

Own illustration based on (**damm2001lscs**).

others (**vanderAalst2012**). The thesis at hand focuseson DDDT in the context of simulation. Such data-driven models often involve complex inference steps not visible to the modeller, termed opaque **??**. Thus, they can not be rendered by PM models which are transparent.

2.3.2 Object-Centricness in Process Mining

The problem with traditional PM lies in its single dimensionality of perspective. For each process analysis, a new play-out has to be performed. Interactions between different objects are not captured (**van2023object**). One event may be related to different cases (convergence) or from the perspective of a case, there may be several equal activities within a case (divergence) (**van2019object**). Recently, object-centric event data (OCED) have been proposed as a new data basis for PM (**van2019object**). OCED logs (OCEL) are a generalization of classical event logs. Such traditional event logs use the case ID as a label to group and distinguish events. They assume that the process model describes the start and end of a single object. Each event refers to exactly one object (case) (**van2023object**). OCED overthrows these assumptions by assuming that events may relate to multiple objects. To account for this new logic, OCEL coins the terms events, objects, types, attributes and qualifier. Each object has exactly one object type. Several objects may have the same type. An object type can be a description of the function such as machine, customer, supplier or activity descriptions such as invoice, request. Objects are instances of these types. Events in particular have an event type, termed activity. The same non-uniqueness applies here—many events can have the same type like “processing complaint’, “cooking coffee’. Each event is described by exactly one type. OCED assumes event atomicity; each event is indivisible. Each event has one timestamp. Compared to traditional PM, events may relate to multiple objects through a qualifier (event to order, E2O). Such a qualifier may be, considering the event “printing

label' and object "printing station', the label to be printed. Objects may be related to multiple objects (order to order, O2O). O2O relationships are frozen (static) and are assumed to not change. The O2O relationship may be used to describe the order of producing a product. For example, the O2O relation "main pcb to gyroscope' may say that the main pcb has to be produced before the gyroscope. Another O2O relation can be an order, the connection between the customer object and the ordered product. It is worth mentioning that objects can also be related indirectly together through two E2O relations: The E2O relation 'producing' may connect the event "machine 1 produces' with the object 'gyroscope'. The E2O relation 'check' may connect the event "machine 1 checks' with the object "main pcb', thus connecting the two objects 'gyroscope' and "main pcb' indirectly via two events (**van2019object**). E2O relations are dynamic. They may change over time, involving different objects. In the given example, "main pcb' would be checked with 'display' instead of 'gyroscope'.

Events and objects have attributes (keys), possessing values. Event attribute values refer to exactly one event and one event attribute; they are not shared. For example, the cost for one event may have the value $10€$. The same logic applies to objects as well, one event attribute value refers to exactly one object and one object attribute. Because several events may have the same type and several objects may have the same type as well, each event or object type may refer to any number of event or object attributes. They open interpretative possibilities for the modeller by considering them as expected attributes for the event or object type. The given example may be the event type 'producing' which may have the event attribute 'duration' and 'cost'. The object type 'gyroscope' may have the object attribute 'weight' and 'size'. One may average the different object attribute values of one type cluster to generate object type KPIs. A key specificity of object attribute values lies in the fact that they have a timestamp. Event attribute values do not. Latter information would be redundant because events do already have a timestamp, see above. Event attribute values may have cardinality one per event. N event attributes have n values. This does not apply to object attribute values. They may have multiple values because of their nature to change over time (**van2023object**). This is why each object attribute value has a timestamp. The attribute value is in conclusion unique for a given object during a given time given one attribute. The given example may be the object attribute 'weight' of the object "main pcb' which may change over time. The object attribute value 'weight' may have the value $100g$ at time t_1 and $120g$ at time t_2.

OCEL thus extends traditional PM by accommodating the multi-object nature of complex processes. Unlike classical event logs—which restrict events to a single case—OCEL captures dynamic interactions among multiple objects via E2O relations and static inter-object dependencies through O2O relations. This enriched framework enables a more comprehensive representation of real-world processes, where events may concurrently affect several objects. Timestamped object attribution values enable the modeller to perform temporal analysis and KPI derivation (**??**). Overall, OCED provides a more detailed, semantically rich representation of complex, interconnected processes. (**van2023object**).

2.3.3 Process Mining as Enabling Technology

PM uses event logs to develop process models or to enhance existing ones, so these logs may serve as a foundation for VVUQ of models in general. Live twin data often can be exported to the event log format. Several standardizations have been proposed, with the IEEE XES standard being the most widely used (**van2016data**). The XES standard defines a common format for event logs, enabling the exchange of event data between different software frameworks. The idea lies in exporting twin decisions or live data as event logs and then use PM tools to perform VVUQ. Replay may be used on SBDT simulated traces in comparison with actual event data to reveal mismatches in sequences or timing. For example, the SBDT could have predicted a circular process. Replay can be applied to further analyse this bottleneck. Play-Out can empower the modeller to sample a big amount of exemplary traces to gain KPIs. OCED object attribute values may offer even more insights. PPC systems (**??**) often times deliver even more data to enrich the event log so that VVUQ can be performed easier. If event log extraction out of the SBDT is performed online, VVUQ can be performed on the fly. Both sources of information are incorporated in the framework, see **??**.

2.4 VVUQ in the Context of Simulation-Based Digital Twins

The previous sections introduced the concepts of DES, PPC, relevant KPIs, DT, PM and OCED. This section now focuseson VVUQ in the context of SBDT. The thesis at hand uses the term VVUQ to describe the process of verifying, validating and quantifying the uncertainty of a SBDT. V&V has a long history in manufacturing and DES (**Bitencourt2023**). **sel2025survey<empty citation>** add uncertainty quantification as a main interest. Their framework is applied in the medical domain, but they mention reasonable arguments regarding efficiency and safety of SBDT in general.[1] Thus, the thesis considers VVUQ instead of merely V&V efforts. The following section introduces the basic concepts of VVUQ and its relevance for SBDT.

2.4.1 Development process of VVUQ Concepts

With the uprising of simulation models in the early 1950s (**evans1967simulation**), the need for VVUQ arose unknowingly to the modellers. The usability of such simulations was deemed high as long as the results were promising, increasing trust in the technology (**durst2017historical**). Blind trust does not validate models. Contrarily, if the results more or less were satisfactory, the model was considered validated (**bonani2003physics**).

The first effort to define and perform verification was performed by **machlup1955problem<empty citation>** defining verification as “including the correctness of mathematical and logical arguments, the applicability of formulas and equations (...), the reliability and exactness of observations, the reproducibility of experiments, the explanatory or predictive value of generalizations.”. **naylor1967verification<empty citation>** further refined this definitions by introducing the idea of “goodness of fit’. Latter describes the capability of the model to correctly reflect the modelled system. During the 1970s, researchers like (**schlesinger1979terminology**) de-

[1] To go into detail, they deem erroneous data through sensor failure, model opacity (**??**) and the speed of model self-adaption as the necessities for UQ.

fined validation as achieving a "satisfactory range of accuracy consistent with the application', while Ignall argued for validation against simulations rather than analytical models (**ignall1978using**). Sargent's work from 1979 to 1984 proposed methods like user collaboration and independent verification and validation (V&V), detailing techniques such as sensitivity analysis and Turing tests (**Sargent2010wsc**). Balci developed a taxonomy and emphasized continuous V&V, reflecting the need for ongoing assessment (**balci2012life**).

By 2004, modern V&V emerged. Considering model fidelity of todays approaches, **Oberkampf2004amr<e** introduced widely recognized definitions of V&V:

> *Verification* is the process of determining that a model implementation accurately represents the developer's conceptual description of the model and the solution to the model.
>
> *Validation* is the process of determining the degree to which a model is an accurate representation of the real world from the perspective of the intended uses of the model. (**Oberkampf2004amr**)

Verification thus concerns itself with the *correctness* of the given model (**Sargent2010wsc**), while validation evaluates the *quality* of explanations quantitively (**Oberkampf2004amr**). **iso2017systems<empty citation>** span the concept of validation to computational models, where valid models correctly reflect the users intended functionality. PPC may provide the modeller with relevant KPIs to assess both.

Uncertainty quantification (UQ) is a relatively new field in V&V. It aims to quantify the uncertainty of the model and its predictions through the whole lifecycle of the model, including training, inference and prediction (**sel2025survey**). UQ empowers the modeller to define confidence intervals to correctly emphasize the stochastic nature of the model predictions (**volodina2021importance**). This is crucial for SBDT, where real-time decisions rely on accurate representations, as **francis2021towards<empty citation>** highlights the need for continuous updates. UQ differentiates *aleatory* uncertainty, which is inherent to the data, and *epistemic* uncertainty, which is due to lack of knowledge (**sel2025survey**). Aleatory uncertainty may arise due to sensor errors in the physical entity, generating noise. Epistemic uncertainty may arise due to lack of data or knowledge about the physical entity (**thelen2023comprehensive**). Epistemic uncertainty can be reduced. **abdoune2022handling<empty citatio** list UQ challenges and potential solutions in the context of SBDT.

For SBDT, VVUQ is not a one-time activity but rather a continuous process (**??**), ensuring DT mirror physical systems accurately. This is important for industries like manufacturing, where decisions based on DT have significant implications **??**. The historical development of VVUQ, from early verification to integrated UQ, reflects the growing complexity of simulation models. As SBDT become central to decision-making, robust VVUQ practices ensure reliability, linking back to foundational concepts introduced earlier, see **??**. The concepts of VVUQ are embedded in the context of PM and OCED, which may further assist described endeavours. Especially for UQ, PM offers a rich data source to quantify uncertainty and

validate models by providing a *degree of fit* of the given process model.

2.4.2 VVUQ for Automatically Generated Models

Automatically generated models largely stem from the DES discipline (**??**), termed automatic simulation model generation, ASMG (**mildeautomated**; **Charpentier2014**). They may use data-driven techniques which reduce the manual effort to create and update themselves. These models adapt quickly to changing conditions, making them valuable for dynamic environments like manufacturing. However, ensuring their reliability requires specific VVUQ. Because ASMG models often are black boxes, traditional VVUQ methods may not be applicable. The challenge lies in understanding the models internal logic and ensuring it accurately reflects the physical system. If sophisticated data driven methods have been applied, several *key challenges* arise, hindering successful VVUQ:

- **Model Opacity:** ASMG models often use complex machine learning algorithms like neural networks. Such black-box models are difficult to interpret, making it hard to understand their internal logic. This opacity may hinder VVUQ, as the modeller cannot easily identify errors or biases.
- **Data Dependency:** Data-driven models rely on high-quality data. If the data is biased or noisy, the models predictions may be inaccurate.
- **Dynamic Model Adaptation:** Models that continuously learn and adapt, such as those using online learning, require ongoing validation to ensure they remain valid as new data is ingested. This dynamic nature introduces the risk of concept drift (**lu2018learning**), where the underlying process changes over time, potentially degrading model performance.
- **Quantifying Uncertainty:** Model predictions are stochastic in nature. For applications where precision is crucial, such as in manufacturing, uncertainty needs to be quantified.

To assess these challenges, the thesis defines the following *key requirements* for VVUQ of automatically generated models, especially SBDT:

- **Model Interpretability:** Models should be interpretable to ensure the models internal logic is transparent and understandable, enabling the modeller to identify errors or biases. Developing and employing techniques to make the decision-making processes of automatically generated models more transparent is crucial for VVUQ.
- **Upholding Data Quality:** Procedures to ensure that the data used for model generation and validation is accurate, complete, and representative of the operational environment are important. This includes data cleaning, preprocessing, and plausibility checks to identify and mitigate issues like missing values, outliers, or biases. For instance, in manufacturing digital twins, sensor data must be validated for accuracy and consistency to ensure reliable model outputs (**rodriguez2023updating**).
- **Validatable Algorithms:** Besides ensuring model interpretability, the algorithms used for model generation must be validatable.
- **Continuous Validation:** VVUQ processes have to work in real-time to ensure the

model remains valid as new data is ingested. This requires continuous monitoring and validation of the models predictions against real-world data. Techniques like online validation (**francis2021towards**) can help ensure the models accuracy and reliability over time.

- **Integration:** VVUQ processes have to be integrated into existing model- and PPC infrastructure to be able to perform VVUQ on the fly. This requires close collaboration between data scientists, domain experts, and IT specialists to ensure the integration of VVUQ processes into the model lifecycle.
- **Scalability:** VVUQ processes have to be scalable as the underlying model or data evolves over time.

As noted by **francis2021towards<empty citation>**, the lifecycle SBDT has implications for its VVUQ as well: VVUQ must accompany the SBDT in all phases, from conceptualization to deployment and operation. The key requirements outlined above provide a foundation for developing robust VVUQ processes for automatically generated models.

2.4.3 Traditional versus Machine Learning-Based Approaches

VVUQ can be performed using traditional methods or more sophisticated approaches. Traditional verification techniques may include code inspection, unit testing or debugging (**maniaci2018verificati**
If closed solutions to simulated models are available, they may be used to validate the simulation. Traditional validation techniques involve comparisons between model predictions and real-world data, such as statistical tests or sensitivity analysis. In the context of manufacturing, simple experiments or historical data can be used. The consultation of experts is another possibility (**shao2023credibility**).

Machine Learning-Based VVUQ

Sophisticated approaches may include the use of machine learning (ML) techniques to enhance VVUQ processes. ML can be used to identify patterns in data, detect anomalies, and improve model predictions. In addition to supervised and unsupervised learning shortly introduced in **??**, semi-supervised learning and reinforcement learning are two other approaches. Supervised learning may be used where labels regarding the validity or non-validity of an OCEL (see **??**) are given. Unsupervised learning may provide such labels as learned categories if they are missing or may assist with finding common patterns in the data. Unsupervised techniques are context-agnostic and assume no patterns in the data, see **hastie2009unsupervised<empt**
for the reverse problem of encoding a priori information in unsupervised algorithms. Semi-supervised learning combines labelled and unlabelled data to improve model performance. It somehow forms a middle ground where lots of unlabelled data is available and is then grouped by the algorithm. The scarce data is then used in the holdout set to perform testing (**learning2006semi**). Reinforcement learning is not commonly applied in VVUQ of SBDT. ML techniques are often referred to as ‘oracles’ when used for VVUQ because of their key challenge of opacity, **??**.

Challenges in Data Preparation and Feature Selection

Before discussing the application of ML techniques in VVUQ, several problems hindering successful application of ML in VVUQ are identified. The first problem is data quality (**wu2025uncertainty**). Data quality may degrade the performance and reliability of ML-based VVUQ approaches. Firstly, *missing values* in the dataset can hinder the training process and potentially introduce biases. To account for this, imputation strategies or simply removing defect rows may be a solution (**gudivada2017data**). The modeller has to keep in mind that this may corrupt the VVUQ process. Secondly *outliers*, which are data points that deviate significantly from the rest, create a challenge as they might represent real trends in the data (and thus containing valuable edge cases for VVUQ) or simply be erroneous entries, requiring care from the modeller. Thirdly *noise* can superimpose the underlying patterns that ML algorithms aim to learn, reducing the accuracy and generalization ability of the VVUQ models (**liu2020noise**). Such noise may be random, for example by measuring environmental influence, or systematic, for example emitted by erroneous sensors. The first kind of noise can be reduced by smoothing or filtering.

Inconsistent data is another mistake to avoid in data gathering. It may arise from deviations in formats, units or representations of the same information. It can lead to confusion for ML algorithms and require standardization and cleaning (**mahanthappa2021data**). Duplicate data entries may also generate false trends in the data. Imbalance introduced by duplicates or measurement errors can bias the model in predicting only the majority class when a classification problem is at hand. Beyond data quality, the modeller has to make sure that the data is representative and bias-free. The training data used for VVUQ models must accurately reflect the environment. Biases can arise through human intervention or environmental influence (**liu2020noise**). Finally, the way data is split into training, testing, and validation sets is important for avoiding overfitting, where the model memorizes the training data and fails to generalize to unseen data.

After ensuring data quality, the modeller has to consider suitable features for the given problem. Features are columns in the dataset describing characteristics of the data points (rows) through values. OCEL provide a relatively strict feature set in which the modeller has to operate. This has the advantage that several of the upper challenges may be eliminated because the data can be exported through a standardized interface. Successful VVUQ methods may require additional features nonetheless. Only the most relevant features may be selected (**geron2022hands**). Feature engineering describes the endeavour of creating new features through combining existing features or through a new data gathering process. The incorporation of features in model training is coined feature selection. In the given use case, an adaptive feature a feature selection based on twin components may be useful, see **??**.

Classification Methods for the Detection of Model Deviations

Extensive work has been done on the topic of ML-based deviation detection. Such deviations are called ‘anomalies’ and the process is termed “anomaly detection’ (**kharitonov2022comparative**). Anomalies are data points that deviate significantly from the majority of the data. They can be classified into three categories: point anomalies, contextual anomalies and collective anoma-

lies (**chandola2009anomaly**). Point anomalies are single data points that differ significantly from the rest of the dataset. Contextual anomalies are data points that are normal in one context but anomalous in another. They appear less often than point anomalies and have less statistical weight. Collective anomalies are groups of data points that are anomalous when considered together but may not be anomalous individually. Anomalies are of special interest in the context of VVUQ, as they can indicate potential issues with the model or the data, see **??**.

Anomaly detection and VVUQ share common goals of identifying deviations from expected behaviour and ensuring the reliability of models. The two fields can benefit from each other, as VVUQ can provide a framework for evaluating the performance of anomaly detection algorithms, while anomaly detection techniques can enhance VVUQ processes by identifying potential issues in models or data. Several algorithms exist to detect anomalies, including statistical methods, clustering-based methods, and supervised learning methods. If the data is sufficiently labelled and preprocessed, supervised methods are promising. Algorithms such as Support Vector Machines (SVM), Neural Networks (NN), Decision Trees, and Random Forests can learn from this labelled data to classify new model outputs or behaviours as either normal or divergent. However, a common challenge in anomaly detection scenarios is the issue of class imbalance, where instances of normal behaviour are more common than the occurrences of deviations. This imbalance needs to be carefully addressed during model training to prevent the classifier from being biased towards the majority class. Clustering-based unsupervised methods detect anomalies through grouping similar data points together. Algorithms like K-Means, DBSCAN, and Hierarchical Clustering can be used to identify clusters of normal behaviour, with anomalies being those points that do not belong to any cluster. They measure the distance between data points to cull anomalous points as reaching far outside of a group of common data points. Oftentimes the assigned groups can be plotted, yielding further insights in the nature of the detected anomalies. Statistical methods, on the other hand, rely on the assumption that the data follows a certain distribution. They identify anomalies based on statistical properties such as mean, variance, and standard deviation (**chandola2009anomaly**).

Metrics for Model Quality Measurement

Another important aspects of assessing model quality is through metrics. The following section introduces the most common metrics used in VVUQ of SBDT. The introduction is supported by an exemplary OCEL:

Accuracy

Accuracy measures the overall correctness of the classification model. It represents the ratio of correctly classified instances to the total number of instances in the dataset (**fahrmeir2016statistik**).

$$\text{Accuracy} = \frac{\text{Number of Correct Predictions}}{\text{Total Number of Predictions}} = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.5}$$

Table 2.2: A fragment of a manufacturing OCEL with type annotations. Each row represents an event with associated process execution details. The columns are annotated as follows: `process_execution_id` (int), `order_id` (int), `start_time` (datetime), `end_time` (datetime), `part_id` (int), `process_type` (int), `process_id` (int), `resource_id` (int), and `is_valid` (bool).

process_execution_id	order_id	start_time	end_time	part_id	process_type	process_id	resource_id	is_valid
0	2529	2020-04-22 14:21:07	2020-04-22 14:21:31	-1	0	0	0	True
1	2529	2020-04-22 14:23:35	2020-04-22 14:23:58	-1	1	1	1	True
2	2529	2020-04-22 14:21:09	2020-04-22 14:21:33	-1	0	0	0	True
3	2529	2020-04-22 14:23:36	2020-04-22 14:23:58	-1	1	1	1	True
4	2529	2020-04-22 14:21:11	2020-04-22 14:21:35	-1	0	0	0	True
5	2529	2020-04-22 14:23:36	2020-04-22 14:23:58	-1	1	1	1	True
6	2529	2020-04-22 14:21:13	2020-04-22 14:21:37	-1	0	0	0	True
7	2529	2020-04-22 14:23:36	2020-04-22 14:23:58	-1	1	1	1	True
8	2529	2020-04-22 14:21:15	2020-04-22 14:21:39	-1	0	0	0	True
9	2529	2020-04-22 14:23:37	2020-04-22 14:23:58	-1	1	1	1	True
10	2529	2020-04-22 14:21:18	2020-04-22 14:21:41	-1	0	0	0	True
11	2529	2020-04-22 14:23:37	2020-04-22 14:23:57	-1	1	1	1	False
12	2529	2020-04-22 14:21:20	2020-04-22 14:21:44	-1	0	0	0	False
13	2529	2020-04-22 14:23:37	2020-04-22 14:23:57	-1	1	1	1	False
14	2529	2020-04-22 14:21:22	2020-04-22 14:21:46	-1	0	0	0	False
15	2529	2020-04-22 14:23:38	2020-04-22 14:23:57	-1	1	1	1	False

Own tabulation based IoT Factory data.

where TP denotes the count of true positives, TN represents true negatives, FP shows false positives, and FN indicates false negatives.

This metric offers a high-level overview of the models performance by showing the proportion of predictions that align with the actual classes. Accuracy may be a misleading metric when dealing with imbalanced datasets (**fahrmeir2016statistik**). In manufacturing, where the occurrence of invalid processes might be significantly lower than valid ones, a high accuracy score could be achieved by a model that often predicts the majority class, failing to effectively identify the critical minority class of invalid processes. Therefore, relying only on accuracy might not provide a complete or accurate picture of the models utility in identifying anomalies.

Precision

Precision (positive predictive value) focuses on the quality of the positive predictions made by the model. It measures the proportion of instances that the model predicted as positive which were indeed positive.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2.6}$$

A high precision score shows that when the model predicts a manufacturing process as ´valid' (assuming ´´valid' is the positive class), it is highly likely to be truly valid. This is important in manufacturing quality control to minimize the occurrence of false alarms, which can lead to interruptions in the production line and increased operational costs (**kharitonov2022comparative**).

Sensitivity

Recall, also called sensitivity or the true positive rate (TPR), shows the models ability to identify all the actual positive instances within the dataset. It measures the proportion of actual positive instances that were correctly classified as positive by the model (**fahrmeir2016statistik**).

$$\text{Recall} = \frac{TP}{TP + FN} \tag{2.7}$$

In the context of manufacturing, a high recall for the 'valid' class is crucial for ensuring that the model effectively detects the majority of the truly valid processes. Failing to identify a valid process (a false negative) can have consequences, potentially leading to the production of defective goods that may lead to costs related to rework, scrap, or customer dissatisfaction (**kharitonov2022comparative**). Therefore, a model with high recall minimizes the risk of overlooking critical quality issues in the manufacturing process.

F1-Score

The F1-score provides a balanced measure of the classification models performance by calculating the harmonic mean of precision and recall. This metric is particularly useful when handlingdatasets that contain an imbalance in the class distribution.

$$\text{F1-score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = \frac{2TP}{2TP + FP + FN} \tag{2.8}$$

By considering both the precision (the accuracy of positive predictions) and the recall (the ability to find all positive instances), the F1-score offers a single metric that summarizes the trade-off between these two important aspects of a classifier's performance.

Confusion Matrix

A confusion matrix is a specific type of contingency table that provides a detailed breakdown of the performance of a classification model by displaying the counts of true positives, true negatives, false positives, and false negatives. For a binary classification problem, such as predicting whether a manufacturing process is valid or not, the confusion matrix typically has the form of a 2x2 table (**fahrmeir2016statistik**).

Table 2.3: Confusion Matrix for Binary Classification

Actual Class	**Predicted Class**	
	Positive (True)	Negative (False)
Positive (True)	True Positive (TP)	False Negative (FN)
Negative (False)	False Positive (FP)	True Negative (TN)

Own tabulation based on (**fahrmeir2016statistik**).

This matrix offers a view of the models predictive behaviour, allowing for an analysis of the

different types of errors it makes. True positives represent the cases where the model correctly predicted a positive outcome (a valid process was correctly identified). True negatives indicate instances where the model correctly predicted a negative outcome (an invalid process was correctly identified). False positives occur when the model incorrectly predicts a positive outcome for a negative instance (an invalid process was wrongly predicted as valid). False negatives arise when the model incorrectly predicts a negative outcome for a positive instance (a valid process was missed and classified as invalid). Analysing the confusion matrix for the 'is_valid' prediction in the context of manufacturing can reveal crucial information about the models tendencies, such as whether it is more prone to generating false alarms or to missing actual defects, which has direct implications for the design and implementation of quality control strategies.

While the primary focus based on the example data is on classification, the model might also be employed for regression tasks in manufacturing, such as predicting continuous variables like throughput times or resource utilization levels. In such scenarios, the following regression metrics are crucial for evaluating the models predictive accuracy.

Mean Squared Error (MSE)

Mean Squared Error (MSE) shows the average of the squared differences between the values predicted by the model and the actual observed values.

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.9}$$

where y_i represents the actual value of the target variable for the i-th instance, $\hat{y}_i$ is the corresponding predicted value, and n is the total number of data points in the dataset.

MSE provides a measure of the overall prediction error. The squaring of the differences means that larger errors contribute more to the final MSE value, making it sensitive to outliers in the predictions. In the context of manufacturing, if the model were predicting throughput time, a high MSE would indicate that, on average, the squared difference between the predicted and actual throughput times is large, suggesting a lower accuracy in the model's predictions (**fahrmeir2016statistik**).

Mean Absolute Error (MAE)

Mean Absolute Error (MAE) measures the average of the absolute differences between the predicted and actual values.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{2.10}$$

MAE offers a more direct and interpretable measure of the average magnitude of the errors in the models predictions. Unlike MSE, MAE treats all errors equally, without giving dispro-

portionate weight to larger errors (**fahrmeir2016statistik**). In manufacturing applications, such as predicting resource utilization, MAE would represent the average absolute percentage point difference between the models predictions and the actual utilization rates, providing a clear indication of the typical size of the prediction errors.

Mean Absolute Percentage Error (MAPE)

Mean Absolute Percentage Error (MAPE) calculates the average percentage error between the predicted and actual values.

$$\text{MAPE} = \frac{1}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right| \times 100\% \tag{2.11}$$

MAPE is particularly useful when the scale of the target variable varies, as it provides an error measure in relative terms. The percentage error is often easier to understand and communicate than absolute errors, especially in a business or operational context.

Performance Evaluation using ROC Curves and AUC

For binary classification tasks, such as the prediction of 'is_valid' status in manufacturing processes, Receiver Operating Characteristic (ROC) curves and the Area Under the Curve (AUC) provide an evaluation of the models performance.

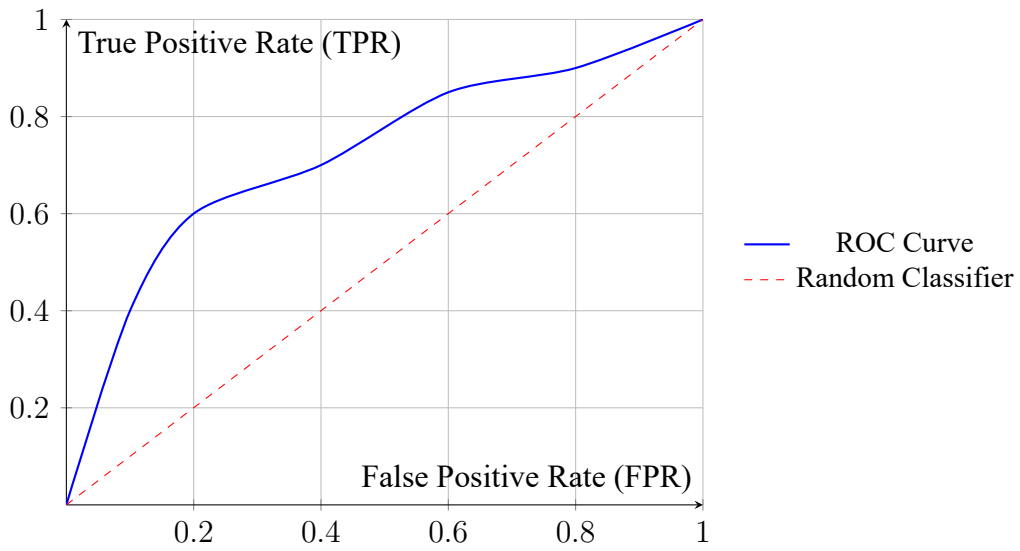

Figure 2.3: Example ROC Curve

Own tabulation based on (**geron2022hands**).

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR})\, d\text{FPR} \tag{2.12}$$

An ROC curve is a graphical representation that plots the true positive rate (TPR or recall) against the false positive rate (FPR) at various classification thresholds. By examining the

curve, one can visualize the trade-off between the models sensitivity (its ability to correctly identify positive instances) and its specificity (its ability to correctly identify negative instances) across different decision points. The value of AUC ranges from 0 to 1. An AUC of 1 indicates a perfect classifier, while an AUC of 0.5 suggests a performance no better than random guessing. In the context of predicting 'is_valid' in manufacturing, a higher AUC for the network would signify that the model is better at distinguishing between valid and invalid processes, regardless of the chosen classification threshold. This is particularly valuable when the class distribution is imbalanced, as ROC and AUC provide a more robust evaluation compared to metrics like accuracy that can be skewed by the majority class.

Now that the basic concepts of VVUQ and the metrics used to assess model quality have been introduced, the next section will delve into the specific models employed in this work. The focus will be on the ResNet Bi-LSTM Multi-Head attention network, which is designed to automatically generate VVUQ models for SBDT. This model unites the strengths of LSTM networks and attention mechanisms to handle sequential data and improve prediction accuracy.

Bidirectional LSTM Networks for Sequence-Based Anomaly Detection

Recurrent Neural Networks (RNNs)

The following section describes the model types on which the automatic VBVVQ is based. The model types are introduced in a bottom-up fashion, starting with the most basic model type and building up to the more complex models. The first model type is the Recurrent Neural Network (RNN), which is the basis for the Long Short-Term Memory (LSTM) networks. The LSTM networks are then used in a bidirectional fashion, which is the basis for the ResNet Bi-LSTM Multi-Head attention network. Because the OCEL data is sequential in nature, the RNNs are used to model the sequential data. The LSTM networks are used to overcome the limitations of the RNNs, and the bidirectional LSTM networks are used to improve the performance of the LSTM networks. The ResNet Bi-LSTM Multi-Head attention network is then introduced as a more advanced model that combines the strengths of both LSTM and attention mechanisms.[2]

Recurrent Neural Network (RNN) were the first novel networks to handle sequential data. Unlike feedforward neural networks that process each input independently, RNNs are designed to handle sequences by maintaining an internal memory, known as the hidden state, which summarizes past information (**geron2022hands**). As sequential data $\{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_T\}$ is fed into the network step by step (indexed by t), the RNN processes each element $\boldsymbol{x}_t$ while simultaneously updating its hidden state $\boldsymbol{h}_t$. This hidden state acts as the network's memory, retaining information from previous time steps. It can be thought of as the long-term memory of the network.

[2] In this section, specific mathematical notation is used. Vectors, such as inputs, hidden states, cell states, outputs, and biases, are written by bold lowercase letters (e.g., $\boldsymbol{x}, \boldsymbol{h}, \boldsymbol{c}, \boldsymbol{y}, \boldsymbol{b}$). Matrices, primarily representing weights, are represented by uppercase letters (e.g., W, U). The symbol σ denotes a generic non-linear activation function, while σ_h and σ_y specifically refer to activation functions in the hidden and output layers. In **??**, σ will refer to standard deviation. Commonly used specific activation functions like hyperbolic tangent and softmax are written as tanh and softmax. Element-wise multiplication is indicated by the symbol $\odot$.

The core computation within an RNN at time step t is calculating the new hidden state $\boldsymbol{h}_t$ based on the current input $\boldsymbol{x}_t$ and the previous hidden state $\boldsymbol{h}_{t-1}$. This relationship is typically defined as:

$$\boldsymbol{h}_t = \sigma_h(W_{xh}\boldsymbol{x}_t + W_{hh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_h) \tag{2.13}$$

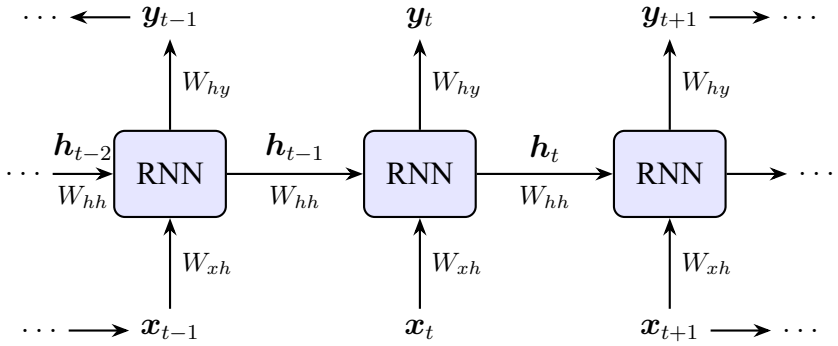

Figure 2.4: Unfolded architecture of a Recurrent Neural Network (RNN) over time, illustrating the computation of the hidden state $\boldsymbol{h}_t$ according to **??**.

Own illustration based on (**geron2022hands**).

where W_{xh} is the weight matrix connecting the input to the hidden layer, W_{hh} is the weight matrix for the recurrent connection from the previous hidden state to the current hidden state, $\boldsymbol{b}_h$ is the bias vector for the hidden layer, and σ_h is a non-linear activation function, commonly the hyperbolic tangent (tanh) or ReLU. The initial hidden state $\boldsymbol{h}_0$ is typically initialized to zeros or learned.

The output $\boldsymbol{y}_t$ at time step t can then be computed from the hidden state:

$$\boldsymbol{y}_t = \sigma_y(W_{hy}\boldsymbol{h}_t + \boldsymbol{b}_y) \tag{2.14}$$

Here, W_{hy} is the weight matrix from the hidden layer to the output layer, $\boldsymbol{b}_y$ is the output bias vector, and σ_y is an activation function suitable for the task, for example softmax for classification tasks.

One characteristic of RNNs is *parameter sharing*: The weight matrices (W_{xh}, W_{hh}, W_{hy}) and biases $(\boldsymbol{b}_h, \boldsymbol{b}_y)$ are the same across all time steps $t = 1, ..., T$. This allows the model to generalize learned patterns regardless of their position in the sequence and reduces the number of parameters to learn. This process is often understood by thinking of the network as being 'unfolded' over time, creating a deep feedforward-like structure where each layer corresponds to a time step but utilizes shared weights (**medsker2001recurrent**).

In contrast to their conceptual innovation, standard RNNs face challenges when learning dependencies over long sequences. During training using Backpropagation Through Time (BPTT), gradients are propagated backward through the unfolded network representation. The repeated multiplication involving the recurrent weight matrix W_{hh} (specifically, its Jacobian matrix containing the first derivatives) can cause gradients to either shrink exponentially towards zero (vanishing gradients) or grow uncontrollably (exploding gradients) (**hochreiter1998vanishing**). Vanishing gradients hinder the models ability to capture long-

range dependencies, as updates to weights connecting distant past inputs become negligible. Exploding gradients can lead to numerical instability during training (**philipp2017exploding**).

Long Short-Term Memory Networks (LSTMs)

To address the vanishing gradient problem and learn long-range dependencies, Long Short-Term Memory Networks (LSTMs) were introduced (**hochreiter1997long**). LSTMs have a more advanced internal structure within each recurrent unit, often called an LSTM cell.

The key innovation of the LSTM cell is the introduction of a *cell state*, $\boldsymbol{c}_t$, which acts as an information conduit, allowing information to flow through time with minimal modification. The long-term preservation is a key distinction from the RNN, where the hidden state gets overwritten. The flow of information into, out of, and within the cell state is regulated by three specialized gating mechanisms: the forget gate, the input gate, and the output gate. These gates use sigmoid activation functions (σ), which output values between 0 and 1, representing the proportion of information allowed to pass.

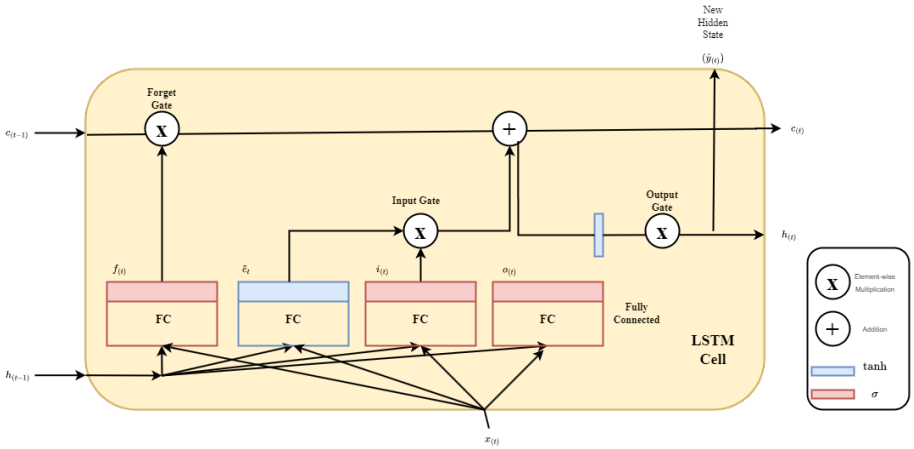

Figure 2.5: Visual representation of the LSTM cell computations detailed in Equations **??** to **??**. The diagram shows how inputs $\boldsymbol{x}_t$ and $\boldsymbol{h}_{t-1}$ interact with the forget gate ($\boldsymbol{f}_t$), input gate ($\boldsymbol{i}_t$), candidate state ($\tilde{\boldsymbol{c}}_t$), and output gate ($\boldsymbol{o}_t$) to update the cell state from $\boldsymbol{c}_{t-1}$ to $\boldsymbol{c}_t$ and compute the hidden state $\boldsymbol{h}_t$.

Source: Own illustration based on (**geron2022hands**).

At each time step t, given the input $\boldsymbol{x}_t$, the previous hidden state $\boldsymbol{h}_{t-1}$, and the previous cell state $\boldsymbol{c}_{t-1}$, the LSTM cell does the following computations:

1. **Forget Gate ($\boldsymbol{f}_t$):** Decides which information to forget from the previous cell state $\boldsymbol{c}_{t-1}$.

$$\boldsymbol{f}_t = \sigma(W_f \boldsymbol{x}_t + U_f \boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{2.15}$$

2. **Input Gate ($\boldsymbol{i}_t$):** Determines which new information from the input and previous hidden state should be added in the cell state. This involves two parts:
   - The input gate layer decides which values to update:

$$\boldsymbol{i}_t = \sigma(W_i \boldsymbol{x}_t + U_i \boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{2.16}$$

- A candidate cell state $\tilde{\boldsymbol{c}}_t$ is created with potential new values, typically using a tanh activation:

$$\tilde{\boldsymbol{c}}_t = \tanh(W_c \boldsymbol{x}_t + U_c \boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{2.17}$$

3. **Cell State Update ($\boldsymbol{c}_t$):** The old cell state $\boldsymbol{c}_{t-1}$ is updated to the new cell state $\boldsymbol{c}_t$. This involves element-wise multiplication ($\odot$) to forget parts of the old state (via $\boldsymbol{f}_t$) and add parts of the new candidate state (via $\boldsymbol{i}_t$).

$$\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \tilde{\boldsymbol{c}}_t \tag{2.18}$$

4. **Output Gate ($\boldsymbol{o}_t$):** Determines what part of the (filtered) cell state $\boldsymbol{c}_t$ should be outputted as the new hidden state $\boldsymbol{h}_t$.

$$\boldsymbol{o}_t = \sigma(W_o \boldsymbol{x}_t + U_o \boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{2.19}$$

$$\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t) \tag{2.20}$$

In these equations, W_*, U_* represent the weight matrices for connections from the input and the previous hidden state, and $\boldsymbol{b}_*$ are the bias vectors.

The gating mechanisms allow LSTMs to selectively remember or forget information over long durations. The cell state's update mechanism, involving addition and element-wise multiplication controlled by gates often close to 1 (especially the forget gate), supports a more stable gradient flow compared to the repeated matrix multiplications in simple RNNs. This characteristic, sometimes associated with the Constant Error Carousel (CEC) concept (**hochreiter1997long**), effectively mitigates the vanishing gradient problem. LSTMs have become a standard tool for various sequence modelling tasks, including time series prediction and machine sequences (**al2024rnn**).

Bidirectional LSTMs (Bi-LSTMs)

While standard LSTMs process sequences chronologically, capturing dependencies on past inputs, many tasks benefit from considering context from both past and future elements. For example, understanding why a specific machine task is performed requires knowledge about the task performed afterwards. Bidirectional LSTMs (Bi-LSTMs) address this by processing the input sequence in both forward and backward directions (**schuster1997bidirectional**).

A Bi-LSTM consists of two separate LSTM layers:

1. A **forward LSTM** processes the input sequence $\{\boldsymbol{x}_1, ..., \boldsymbol{x}_T\}$ from $t = 1$ to T, producing a sequence of forward hidden states $\{\vec{\boldsymbol{h}}_1, ..., \vec{\boldsymbol{h}}_T\}$. The computation for $\vec{\boldsymbol{h}}_t$ follows the standard LSTM equations **??** through **??**.
2. A **backward LSTM** processes the input sequence in reverse order, from $t = T$ down to 1, producing a sequence of backward hidden states $\{\boldsymbol{h}_1, ..., \boldsymbol{h}_T\}$. The computation for $\boldsymbol{h}_t$ uses a separate set of LSTM parameters and processes the sequence $\{\boldsymbol{x}_T, ..., \boldsymbol{x}_1\}$.

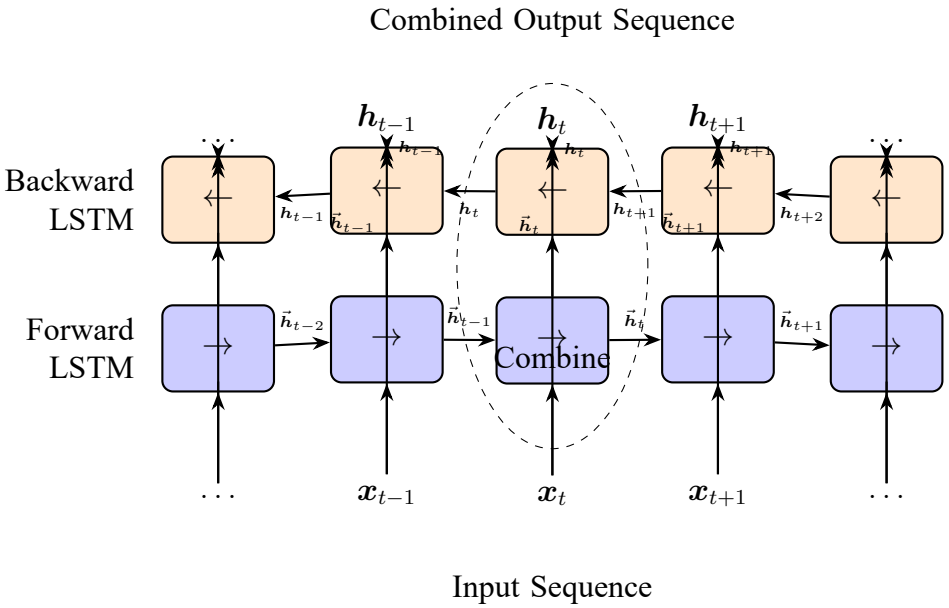

Figure 2.6: Architecture of a Bidirectional LSTM (Bi-LSTM), processing the input sequence ($\boldsymbol{x}_t$) both forwards (generating $\vec{\boldsymbol{h}}_t$) and backwards (generating $\boldsymbol{h}_t$). The final hidden states $\boldsymbol{h}_t$ are produced by combining information from both directions (via concatenation according to Equation **??**).

Own illustration.

At each time step t, the final hidden state representation $\boldsymbol{h}_t$ combines the information from both directions. A common method is concatenation:

$$\boldsymbol{h}_t = [\vec{\boldsymbol{h}}_t; \boldsymbol{h}_t] \tag{2.21}$$

where $[;]$ denotes vector concatenation. Other combination methods like summation or averaging are also possible. This combined state $\boldsymbol{h}_t$ contains information about the input $\boldsymbol{x}_t$ informed by both its preceding and succeeding context within the sequence.

Bi-LSTMs have proven effective in tasks requiring contextual understanding, such as Named Entity Recognition or sentiment analysis (**al2024rnn**). However, they are computationally more expensive than unidirectional LSTMs due to the doubled network structure.

Residual Networks (ResNets)

As neural networks became deeper to model more complex functions, researchers encountered the *degradation problem*: simply stacking more layers could lead to higher training error, even though a deeper network should theoretically be able to represent the functions learned by a shallower one (**he2016deep**). This issue, different from overfitting, showed difficulties in optimizing very deep networks. Residual Networks (ResNets) were introduced to overcome this challenge. The core idea is to reframe the learning process by having layers learn a *residual function* with respect to their input, rather than learning the desired underlying mapping directly. This is achieved using *residual blocks* containing skip connections (or shortcut connections).

Such a block of layers aiming to learn a mapping is denoted $\mathcal{H}(\boldsymbol{x})$. A residual block instead learns a residual function $\mathcal{F}(\boldsymbol{x}) = \mathcal{H}(\boldsymbol{x}) - \boldsymbol{x}$. The output $\boldsymbol{y}$ of the residual block is then computed as:

$$\boldsymbol{y} = \mathcal{F}(\boldsymbol{x}, \{W_i\}) + \boldsymbol{x} \tag{2.22}$$

Here, $\boldsymbol{x}$ is the input to the block, $\mathcal{F}(\boldsymbol{x}, \{W_i\})$ is the function learned by the stacked layers within the block (described by weights $\{W_i\}$), and the $+\boldsymbol{x}$ term is an identity skip connection. This formulation makes it easier for the network to learn an identity mapping by driving the weights in $\mathcal{F}$ towards zero.

These skip connections provide alternative pathways for gradient propagation during back-propagation. Gradients can flow directly through the identity connections, overjumping the layers in $\mathcal{F}$. This avoidsthe vanishing gradient problem in very deep networks, allowing for the successful training of models with hundreds or even thousands of layers (**he2016deep**). While originally developed for computer vision, the principle of residual connections can be applied to other architectures, including sequential models. Using ResNet-like connections within or around (Bi-)LSTM layers can stabilize training and allow for deeper sequential architectures, enabling the capture of more temporal patterns without suffering from degradation.

Multi-head Attention Mechanism

In sequence modelling, particularly with long sequences, not all parts of the input are equally relevant for making a prediction at a given time step. Attention mechanisms allow a model to dynamically focus on the most important parts of the input sequence (**chorowski2014end**). A core idea is *self-attention*, where the mechanism relates different positions of a single sequence to compute its representation (**vaswani2017attention**).

Self-attention works on input vectors, typically representing tokens or time steps in a sequence. For each input vector, three representations are learned through linear transformations: a query ($\boldsymbol{q}$), a key ($\boldsymbol{k}$), and a value ($\boldsymbol{v}$). The attention mechanism computes the output as a weighted sum of the values, where the weight assigned to each value is determined by the compatibility (often measured by dot product) between its corresponding key and the query.

The most common form is *Scaled Dot-Product Attention*, calculated for a set of queries Q, keys K, and values V (where Q, K, V are matrices stacking the $\boldsymbol{q}, \boldsymbol{k}, \boldsymbol{v}$ vectors):

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.23}$$

Here, QK^T computes the dot products between all query-key pairs. The result is scaled by $\sqrt{d_k}$, where d_k is the dimension of the keys, to prevent the dot products from becoming too large and pushing the softmax function into regions with very small gradients. The softmax function normalizes these scores into attention weights, which sum to 1. Finally, these weights are multiplied by the Value matrix V to produce the output, highlighting the values corresponding to the most relevant keys for each query.

Multi-Head Attention enhances this mechanism by performing the attention calculation multiple times in parallel, each with different, learned linear projections of the original Q, K, V. This allows the model to jointly attend to information from different representation subspaces at different positions (**vaswani2017attention**).

Let the number of heads be h. For each head $i \in \{1, ..., h\}$, the input Q, K, V are projected using learned weight matrices W_i^Q, W_i^K, W_i^V:

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \tag{2.24}$$

The outputs of all heads are then concatenated and projected one final time using another learned weight matrix W^O:

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_h)W^O \tag{2.25}$$

This multi-head structure allows each head to potentially focus on different types of relationships, leading to a richer representation. Multi-head attention is a cornerstone of the transformer architecture (**vaswani2017attention**) and is increasingly combined with other models like LSTMs.

The Integrated Architecture for Anomaly Detection

The theoretical components described above can be integrated. For the challenge of anomaly detection in sequential manufacturing process data within the context of SBDT, an architecture combining these elements offers significant advantages.

The proposed model utilizes Bi-LSTM layers as the core sequence processing units to capture the temporal dynamics inherent in manufacturing operations, using their ability to model long-range dependencies as described in Section **??** and **??**. To potentially enable deeper Bi-LSTM stacks and stabilize training, residual connections, inspired by ResNet principles (Section **??**), can be incorporated within or between these layers. Following the Bi-LSTM layers, a multi-head attention mechanism (Section **??**) is applied to the sequence of hidden states generated by the Bi-LSTMs. This allows the model to adaptively weigh the importance of different time steps or features in the sequence when making a final prediction. By focusing on the most salient parts of the process history, the attention mechanism can enhance the models ability to distinguish between normal and anomalous patterns. The multi-head approach enables simultaneous focus on diverse aspects potentially indicative of anomalies. The final output of the attention layer is passed through further feedforward layers to produce the anomaly classification (a probability score indicating deviation from normal behaviour). This integrated architecture aims to use the strengths of each component: Temporal modelling from Bi-LSTMs, improved trainability from residual connections, and context-aware focusing from multi-head attention.

The architechture directly relates to the VVUQ of SBDTs. Accurate anomaly classifica-

tion serves as a form of validation for the DTs representation of normal process behaviour. Analysing the attention weights (derived from Equation **??**) can provide insights into which process steps or features the model considers most indicative of anomalies, helping in the verification and interpretation of both the model and the manufacturing process itself. UQ is then performed by conducting statistical significance testing. The next chapter will describe modern VVUQ techniques within the SBDT context.

2.4.4 Modern VVUQ in the Context of SBDT

ML-based VVUQ approaches heavily rely on data. In the preceding sections, the thesis introduced several interfaces which may serve as suitable data sources. PM/OCED may provide event logs for VVUQ approaches —ML approaches can ingest those. PPC systems may provide additional data to enrich the event log. This section will shed light on a systematic literature review (SLR) on V&V in the context of DTs. After summarizing the main findings, several V&V approaches with increasing sophistication are presented. The section closes with a discussion of the most promising approaches.

Bitencourt2023<empty citation> conducted a SLR on V&V in the context of DT —a relatively new field. They did not consider uncertainty quantification in their SLR. As **hua2022validation<empt** note, there are no structured and established frameworks for validating DTs. This statement holds for all sectors where DTs are applied. The SLR analysed 269 papers. They applied the 4R framework by **Osho2022jmsy<empty citation>** to describe the capabilities of the analysed twin frameworks. The 4R framework consists of the four dimensions *Representation*, *Replication*, *Reality* and *Relational*. The SLR found that most frameworks (49%) rather developed a DM or DS. Another 26% of DT were only able to *represent* the physical entity. Highly sophisticated DT were the exception, not the rule. Considering this trend, V&V may not be a topic researchers are interested in at first sight. **Bitencourt2023<empty citation>** identified a trend throughout the years of increasing modelling capabilities of the considered DT. Up to the year 2023, the trend is still increasing. They identify more data sources as the main driver for this trend. From the 269 papers, 47% have been applied in the manufacturing domain. Of all classified DT, one key insight is that most authors performed at least one form of V&V.

DTs ranking in the *reality* category where most often verified and validated. The authors deduce that because of the increasing complexity of the model, V&V may become a stressing topic. **Nie2023rcim<empty citation>** propose a multi-agent and cloud-edge orchestration framework leveraging Digital Twin and IIoT to optimize distributed manufacturing resources. Their framework integrates a Convolutional Neural Network (CNN) with a bidirectional Long Short Term Memory (LSTM) module to perform scheduling and self-adaptive strategies for intelligent, real-time production control. They compare their network with an earlier adopted algorithm and its result to perform V&V. Quantifying the validity, they use Mean Squared Error (MSE) and Mean Absolute Error (MAE), see **??**. **Lv2023rcim<empty citation>** defined target scenarios and boundaries to verify their DT for fault diagnosis of a drilling platform. They defined intervals for the metrics to be valid. Validation was performed by conducting a case study. Latter can be termed qualitative validation because the factual similarity

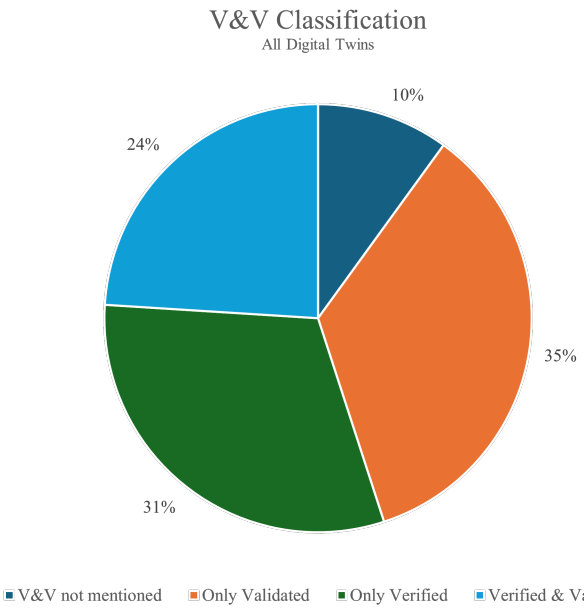

Figure 2.7: Donut chart showing the distribution of V&V methods in the context of DT. Own illustration based on **Bitencourt2023<empty citation>**.

of the case study against the DT has been assessed. Regarding their verification approach, they go one step further than **Nie2023rcim<empty citation>** by defining control intervals. This is somewhat automatic verification, but still requires human intervention when the values leave the defined intervals. **Chen2023**; **Mykoniatis2021jim<empty citation>** both performed visual inspection to verify their DT. Both used cameras or other visual inspection tools to compare the DT results with reality. **Chen2023<empty citation>** used thermal cameras to conduct temperature scans and compared the results with temperature prediction of the twin. **Mykoniatis2021jim<empty citation>** used video recordings to validate their twin behaviour, but consulted KPIs to further validate their twin. Both approaches use aids for measurements and visual inspection. The measurements still have to be conducted and compared manually. Some authors performed statistical V&V, although the measurements were not automated. **Wang2023asoc<empty citation>** compared historical data with the twin data and calculated the mean absolute percentage error of both datasets, **??**. For validation they relied on conducting a case study, thus not fully automating V&V. **Min2019ijinfomgt<empty citation>** performed verification using PPC KPIs, automating the V&V process even further. Their validation included using a validation set and measuring a set of metrics including error of fit and accuracy. This approach can be considered the first step towards a fully automated VVUQ process. **Bitencourt2023<empty citation>** conclude that the majority of the analysed papers performed V&V manually, with only a few authors automating the process. They also note that most authors did not consider uncertainty quantification in their work. And indeed, most work was performed in only conducting verification (31%) where case studies were the method of choice (**Eunike2022**; **jia2023from**; **kumbhar2023digital**; **Leng2020rcim**; **Leng2022aei**). Case studies were also often applied where only validation took place (35%) (**Alam2023**; **Dobaj2022**; **kherbache2022digital**; **Latsou2023**; **Leng2021jclepro**; **Negri2019promfg**). The SLR shows that V&V is a topic of interest in the context of DT, but most authors still rely

on manual methods. The trend towards more sophisticated DTs and the increasing complexity of models will likely drive further research in this area. The SLR also highlights the need for more automated and standardized V&V processes, especially in the context of uncertainty quantification. Very few authors performed fully automated VVUQ for DT. The ones who did will now be discussed in more detail.

K-Nearest Neighbours (KNN) for Digital Twin Accreditation in Manufacturing

The k-Nearest Neighbours (KNN) algorithm, a lazy learning approach grounded in measuring feature similarity, can be used for the accreditation of manufacturing digital twins (**dos2024simulation**). KNN classifies new data points by identifying the majority class among the K nearest neighbours in the training dataset. The determination of "being close' relies on various distance metrics. Most commonly, the Euclidean distance (see **??** is used. Selecting an appropriate value for K is crucial for balancing the models bias and variance. A small K can lead to overfitting, while a large K may provide no insights at all. It is not an eager learner like DTrees or NN because it does not learn a model from the training data. Instead, it stores the training data and makes predictions based on the stored data. This characteristic makes KNN particularly suitable for scenarios where the underlying distribution of the data is unknown or complex. The algorithm's simplicity and interpretability make it a popular choice for various classification tasks, including those in manufacturing contexts.

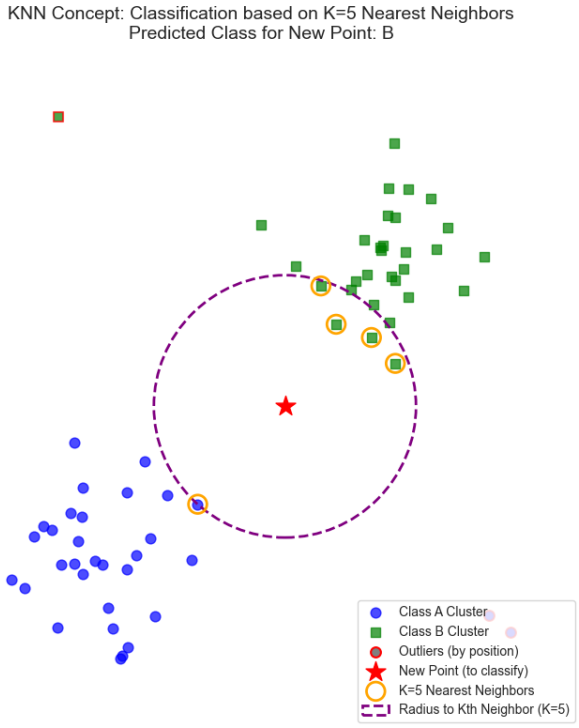

Figure 2.8: KNN algorithm. The algorithm classifies a new data point based on the majority class of its K nearest neighbours in the training dataset. Euclidean distance metric has been used to determine the distance between data points.

Own illustration.

In the given figure, the lazy learning approach of KNN near completion is shown. The star-formed point has to be assigned to class one or two. Based on the count of points in the near region of the unassigned point, the algorithm will decide class two. The approach by

dos2024simulation<empty citation> integrates KNN with p-charts for the concurrent validation of SBDT (**dos2024digital**). This involves monitoring the DTs output over time and using KNN to classify its behaviour. Subsequently, p-charts are employed to statistically monitor the proportion of anomalies detected by KNN. The X-axis represents the discretized time t, the Y-axis represents the proportion of anomalies detected by the KNN. P-charts normally show a relative proportion of nonconforming items in a sample (**acosta1999improved**). **dos2024simulation<empty citation>** enriched their chart with an upper control limit (UCL) and lower control limit (LCL), which stand for acceptable statistical boundaries. This approach is similar to **Nie2023rcim<empty citation>** who also defined control intervals. If the p-value exceeds the boundaries, the twin may deviate from real behaviour. This method offers the advantage of real-time monitoring. However, it is sensitive to the choice of the hyperparameter K and can be computationally expensive for large datasets (**dos2024simulation**). The presence of outliers and noisy data can also impact the accuracy of KNN, and its performance may decrease in high-dimensional spaces. The authors utilized metrics such as accuracy, precision, recall and F1-Score (**??**. They also used cross validation for increasing robustness of the prediction. By varying values for K, an optimal split may be found. Recalling the different kinds of anomalies from **??**, KNN is able to capture point anomalies and contextual anomalies. Collective anomalies are not captured by KNN, as they are not able to learn from the data.

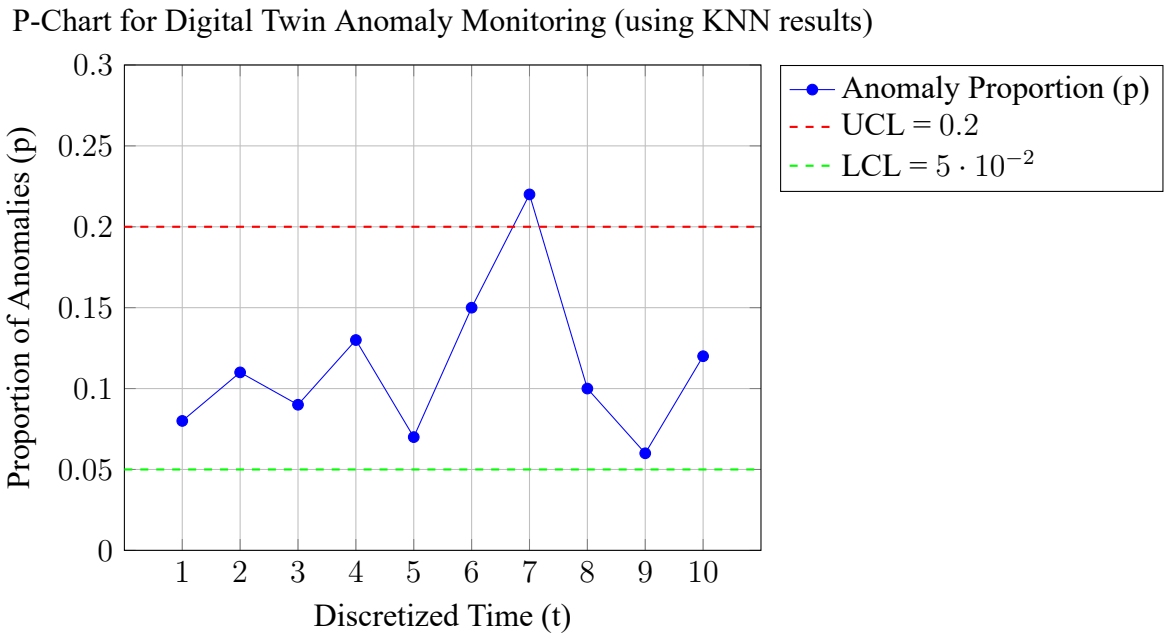

Figure 2.9: Example P-Chart showing the proportion of anomalies detected by KNN over time for DT accreditation, with Upper (UCL) and Lower (LCL) control limits. A point outside the limits (like at t=7) suggests potential deviation.

Own illustration based on **dos2024simulation<empty citation>**.

Time Series Classification for Fault Localization

Time series classification techniques also play an important role in detecting and classifying anomalies within manufacturing DT systems (**Lugaresi2023compind**). These techniques involve training classifiers on simulation data generated by the DT to identify and catego-

rize faults in the real physical system (**dihan2024digital**). A case study involving a scale-model gantry crane demonstrates the application of time series classification for this purpose (**mertens2024localizing**). Other techniques in this domain often involve deep learning models such as CNN, RNN, and LSTM networks. These models perform well at automatically extracting relevant features from time series data (**cao2023real**), outperforming traditional rule-based or statistical methods in complex scenarios. Time series classification offers the benefits of automated anomaly detection and classification, with the potential for early fault prediction. However, a key challenge is the need for sufficient labelled data for training these models (**zemskov2024security**), and real-time performance is often a critical requirement.

Uncertainty Quantification for Digital Twin Validation

V&V is an important aspect of SBDT, but uncertainty quantification (UQ) is equally crucial. UQ aims to quantify the uncertainty associated with the predictions made by the digital twin, providing insights into the reliability and confidence of its outputs (**sel2025survey**). Certainty in predictions creates trust in the application, a desirable property of SBDT when used in practice (**dwivedi2023explainable**). State of the art approaches to tackle uncertainty are Bayesian Neural Networks (BNN) (**li2017dynamic**), Monte Carlo Dropout (MCD), Deep Ensembles (DE) and Gaussian Processes (GP). BNNs are able to quantify epistemic and aleatoric uncertainty (see **??**. They are a type of neural network that introduce uncertainty into its predictions by treating the weights and biases as probability distributions rather than fixed values. Thus, each weight has a mean and a variance parameter for quantifying its uncertainty. During inference, samples are drawn from this distribution to *learn* its characteristics. This allows the model to capture uncertainty in the data and make probabilistic predictions (**li2017dynamic**). MCD, DE and GP train surrogate models (see **??**) to approximate the uncertainty in the models predictions.

MCD is a technique that uses dropout (**srivastava2014dropout**) during both training and inference phases to approximate the uncertainty in the models predictions, unlike traditional dropout which is only used during training. By randomly dropping out neurons during inference with a chosen probability p, MCD generates multiple predictions, which then are averaged and can be used to estimate uncertainty. The key insight here is that effectively the sampling happens from a dozen different NN because of their different weight configurations. Thus, ensemble-learner like accuracy without the computational cost of training multiple complete models like in DE has been reached. DE involves training multiple models with different initializations and architectures, and then combining their predictions to obtain a more robust estimate of uncertainty (**rahaman2021uncertainty**). The difference between DE and MCD lies in statistically independent NNs, meaning they do not share statistical interaction processes. Furthermore, they are initialized randomly, whereas MCD networks share the same weight initialization. MCD and DE both provide good estimates for uncertainty, but it is recommended to apply both techniques after fine-tuning of the DNN architecture (**kamali2024advancements**). GPs are a non-parametric Bayesian approach that models the underlying function as a distribution over functions (**bilionis2012multi**). They provide a measure of uncertainty in their predictions by estimating the variance and mean of

the predicted outputs (**Burr2025TEADT**).

Overall VVUQ for SBDT is conducted with varying efforts. There is not a lot of research on fully automated VVUQ processes. The SLR by **Bitencourt2023<empty citation>** shows that most authors still rely on manual methods, if VVUQ has been performed at all. The trend towards more sophisticated DTs and the increasing complexity of models will likely drive further research in this area. The SLR also highlights the need for more automated and standardized VVUQ processes, especially in the context of UQ. Very few authors performed fully automated VVUQ for DT. The ones who did were presented in this section.

This chapter established the theoretical foundation for the thesis, focusing on SBDT. It covered key concepts including the simulation of DMFS using DES, and an analysis of different DT architectures such as SBDT, DDDT, and HDT. The role of KPIs in balancing SBDT trade-offs was described. PM, particularly using OCEL for validation due to its ability to handle complex interactions, was explored. Furthermore, the chapter addressed VVUQ, contrasting traditional methods with modern ML approaches like KNN and BNN for ensuring SBDT reliability. The need for automated VVUQ frameworks and associated challenges, like UQ, were highlighted. These theoretical elements underpin the development of the proposed SBDT VVUQ framework aimed at enhancing reliability, which will be detailed in the next chapter.

Chapter 3: Methodology

This chapter develops the methodological approach for automatically performing VVUQ for automatically generated SBDT in the manufacturing domain. The foundational methodology uses a data-driven framework and applies ML techniques to verify and validate the SBDT. The following chapter first derives requirements besides the identified key requirements given in **??**, categorizes them and presents a conceptual blueprint. It is based on the theoretical findings from Chapter **??** and designed to ensure that the VVUQ process is systematic, reproducible, and adaptable to different use cases. It then describes the statistical significance testing method used to validate the results of the VVUQ process.

3.1 Framework Design

3.1.1 Requirements Engineering

An analysis of the requirements for the proposed VVUQ framework is essential. **sindhgatta2005functional<** distinguishes between functional (FR, **van2001goal<empty citation>**) and non-functional requirements (NFR, **glinz2005rethinking<empty citation>**). Functional requirements define the specific functions and features that the system must provide, while non-functional requirements specify the quality attributes, constraints, and performance criteria that the system must possess. Additionally, technical requirements (TR) and operational requirements (OR) are considered. TR specify the technical specifications of the system which are necessary to meet the functional requirements (**chikh2012new**), while OR show the operational constraints and conditions under which the system must operate (**incose2023incose**).

The following **??** summarizes the key requirements identified in **??** and adds requirements which have been derived from the theoretical findings in Chapter **??**.

FR include low-latency real-time data integration for synchronization, automatic VVUQ featuring anomaly detection, UQ, and adaptive recalibration, plus alarm management and automated KPI-based performance evaluation. NFR include dynamic scalability, robustness against uncertainties like noise and concept drift, security, interoperability, user-friendliness enabling easy monitoring and visualization, and continuous learning capabilities. TR demand a scalable cloud/edge architecture supporting many data formats and protocols such as OPC UA and MQTT, integration of the ResNet BiLSTM network, modules for uncertainty quantification, defined hardware resources, and data source management with quality standards. Finally, OR stresses high-standard data stewardship, continuous monitoring via logging, defined maintenance procedures, and a flexible runtime environment.

3.1.2 An Automated VVUQ Framework for Automatically Generated SBDTs

The framework consists of five interconnected layers that form a closed-loop system with continuous data flow, validation and improvement. At the start, diverse data sources such

Functional Requirements	Real-Time Data Integration
	Automatic VVUQ
	Anomaly Detection
	Adaptive Recalibration
	Alert System
	Performance Monitoring
Non-Functional Requirements	Dynamic Scalability
	Resilience against Uncertainty
	Safety
	Interoperability
	Ease of Use
	Continuous Optimization
	Continuous Learning
Technical Requirements	Scalable Architecture
	Data Compatibility
	Hardware Requirements
	Historical Data
	Meta Data
	Labelled Data
	Data Quality
Operational Requirements	Data Stewardship
	Certifications and Monitoring
	Continuous Maintenance
	Runtime Environment

Figure 3.1: Key Requirements for the VVUQ framework differentiated by FR, NFR, TR, and OR. Source: Own illustration.

as PPC systems, DES outputs, PM, expert knowledge, sensor networks, and CRM feedback provide the raw inputs, which are centralized in a data lake with structured views managed by a data warehouse. The data warehouse enriches the incoming data with engineering features. The DL collects and integrates this data, preprocesses it through cleaning, normalization and feature extraction. The DL performs a subset of the actions of the data warehouse, but tailors the data specifically for the SBDT. This structured approach meets real-time integration and quality requirements considered above.

Within the TL, the physical entity and its SBDT are in connection through a bi-directional data flow that ensures real-time synchronization. Evidently this framework does not allow DS or DM, because the closed-loop structure would be interrupted here. Complementing this, the VVUQ Layer is dedicated to the automatic VVUQ processes. This layer forms the core of the solution designed to answer how automated VVUQ can be efficiently implemented while maintaining accuracy (**??**) and determining which data-driven approaches are best suited for this task (**??**). It ensures that the SBDT represents both the conceptual model and the physical entity, using advanced methods such as a ResNet Bi-LSTM Attention model. The choice of this specific deep learning architecture represents the frameworks proposed answer to identifying the most suitable data-driven approaches for detecting discrepancies between simulated behaviour and real operational data (**??**). Furthermore, this DL approach, which processes historical and real-time data, provides robust anomaly detection capabilities, triggers recalibration when necessary, and validates outputs against actual measurements, thus fulfilling adaptive recalibration and anomaly detection requirements.

Translating these technical evaluations into corporate processes and recommendations is the role of the DSL. This layer synthesizes the VVUQ assessments into prioritized *short-term* decision recommendations, monitors KPIs, identifies data gaps, and manages alerts. By serv-

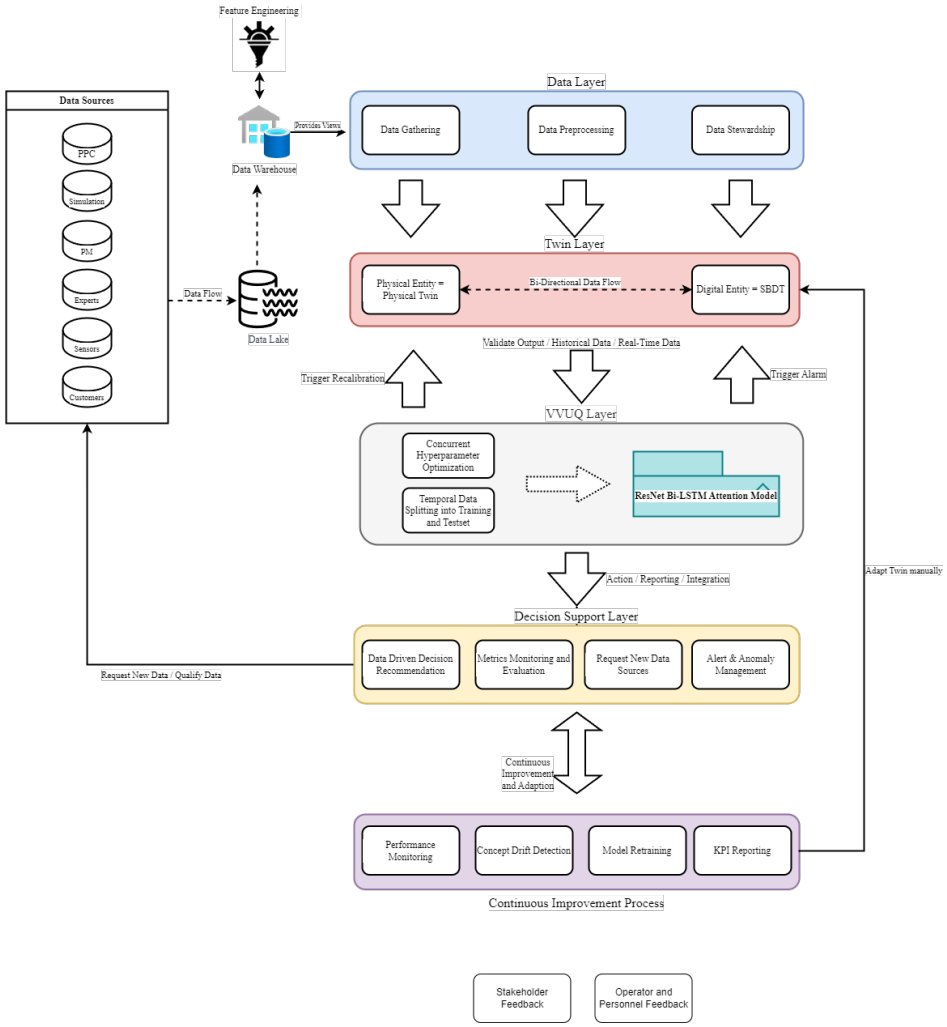

Figure 3.2: Framework for VVUQ of SBDT in the manufacturing domain. The framework starts with the data sources which all lead into the data lake. The data warehouse provides the Data Layer (DL) with different views. The DL further enriches the data to feed it into the Twin Layer (TL). The TL contains the DT and the physical entity. The TL is connected to the VVUQ Layer (VVUQL). It incorporates the ResNet BiLSTM network for VVUQ of the twin. It can trigger alarms and recommendations for action. The VVUQL is connected to the Decision Support Layer (DSL) which provides different data analysis and visualization tools. The DSL is responsible for the short-term decision making to manage the VVUQ process. The DSL is connected to the user interface (UI) which provides the user with a dashboard for monitoring and controlling the system. The DSL can request new data from the Data Sources. It also is connected to the Continuous Improvement Process layer (CIP) which is responsible for the long-term decision making.

Source: Own illustration.

ing as the interface between technical processes and operational decision-making, it ensures that manufacturing personnel receive insights. The CIP layer further enhances the system by monitoring performance, detecting concept drift, scheduling model retraining based on accumulated data, and generating KPI reports. The CIP layer manages *long-term* feedback. Through an "Adapt Twin' process, this layer feeds insights back into the TL, ensuring that the digital twin evolves in connection with changes in the physical entity.

3.1.3 Online Validation and Continuous Feedback Loop

The entire framework operates as a closed-loop system characterized by continuous data collection from diverse sources, real-time synchronization between physical and digital entities, ongoing validation of simulation outputs against physical measurements, and a systematic flow of decision recommendations and alerts. This architecture ensures interoperability by providing standardized interfaces between layers and existing manufacturing systems while maintaining the flexibility to adapt to various manufacturing contexts. The framework transforms VVUQ from a periodic technical assessment into an ongoing process that enhances DT quality and decision support capabilities. By meeting comprehensive functional, non-functional, technical, and operational requirements, this framework not only improves the accuracy and effectiveness of simulation-based digital twins but also facilitates their practical application as decision support tools in modern manufacturing environments. Stakeholder feedback and personnel suggestions are integrated into the framework through the DSL and CIP layers, ensuring that the system evolves in response to changing needs and conditions.

3.2 Permutation Testing for Statistical Significance

While performance metrics such as accuracy, precision, recall, F1-score, or ROC AUC provide valuable insights into the effectiveness of a model or the magnitude of difference between datasets, they do not quantify the statistical significance of the observed results. An apparently strong performance or large difference could potentially arise due to random chance, especially with limited data or complex models. To assess whether an observed outcome is statistically meaningful or likely random, permutation testing provides a robust, non-parametric approach (**welch1990construction**).

Permutation testing is particularly useful when the underlying distribution of the test statistic is unknown or difficult to derive analytically, which is often the case with complex machine learning models or custom evaluation metrics. The core idea is to empirically generate a distribution of the test statistic under the null hypothesis (H_0) – the hypothesis that there is no real effect or difference. For example, a classifier cannot distinguish between classes better than random chance, or two data samples originate from the same underlying distribution.

The process involves the following steps:

1. **Define the Null Hypothesis (H_0):** State the specific null hypothesis being tested. For instance, in a classification task comparing two data sources (real vs. simulated), H_0 might be that the data source labels are independent of the input features.
2. **Choose a Test Statistic (S):** Select a metric to quantify the effect or performance

of interest. This could be the difference in means between two groups, a correlation coefficient, classifier accuracy, ROC AUC, or another relevant measure.

3. **Compute the Observed Statistic (S_{obs}):** Calculate the chosen test statistic S on the original, unpermuted dataset.
4. **Generate the Null Distribution:** Repeat the following steps a large number of times (N, e.g., $N = 1000$ or more):
   - Create a permuted dataset by randomly shuffling the relevant labels or group assignments while keeping the features intact. For example, in a two-class classification problem, shuffle the class labels across all data instances. This process breaks the potential association between features and labels, simulating the scenario under H_0.
   - Compute the test statistic S on this permuted dataset, denoted as S_{perm}.

   The collection of these N permuted statistics ($S_{perm,1}, S_{perm,2}, ..., S_{perm,N}$) forms the empirical null distribution.
5. **Calculate the p-value:** The p-value represents the probability of observing a test statistic as extreme as, or more extreme than, S_{obs} under the assumption that H_0 is true. It is calculated as the proportion of permuted statistics that are greater than or equal to (or less than or equal to, depending on the hypothesis direction) the observed statistic:

$$p = \frac{\sum_{i=1}^{N} \mathbb{I}(S_{perm,i} \geq S_{obs})}{N} \tag{3.1}$$

   where $\mathbb{I}(\cdot)$ is the indicator function (1 if the condition is true, 0 otherwise). For a two-sided test, the calculation involves considering extreme values in both tails of the null distribution.
6. **Interpret the Result:** Compare the calculated p-value to a pre-defined significance level α (commonly $\alpha = 0.05$ or $\alpha = 0.01$).
   - If $p < \alpha$: Reject the null hypothesis H_0. This indicates that the observed result (S_{obs}) is statistically significant and unlikely to have occurred merely by chance. There is evidence for the alternative hypothesis (e.g., the classifier performs significantly better than chance, or the two groups are significantly different).
   - If $p \geq \alpha$: Fail to reject the null hypothesis H_0. This means there is insufficient statistical evidence to conclude that the observed result is different from what might be expected under random chance, given the current data and test.

By applying permutation testing, one can add statistical rigor to the evaluation of models and comparisons between datasets, providing stronger support for the conclusions drawn from empirical results. This method will be employed in the validation phase of this thesis (Chapter **??**) to assess the significance of the findings.

Chapter 4: Implementation

This chapter shows the technical implementation of the ResNet-BiLSTM-Attention framework **Fischer2025ResNetBiLSTM<empty citation>** by translating the conceptual framework from **??** into a functional system that processes manufacturing OCEL, extracts relevant features, trains models, and evaluates the fidelity of SBDT.

4.1 Architecture and System Setup

4.1.1 Architecture

The implementation follows a modular architecture. It can process both streams and batches of manufacturing data.

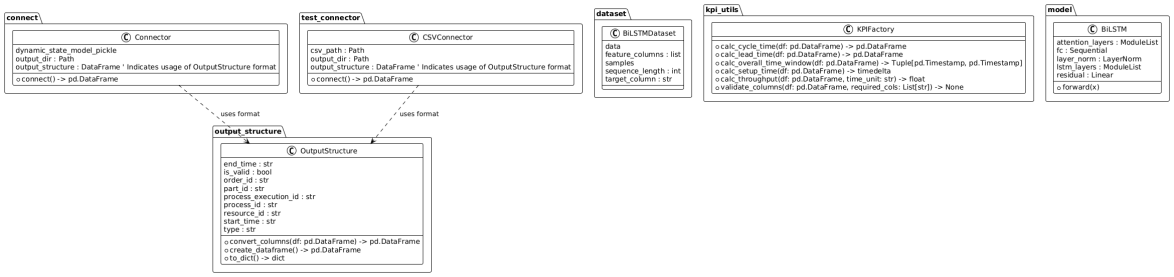

Figure 4.1: Unified Modelling Language (UML) diagram of the ResNet-BiLSTM-Attention framework for validating SBDT in manufacturing environments.

Source: Own illustration.

The UML model (**PlantUML**) in **??** illustrates the systems architecture, which consists of several components that interact to achieve the validation of SBDT. The components include:

- **Data Connectors:** In the given code, two classes `InputStructure` and `OutputStructure` are responsible for reading and writing data. The `InputStructure` class reads raw data from the twin simulation, while the `OutputStructure` class writes the processed data into the OCEL format developed for this framework, see **??**. The connector assigns IDs by enumeration for different parts, resources, and processes. The IDs are used to identify the different components in the manufacturing process. This is a requirement for OCED. The mapping is returned as JSON files for each category, respectively.
- **KPIFactory:** Contains utility functions to calculate a diverse set of KPIs for PPC evaluation of the process.
- **Baseline Model:** Implements a baseline model for comparison with the ResNet-BiLSTM-Attention model. This model serves as a reference point to evaluate the performance of the more complex architecture.
- **PyTorch DataSet and DataLoader:** Handles the conversion of raw data into a format suitable for the ResNet-BiLSTM-Attention model.

- **ResNet-BiLSTM-Attention:** The core model that combines ResNet and BiLSTM architectures with attention mechanisms to learn from the data.

4.1.2 Tech Stack and Setup

The implementation is built with Python 3.12 (**Python**), using the following frameworks and libraries:

- **PyTorch:** Powers the deep learning components, chosen for its dynamic computational graph that supports complex architecture development and debugging (**PyTorch**).
- **Pandas & NumPy:** Handle data manipulation, transformation, and numerical operations (**NumPy**; **Pandas**).
- **Scikit-Learn:** Provides implementation of the baseline model and evaluation metrics (**Scikit-Learn**).
- **Matplotlib & Graphviz:** Generate visualizations of model architecture and performance (**Matplotlib**; **Graphviz**).
- **UV Package Manager:** Ensures reproducible dependency management with exact version pinning (**UV**).

PyTorch was preferred over TensorFlow due to its flexibility and ease of use, especially for research purposes. The implementation is designed to be modular and extensible, allowing for future enhancements and adaptations to different manufacturing environments. The system is designed to run on a standard workstation with a multi-core CPU and an optional GPU for accelerated training. The given framework utilizes CUDA (**NVIDIA_CUDA**) for GPU acceleration.

Data Preprocessing

After laying out the architecture and system setup, this section focuses on the data preprocessing steps necessary for preparing the input data for the baseline model.

4.1.3 OCEL Format

Both models used in this thesis require the input data to be in the OCEL format, see **??**. The columns are inspired by the SBDT comparison model by **schwede2024learning<empty citation>**. For the given framework, the format is expected as input in the following format:

The terms are consistent with the upper cited framework. For the model components, following features may be considered:

1. **Time Model:** `duration`, `sequence_number`, `hour_of_day_cos`, `hour_of_day_sin`, `day_of_week_cos`, `day_of_week_sin`, `is_break`, `is_not_weekday`.
2. **Transition Model:** `part_id`, `resource_id`, `sequence_number`, `duration`.
3. **Transformation Model:** `part_id`, `process_id`, `sequence_number`.
4. **Quality Model:** Exclusion of quality information in the given model.
5. **Resource Model:** `resource_id`, `part_id`, `process_id`.
6. **Resource Capacity Model:** `resource_id`; Note: Insufficient data available for detailed capacity modelling.

Table 4.1: Detailed structure, data types, and description of the processed manufacturing OCEL.

Column Name	Data Type	Description
process_execution_id	int	Unique identifier for the specific process recorded.
order_id	Index (int/str)	Identifier for the overall manufacturing order this event belongs to.
start_time	Timestamp [UTC]	The precise timestamp marking the beginning of the event, adjusted to UTC.
end_time	Timestamp [UTC]	The precise timestamp marking the end of the event, adjusted to UTC.
duration	float (seconds)	The calculated duration of the event (end_time - start_time) in total seconds.
part_id	int	Identifier for the specific part or component being processed or handled during the event.
resource_id	int	Identifier for the machine, station, or other resource involved in the event.
process_id	int	Identifier indicating the type of process step or operation performed (e.g., milling, assembly).
type	str	A textual description or category classifying the type of data recorded.
is_valid	bool	Boolean flag indicating whether the recorded event sequence or outcome is considered valid in the given setting.

Source: Own tabulation.

7. **Process Model:** process_id, duration, sequence_number.
8. **KPI-based Features:** throughput, cycle_time_sec, lead_time_sec, setup_time_sec.

This table does not forbid adding relational logic by the modeller. For example, each ID may be a foreign key to another table. Each row in the OCEL represents a single event instance. This schema aligns with OCED principles (**??**) by explicitly linking each recorded event instance (process_execution_id per timestamp) to the multiple object instances (order_id, part_id, resource_id, process_id) involved in its execution. This inherent multi-object relationship within each event record is important for modelling complex process dependencies. The structure empowers the representation of complex control flows often found in manufacturing. Parallel execution paths (AND-split $\wedge$)[1] can be inferred by identifying events associated with different resources or process steps occurring *within* over-

[1] The AND-split pattern means concurrent execution paths within a process, where multiple activities can be executed simultaneously. Here, this can be the case when different parts are prepared for further manufacturing in parallel because they dont depend on each other.

lapping time intervals (`start_time`, `end_time`) but related to shared object instances, such as a common `order_id`. Alternative paths and process variants (exclusive OR-split $\oplus$)[2] are explicitly captured through the diversity of event sequences observed across different process instantiations, for example grouped by `order_id`). The log records exactly which path or sequence of activities occurred for each instance. The object-centric nature enriches this analysis by providing context that can explain why a particular variant or choice was executed. The OCEL retains its sequential linear character by grouping it by `order_id` and sorting ascending related to `end_time`. This allows for the reconstruction of the process flow. The use-case in **??** will engineer further features from the OCEL format.

While conciseness of the data structure was a given requirement, the OCEL format is not fully compliant with the OCED standard (**van2023object**). The OCEL format used in this thesis is rather simplified. Specifically, this simplification means the schema does not include distinct tables for object instances and their types, explicit modelling of static O2O relationships (**??**) or the capability to store timestamped attributes associated directly with objects rather than events. Despite these omissions the implemented structure retains the core OCED principles. The goal is rather to *learn* these relationships from the data itself.

4.2 Model Implementation

After the basic OCEL format is established, the next step is to implement the models. The implementation consists of two main components: a baseline model and the black-box model. The baseline model serves as a reference point for evaluating the performance of the more complex architecture. The ResNet-BiLSTM-Attention model is designed to learn from the data and make predictions based on the features extracted from the OCEL format, as well as the baseline model does. The key distinction lies in interpretability: While the Decision Tree Classifier (DTRee) baseline (white-box) enables direct rule extraction, the ResNet-BiLSTM-Attention (black-box) trades explainability for sequential pattern capture. In the course of the experiments, the baseline model can serve as a VVUQ tool by itself. Data leakage may be diagnosable by analysing the results of the `DecisionTreeClassifier`. Furthermore, model components can be analysed by themselves through adaptive feature selection (AFS) during training. If the classifier was able to learn well on the dataset, the model component may be present in the dataset and thus the SBDT was able to encode this information in the data. To avoid that artifacts or random noise was learned, sanity checks are a common choice (see **??**, (**adebayo2018sanity**)). The ResNet-BiLSTM-Attention model is a black-box model, which means that the decision-making process is not easily interpretable. The `DecisionTreeClassifier` will thus from now on often be referred to as 'white-box model' or 'baseline model'. The ResNet-BiLSTM-Attention model will be referred to as 'black-box model'. The implementation of both models is described in detail in the following sections.

The general modelling problem is as follows: The models receive a binary classification

[2] The XOR split means mutually exclusive execution paths, where exactly one path is chosen from multiple alternatives. For the given DMFS, different product configurations may be produced by performing mutually exclusive process steps.

task. The task is to predict whether the process execution is 'valid' or not. Validity, in the sense, is also including verification and uncertainty quantification. If the row or process is not valid ('accurate'), the modeller has to further identify the root cause. The models thus serve as a diagnostic tool to conduct more in-depth analysis. UQ is achieved by conducting statistical tests like permutation testing to generate p-values. Verification is achieved by the manual efforts of the modeller[3]. The `is_valid` column in the OCEL format serves as the target variable. The models are trained on a subset of the data, and their performance is evaluated on a separate test set. The models are compared based on various metrics, including accuracy, precision, recall, AUC and F1-score, see **??**. The goal is to determine which model performs better in terms of predicting the validity of process executions. Both models are compared using the same metrics. The models are trained and evaluated using the same dataset, ensuring a fair comparison. The results are presented in **??**. The data has been sorted by `end_time` and grouped by `order_id` to ensure that the white-box model has a chance to learn sequential patterns as well. By nature, DTrees are not able to learn sequential patterns. The ResNet-BiLSTM-Attention model is able to. For the train-test split, a random split is used. Despite sequential nature, random splitting was prioritized to mitigate temporal bias from incomplete process traces (**morita2022investigation**). Temporal bias refers to wrongly guessed correctness of patterns which are induced by choosing wrong cut-off points for splits.

4.2.1 Whitebox Baseline Model

As a baseline model, a white-box model is implemented to provide a reference point for the performance of the ResNet-BiLSTM-Attention model. The white-box model is based on a simple Dtree classifier, which is interpretable and easy to understand. This model serves as a benchmark for evaluating the performance of the more complex ResNet-BiLSTM-Attention model. For the concrete implementation, the `DecisionTreeClassifier` from the `sklearn.tree` module is used (**Scikit-Learn**).

4.2.2 ResNet BiLSTM Multi-Head Self-Attention Network

As an alternative model, the ResNet-BiLSTM-Attention model is implemented, see **??**. This model combines the strengths of residual networks (ResNet) and bidirectional long short-term memory networks (BiLSTM) with multi-head self-attention mechanisms. The model is implemented using the PyTorch modules `torch.nn`, `torch.functional` and `torch.optim` (**PyTorch**). Before the data is ingested by the network, specific `DataSet` and `DataLoader` classes are implemented to handle the data. The `DataSet` class is responsible for loading the data and transforming it into a format suitable for the model. The `DataLoader` class is responsible for batching the data and shuffling it during training.

The architecture of the ResNet-BiLSTM-Attention model is shown in **??**. The `torch.nn` module integrates Bidirectional Long Short-Term Memory (BiLSTM) layers with Multi-Head Self-Attention and Residual Connections. The model architecture consists of several layers, including convolutional layers, BiLSTM layers, and attention mechanisms.

[3] Of course, this undermines our efforts to present a fully automatic VVUQ framework. A lot of time is saved in the time span when no validity breach is detected. This is the main advantage of this approach.

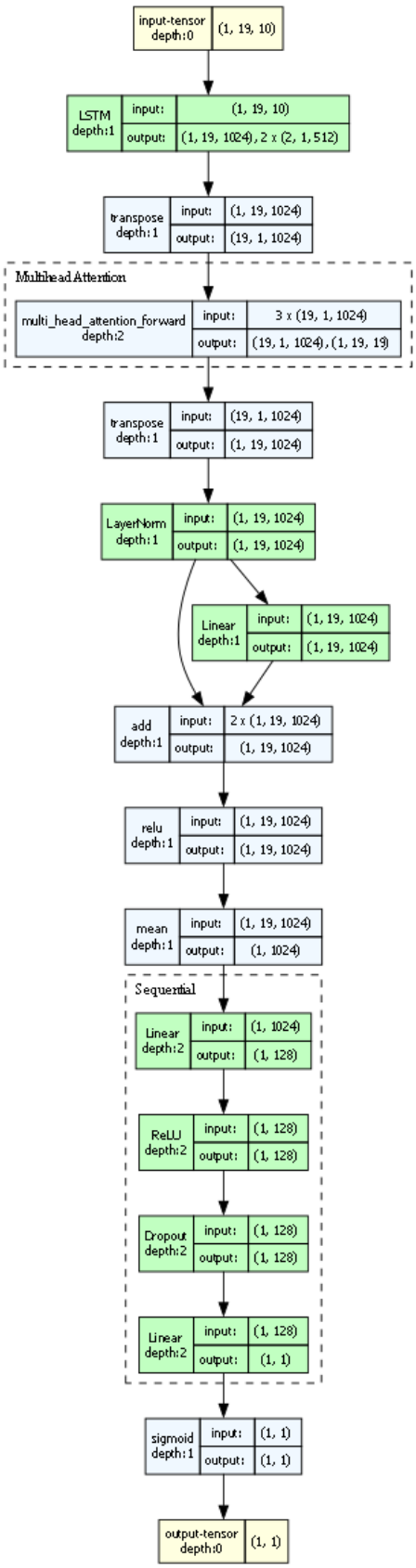

Figure 4.2: ResNet-BiLSTM-Attention architecture. Several layers are stacked, including BiLSTM, residual connections, and attention mechanisms. The model processes input sequences, applies self-attention, and outputs a single value for binary classification.

Source: Own Torchview illustration.

1. **Input/Configuration:** The model is initialized with hyperparameters defining its structure: `input_size` (number of input features, D), `hidden_size` (dimensionality of LSTM states, H), `num_layers` (number of stacked LSTM/Attention blocks, N), and `attention_heads` (number of parallel attention heads, A).
2. **BiLSTM Layers:** The core consists of N stacked BiLSTM layers (`nn.LSTM` with `bidirectional=True`). Each layer processes the input sequence in both forward and backward directions, capturing dependencies from past and future context. The output dimensionality of each BiLSTM layer is $2H$. Dropout is applied between layers for regularization.
3. **Multi-Head Self-Attention:** Following each BiLSTM layer, a `nn.MultiheadAttention` layer is applied. It performs self-attention on the BiLSTM output sequence, allowing the model to weigh the importance of different time steps relative to each other within the sequence.
4. **Layer Normalization & Residual Connections:** `nn.LayerNorm` is applied after the attention mechanism within each block to stabilize activations. A residual connection adds the input to this block to the output before a final `ReLU` activation, facilitating gradient flow in deeper networks.
5. **Sequence Pooling:** After the final LSTM/Attention block, the output sequence (shape: $Batch \times SequenceLength \times 2H$) is aggregated across the sequence dimension using temporal mean pooling (`torch.mean`), resulting in a single fixed-size vector (shape: $Batch \times 2H$) representing each input sequence.
6. **Final Classifier:** A feed-forward network (`nn.Sequential`) processes the pooled representation. It typically includes one or more linear layers with `ReLU` activations and Dropout for regularization, culminating in a final linear layer producing a single output logit.
7. **Output Activation:** A sigmoid function (`torch.sigmoid`) is applied to the final logit to produce a probability score between 0 and 1, suitable for binary classification. A threshold $\tau = 0.9$ discriminates the two classes (valid/invalid) during evaluation.
8. **Weight Initialization:** Linear layers within the network are initialized using Kaiming Normal initialization (`nn.init.kaiming_normal_`), a standard practice often beneficial for layers followed by `ReLU` activations.

For training, the model is set to training mode using `model.train()`. The model weights are updated using the *Adam* optimizer `torch.optim.Adam` (**??**, (**kingma2014adam**)) with an initial learning rate of 0.001. The loss function employed is binary cross-entropy (`nn.BCELoss`, **??**), which quantifies the difference between the predicted probabilities and the true binary labels. Input data is processed in sequences of length 19. To prevent overfitting, dropout with a probability of 0.3 is applied within the BiLSTM layer and also in the final fully connected sequence. The BiLSTM architecture itself consists of 1 layer (`num_layers=1`), with a hidden size of 512 per direction (`hidden_size=512`). A multi-head attention mechanism with 4 attention heads (`attention_heads=4`) is applied after the BiLSTM layer. The final classification head uses an intermediate dense layer with 128 units before the output neuron. See **??** and following for a description of the theoretical background.

Instead of a fixed learning rate, a learning rate scheduler, specifically *ReduceLROnPlateau* (`torch.optim.lr_scheduler.ReduceLROnPlateau`), adjusts the rate during training. This scheduler monitors the training loss and reduces the learning rate by a factor of 0.1 if the loss does not show improvement for 5 consecutive epochs (`patience=5`). The model is trained for a default of 10 epochs (`num_epochs=10`), iterating over the training dataset in mini-batches of size 32 (`batch_size=32`), with shuffling enabled for the training data. In each epoch, the model performs forward and backward passes to compute the loss and gradients, subsequently updating the model parameters via the optimizer. The training loss is tracked per epoch to monitor convergence.

During evaluation (as seen in `evaluate_model`), predicted probabilities are converted to binary labels using a threshold of $\tau = 0.9$. τ is chosen higher than the default because of the low tolerance for false positives (classifying simulated data as real) in the manufacturing context. Higher thresholds are commonly applied in anomaly detection or when the cost of a false positive is high. This value does not stand in contradiction to the rejection rate (RR) derived from the permutation testing (**??**, **??**). The threshold of 0.9 is a specific decision boundary for classifying individual instances post-training, impacting metrics like precision and recall at that specific cut-off point. In contrast, the permutation test assesses the overall statistical significance of the difference between the real and simulated data distributions, as captured by the model and features. It uses the ROC AUC score, which measures the models ability to distinguish between classes across *all possible thresholds*, not just one. The rejection rate RR then quantifies how consistently this statistically significant difference is found across multiple independent test runs. Therefore, a high RR indicates that the model can reliably detect differences between real and simulated data, regardless of the specific threshold 0.9 chosen for operational classification based on risk tolerance.

4.3 Model Evaluation

Based on the reference in **??**, the evaluation of the models is performed using various metrics. To achieve this, the model is switched to evaluation mode, `model.eval()`, to ensure deterministic behaviour. This disables dropout layers so that all neurons are used for the forward pass to achieve predictions. Gradient calculations are disabled through `torch.no_grad()`. `BatchNorm` layers now use their learned estimates of mean and variance parameters to process the data. During evaluation mode, the model processes the test set batch by batch, yielding output probabilities. These are compared against the true labels. The evaluation metrics are implemented in the `evaluate_model()` method. The evaluation metrics include a classification report, confusion matrix **??**, accuracy **??**, precision **??**, recall **??**, F1-score **??** and ROC AUC score **??**. The metrics are calculated using the `sklearn.metrics` module. The evaluation metrics are used to compare the performance of the baseline model and the ResNet-BiLSTM-Attention model as well as the standalone performance on the holdout set. The evaluation metrics are also used to diagnose the models and identify potential issues with respect to the bespoken adaptive feature selection procedure. For the assignment of the label 'valid' or 'invalid', a threshold τ of 0.9 is used. This value is higher than the usual threshold

of 0.5. This means that if the predicted probability is greater than or equal to $\tau = 0.9$, the process execution is classified as 'valid'. If the predicted probability is less than 0.9, the process execution is classified as 'invalid'. Setting the boundary so high is a conservative approach. It ensures that only the most confident predictions are classified as 'valid'. This is important for the VVUQ framework, as it aims to identify potential issues in the process execution. τ can be adjusted based on the specific requirements of the application and the desired trade-off between precision and recall.[4] The 0.9 validity threshold reflects manufacturing VVUQ's low tolerance for false positives **??**.

[4]For anomaly detection, setting τ higher is a common approach. In different contexts or sectors, lower thresholds may be better. A careful analysis of the ROC AUC curve may facilitate finding the best value.

Chapter 5: Empirical Validation

The thesis now focuses on the application of the implemented features and methods in a real-world scenario. The goal is to demonstrate the practical applicability of the developed concepts and to validate the theoretical findings presented in the previous chapters.

5.1 Physical Entity: Internet of Things Factory

The framework has been applied on the Internet of Things-Factory (IOT) in Gütersloh, Germany (**IoTFactory2024**). It is a cyber-physical system (CPS) (**baheti2011cyber**) mimicking industry-relevant processes in a smaller scale for research students. It consists of several stations that are partly interconnected via an assembly line or a delivery service conducted by automatic guided vehicles (AGVs). The factory is modular, so processes can be discovered module-wise in isolation. All modules are working on edge but are connected to a cluster that controls them. Theoretically, some stations can perform jobs of others. The main process also evaluated here is circular, meaning that the product is assembled and can be disassembled in a loop. The factory is shown in Figure **??**.

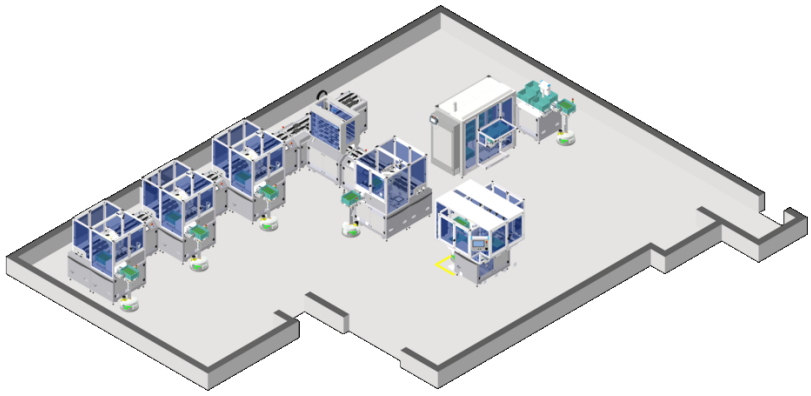

Figure 5.1: Overview of the IOT factory. It consists of three production stations from left to right, which are followed by a sorting station and a packaging station. The stations are interconnected by an assembly line. Isolated from the assembly part, two AGVs are used to transport parts between the warehouse station (upper right) and another flexible workstation (right).

Source: (**IoTFactory2024**)

The robot cells are responsible for performing transformation operations like assembling additional parts or testing functions.

The factory produces an exemplary product, consisting of a back part, a breadboard for several parts and a front panel. The parts to put on the breadboard are a display, gyroscope, an analog board and a weather station.

The only colour produced at the moment is black. Not all parts can be put on the breadboard and there are several parts which conflict in size and location on the breadboard. Back and

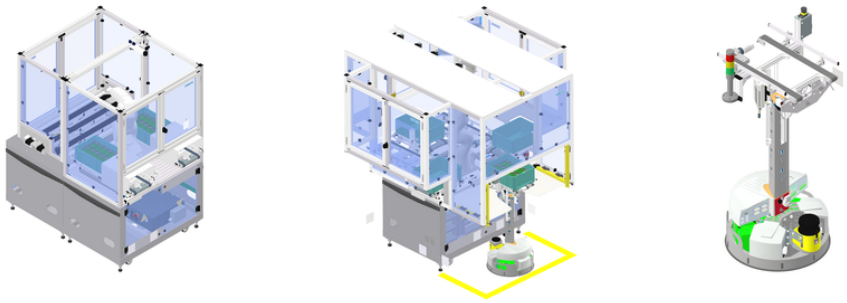

Figure 5.2: Two robot cells. The first cell is the main actor in this exemplary production process. Cell two is not part of the observed process here. The third image shows an AGV which transports boxes with assembled and disassembled parts to the stations.

Source: (**IoTFactory2024**)

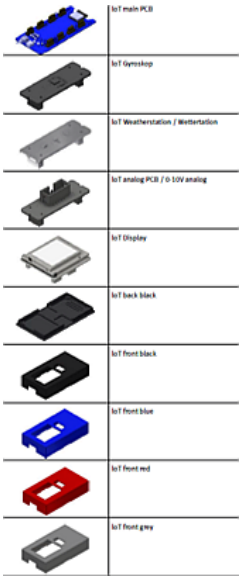

Figure 5.3: The product consists of a back part, a breadboard and a front panel. The breadboard is used to put on several parts, such as a display, gyroscope, analog board and a weather station.

Source: (**IoTFactory2024**)

front cover are necessary parts and are assembled every time. The main part `main_pcb` is also obligatory. It is placed at the beginning. Following that, the display or gyroscope can be assembled. If the display has been chosen, only the analog board or the weather station can be assembled. The weather station takes too much space on the breadboard for the gyroscope to be placed. If the analog board has been placed, the gyroscope is of course possible. The product is finished by placing the front cover and delivering the product to the sink.During the production, the factory gathers data via sensors. The data is saved in a database and can be used for further analysis. The database is relational and supports SQL queries.

Finally, the sequential order of the production process is as follows:

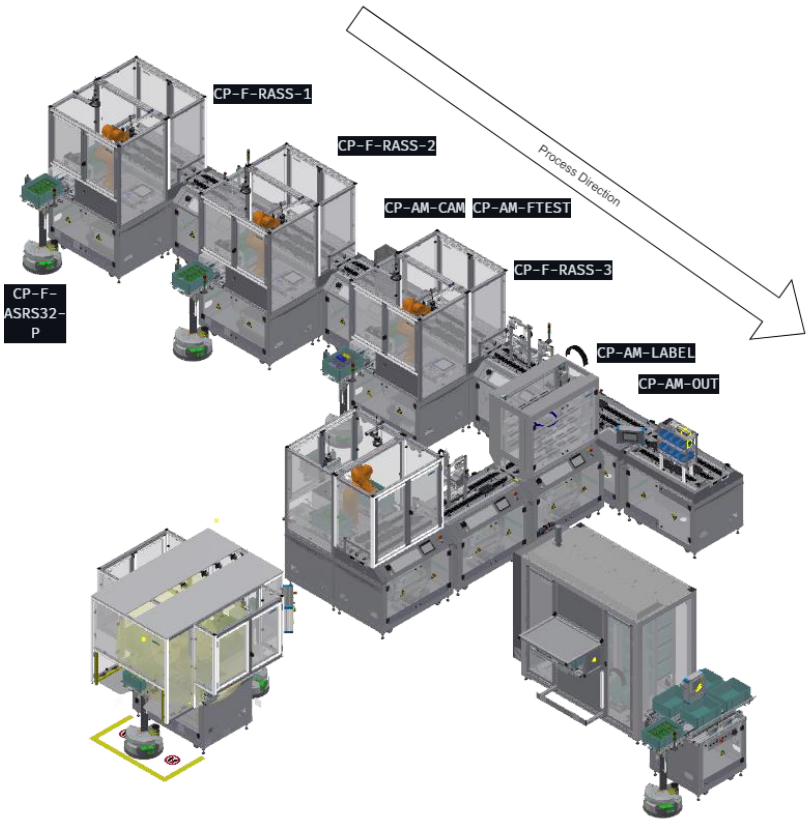

Figure 5.4: The blueprint with transitions between resources.
Source: (**IoTFactory2024**)

Factory Dataset

The dataset has been gathered from the database of the IoT Factory. In the code it is referred to as `real_data`.

The dataset contains 18057 rows and 4696 orders from a time span of 22.04.2020 to 14.05.2024. There are no duplicate rows. Each row describes a unique operation step in an order. Roughly 40 percent of the rows contain missing values, which have been filled with zeros. The data has been mined from a MariaDB SQL database of the IoT Factory. Several tables have been aggregated to form the dataset. Without additional tables, no information about workplans, resources and operation numbers would be available. The dataset is filtered for promising trajectories of meaningful and intact process executions. The filter considers data between the 20.04.2020 and 22.04.2022, 02.05.2022 and 19.07.2022, 02.11.2022, 11.11.2022, 23.01.2022 and March 2023, each data inclusively.

The dataset contains the following columns:

- **WPNo**: The workplan number of the specific workplan.
- **StepNo**: The step number of the specific workplan. The workplan is divided in sequential steps enumerated by this number.
- **ONo**: The order number of the specific order. Each order has a unique number.
- **OPos**: The order position of the specific order. The order position is the position of the order in the workplan.

- **Description**: The description of the specific step in spoken language.
- **OpNo**: The operation number of the specific operation. Each operation has a unique number.
- **NextStepNo**: The next step number of the specific step. This is the step that follows the current step.
- **FirstStep**: The first step of the specific workplan.
- **ErrorStepNo**: The error step number of the specific step. This is the step that is executed in case of an error.
- **Start**: The start time of the specific step.
- **End**: The end time of the specific step.
- **OPNoType**: The operation number type of the specific operation.
- **ResourceID**: The resource ID of the specific resource.
- **ErrorStep**: Is this step an error step (yes/no)?
- **ErrorRetVal**: The error return value of the specific step.
- **Active**: Is this step active (yes/no)?
- **op_desc**: The operation description of the specific operation, which is more specific than the description.
- **ResourceName**: The name of the specific resource.
- **resource_desc**: The description of the specific resource.
- **workplan_desc**: The description of the specific workplan.
- **workplantype_desc**: The description of the specific workplan type.
- **case_id**: The case ID of the specific case.
- **Description_Encoded**: The description of the specific step ordinally encoded.

The operations performed can be seen in the following:

1. **Release a part on stopper 1**: The AGV delivered the box with the parts to be manufactured to the first station. The box is travelling over the assembly line to the first station where the first part is released.
2. **Place cover to assembly place**: The first part is placed on the assembly place. The cover is placed on top of the part as the first piece of the product.
3. **Assemble part from box on RASS1 - MAIN PCB**: The main PCB is assembled on the cover. This is the first configured part of the product. The main PCB is the breadboard of the product.
4. **Switch on PCB**: The main PCB is switched on. This is necessary to activate it.
5. **Assemble part from box on RASS1 - DISPLAY**: The display is assembled on the main PCB. This is the second configured part of the product. The display is one optional part of the product. This factors out one product variant.
6. **Move part to pallet on belt**: The product is moved to the pallet on the belt for further processing.
7. **Measure a part (analog)**: The product is measured. This is necessary to ensure the quality of the product for later steps.
8. **Assemble part from box on RASS2 - ANALOG**: The analog part is assembled on

the product. This is the third configured part of the product. The analog part is also one optional part of the product.

9. **Assemble part from box on RASS2 - GYROSCOPE**: The gyroscope is assembled on the product. This is the fourth configured part of the product. The gyroscope is also one optional part of the product.
10. **Move part to pallet on belt**: The product is moved to the pallet on the belt for further processing.
11. **Check analog**: The analog part is checked. This is necessary to ensure the quality of the product for later steps.
12. **Check gyroscope**: The gyroscope is checked. This is necessary to ensure the quality of the product for later steps
13. **Assemble part from box on RASS3 - FRONT COVER**: The front cover is assembled on the product. This is the last configured part of the product. The front cover is the last part of the product.
14. **Move part to pallet on belt**: The product is moved to the pallet on the belt for further processing.
15. **Test connection to IoT main PCB**: The connection to the IoT main PCB is tested. This is necessary to ensure the quality of the product for later steps.
16. **Test the function of the touch display**: The touch display is tested. This is necessary to ensure the quality of the product for later steps.
17. **Test the analog input/output shield**: Analog PCB is tested.
18. **Test the historical gyroscope data**: The gyroscope is tested.
19. **Print Label**: The label is printed. The label contains information about the product configuration, the time manufactured and the serial number.
20. **Deliver Part**: The final product is delivered to the sink.

Per step, several resources are involved. The following table shows the utilized resources per step:

Table 5.1: Steps and Resources Used

Process Step Name	Resource Name
Release a part on stopper 1	CP-F-ASRS32-P
Place cover to assembly place	CP-F-RASS-1, CP-F-RASS-2, CP-F-RASS-3
Assemble part from box on RASS1 - MAIN PCB	CP-F-RASS-1
Switch on PCB	CP-F-RASS-1
Assemble part from box on RASS1 - DISPLAY	CP-F-RASS-1
Move part to pallet on belt	CP-F-RASS-1, CP-F-RASS-2, CP-F-RASS-3
Measure a part (analog)	CP-AM-MEASURE
Assemble part from box on RASS2 - ANALOG	CP-F-RASS-2
Assemble part from box on RASS2 - GYROSCOPE	CP-F-RASS-2
Move part to pallet on belt	CP-F-RASS-1, CP-F-RASS-2, CP-F-RASS-3
Check analog	CP-AM-CAM
Check gyroscope	
Assemble part from box on RASS3 - FRONT COVER	CP-F-RASS-3
Move part to pallet on belt	CP-F-RASS-1, CP-F-RASS-2, CP-F-RASS-3
Test connection to IoT main PCB	CP-AM-FTEST
Test the function of the touch display	CP-AM-FTEST
Test the analog input/output shield	CP-AM-FTEST
Test the historical gyroscope data	CP-AM-FTEST
Print Label	CP-AM-LABEL
Deliver Part	CP-AM-OUT

Source: **IoTFactory2024<empty citation>**

This process has been identified as the ground-truth process. The process is circular, meaning that the product is assembled and can be disassembled in a loop. The process is also modular, so that the product can be assembled in different configurations. The process is also flexible, the product can be assembled in different ways.

5.2 Digital Entity: Open Factory Twin

The Simulation-Based Digital Twin (SBDT) for the IoT Factory use case was developed using the Open Factory Twin (OFacT) framework (**ofact-intern**). OFacT is an open-source digital twin framework specifically designed for modelling, simulating, and controlling production and logistics environments. Its goal is to support system design, planning, and operational control during the entire lifecycle of such systems.

A principle of OFacT is the separation between the static description of the system and its

dynamic behaviour. This is achieved by distinguishing between:

- **State Model:** This component represents the static structure of the factory, its components (resources, parts, layout), their properties, relationships, and the potential processes or behaviours they can exhibit.
- **Agent Control:** This component implements the dynamic logic that controls the systems operation during simulation, making decisions about resource allocation, process execution sequences, and handling events based on the state model.

The construction of the State Model within OFacT uses structured input methods, such as Excel files, where different sheets correspond to specific classes within the OFacT metamodel, inspired by **schwede2024learning<empty citation>**. To model the IoT Factory scenario (**??**), the relevant components of the OFacT State Model were defined, including:

- **Plant**: The overall entity of production. The plant name used here was `iot_factory`.
- **EntityType**: All entities have to be defined here. This ranges from the parts, resources and AGVs to the factory itself.
- **StationaryResource**: Stationary resources are static and can not move. In this case, these are the RASS stations, measurement-, cam-, function test- and labelling station. The AGVs are not stationary resources, because they can move.
- **Storage**: Storage units contain parts. They are used to traverse the parts through the factory. In this context, the warehouse and box storage, have been modelled.
- **Warehouse**: This is a static storage unit where the parts are stored in boxes until they are processed.
- **WorkStation**: Workstations are resources which perform processes on parts. In our use case, these are the RASS stations.
- **ConveyorBelt**: There is one belt where the storage units traverse through the factory.
- **NonStationaryResource**: There are no non-stationary resources.
- **PassiveMovingResource**: One artificial passive moving resource has been mod
- **Process**: Contains processes and value added processes (VAP). VAP are adding features to parts and modify it. For each activity in the data a VAP has been created.
- **ProcessController**: This controller summarizes all processes to come and connects them.
- **ResourceModel**: Resource groups are formulated for activities like montage, identifying to attach the relevant parts to these resources. This way, main resources and parts are getting matched.
- **ProcessTimeModel**: Each part receives a time simple time distribution to account for its production time.
- **QualityModel**: Each part receives a bernoulli distribution to account for its quality. In the dataset, no quality information existed.
- **TransitionModel**: This model connects possible origins to possible destinations, to that the traversal of the parts can be modelled correctly. The packaging has been modelled as transition model.
- **TransformationModel**: This model contains an artificial transformation model.

- **Time**: Process execution plans get a starting time here.
- **Part**: Parts are connected to their EntityType here. There is also information attached where the part is stored or situated in.
- **Sales**: Lists the features and feature cluster. This matches the building rules. Parts have to be defined as features here.
- **CustomerGeneration**: Customer generation logic.
- **Customer**: List of customers.
- **Orders**: The orders with their requested features.
- **Process**: Contains processes and value added processes (VAP). VAP are adding features to parts and modify it. For each activity in the data a VAP has been created.
- **TransitionModel**: This model connects possible origins to possible destinations, to that the traversal of the parts can be modelled correctly. The packaging has been modelled as transition model.
- **TransformationModel**: This model contains an artificial transformation model.

By defining these elements according to the IoT Factory characteristics, a detailed static model was created within the OFacT framework. For simulation purposes, the state model then gets played out **??** with orders. The orders contain this model then served as the basis for running simulations to generate process execution data, forming the SBDT dataset used in this thesis for comparison against real-world data.

SBDT Dataset

The dataset gathered from the SBDT is referred to as `sim_data`. The OpenFactoryTwin provides a method to deserialize the simulated orders and save them in a CSV file. The dataset then gets converted to the OCEL structure. For this endeavour, a separate connector logic has to be implemented in every use case.[1]

The connector's output follows a standardized structure defined by the `OutputStructure` class, which ensures consistency across different data sources. The connector first deserializes the dynamic state model from a pickle file, accessing the full simulation state. Only actual process executions (not planned ones) are kept for further processing, identified by the 'ACTUAL' flag in their event type. The connector creates mapping dictionaries for various categorical attributes following the OCED standard. The categories can be defined by the modeller beforehand, which is the case here. The IDs are based on enumeration and default to -1 if the category has not been found in the list:

- **Part ID Mapping:** Uses domain expertise to identify part types from process names, normalizing text (lowercase, no whitespace) to match against a predefined list of possible parts. This list contains possible parts from **??**.
- **Process Type Categorization:** Assigns process steps to expert-defined categories such as 'machine,' 'feature,' 'endproduct,' 'test,' and 'transport' based on keywords in process names. These types have been assigned by the modeller and are of free choice.
- **Process ID Mapping:** Creates unique integer identifiers for each distinct process de-

[1] See `src/connector/ofact` for the connector here.

scription based on enumeration.
- **Resource ID Mapping:** Generates unique identifiers for each resource involved in the process executions based on enumeration.
- **Temporal Data Extraction:** The connector extracts start and end times for each process execution.

Process executions are associated with their respective order IDs, establishing the connection between individual process steps and the orders they belong to. All extracted information is then consolidated into a standardized DataFrame structure with properly typed columns as defined in the `OutputStructure` class. The connector also includes a validation step to ensure that the generated DataFrame adheres to the expected structure and data types, raising an error if any discrepancies are found.

A key aspect of the connector's functionality is the non-mandatory integration of domain knowledge into the data transformation process. Instead of relying only on the raw data structure, the connector employs expert-defined categorizations and normalization procedures to ensure semantic consistency in the transformed data.

For example, the part identification logic uses a predefined list of potential parts (such as 'GYROSCOPE,' "MAIN PCB," "FRONT COVER," etc.) and searches for these terms within process descriptions. Similarly, the process type categorization uses domain-specific groupings like 'machine,' 'feature,' and 'test' based on keywords found in process names. The `sim_data` rows will receive the label 0 for `is_valid`, because the black-box model should learn to distinguish between real and simulated data to perform VVUQ. The entries of the `real_data` receive 1 for `is_valid`.

5.3 Data Pipeline

Several preprocessing steps had to be performed to account for the fact that the SBDT simulated only one variant of the product: 'analog', 'cover', 'display', 'gyroscope', 'pcb' and the involved machines. This yielded the necessity to make both datasets congruent to each other. The following steps were performed:

- The `sim_data` dataset was aligned for the same time period as the `real_data` dataset. The SBDT chose the time of order as the production time. In the modelling phase, when adaptive feature selection had been performed to identify if the SBDT was able to learn the Time Model, only *relative* time features like duration were chosen which were developed in **??**.
- The `real_data` dataset was filtered for the same process steps as the `sim_data` dataset. This means that only the process steps which are present in the `sim_data` dataset were kept, producing only one product variant. This has also been applied on the part_id, the process_type and resource_id columns. The IDs based on enumeration had to be mapped to the original IDs in `real_data` to ensure that the correct IDs are used in the simulation. The mapping was done through the JSON files generated by the connector.
- Both datasets have been cleaned and entries containing invalid IDs were removed. This

means that all entries which are not present in the mapping dictionaries were removed. The mapping dictionaries are generated by the connector and contain only valid IDs per definition.
- Only a subset of all performed processes was included (in detail, all `process_id` ≤ 26). These processes all have the `process_type` 'machine', 'feature' or 'endproduct'. The processes with the `process_type` 'test' and 'transport' were removed. This was done to ensure that only the relevant processes are included in the dataset.

This cleaning step was not corrupting the data to facilitate easier learning of the models, it removed epistemic uncertainty. For example, some rows contained invalid names. The `real_data` dataset was then concatenated with the `sim_data` dataset. The concatenation was done by appending the `sim_data` dataset to the `real_data` dataset. The resulting dataset contains all process steps from both datasets. The resulting dataset is referred to as `final_data`. The unification before concatenation was necessary to ensure that no logical flaws are present in the data.

The `final_data` dataset after preprocessing as described contains 1978 rows and 56 orders with 24 features. The following section elaborates how these features were generated.

Feature Engineering

Several features have been engineered to assist both the whitebox and blackbox model on the validation task. The features were furher introduced to empower adaptive feature selection (AFS) regarding the model components of the SBDT (**schwede2024learning**). This way, if the features sufficiently explain the data, the conclusion can be made that the SBDT was able to learn the respective model component.

The following features were generated with domain knowledge:

- **KPIs**: Throughput, setup time, lead time and cycle time have been added to assist the VVUQ, see **??**. They allow PPC VVUQ to be performed. They further can be integrated to check if the Time Model has been learned.
- **duration**: The duration of the process step. This feature is important to understand how long the process step took. It is calculated as the difference between the start and end time of the process step.
- **sequence_number**: The sequence number of the process step. This feature is important to understand the order of the process steps. It is calculated by grouping the data by the process execution ID and then enumerating the process steps within each group based on the end time. This way, the sequence number enumerates each process step per order. It helps the `DecisionTreeClassifier` to learn the order of the process steps.
- **is_not_weekday**: A binary feature indicating whether the process step occurred on a weekend (1) or a weekday (0). No activities have been performed on weekends in the factory, so weekend activity is an anomaly.
- **is_break**: A binary feature indicating whether the process step occurred during a break (1) or not (0). No activities have been performed during breaks in the factory, so break

activity is an anomaly as well.

- **hour_of_day**: The hour of the day when the process step occurred. This feature is important to understand the time of day when the process step took place. It is calculated as the hour of the start time of the process step.
- **day_of_week**: The day of the week when the process step occurred. It is also calculated as the day of the week of the start time of the process step.
- **day_of_week_sin and day_of_week_cos**: A periodic time feature representing the sine and cosine of the day of the week, which helps capture weekly patterns in the data.
- **hour_of_day_cos and hour_of_day_sin**: A periodic time feature representing the cosine and sine of the hour of the day, which helps capture daily patterns in the data.

Days and hours are cyclical features that require special handling. Representing them as raw integers fails to capture their continuity (e.g., hour 23 is close to hour 0). To address this, the feature x with period P using sine and cosine functions has been transformed, effectively mapping it onto a unit circle:

$$x_{\text{sin}} = \sin\left(\frac{2\pi x}{P}\right) \quad ; \quad x_{\text{cos}} = \cos\left(\frac{2\pi x}{P}\right) \tag{5.1}$$

This provides a continuous, two-dimensional representation $(x_{\text{cos}}, x_{\text{sin}})$ that preserves the cyclical nearness of values.

In this work, this transformation is applied to:

- **Hour of Day ('hour_of_day'):** $P = 24$. Features: hour_of_day$_{\text{sin}}$, hour_of_day$_{\text{cos}}$.
- **Day of Week ('day_of_week'):** $P = 7$. Features: day_of_week$_{\text{sin}}$, day_of_week$_{\text{cos}}$.

Figure **??** illustrates this concept for the hour of the day. This encoding helps machine learning models better understand and utilize the cyclical nature of time.

5.4 Validation Methodology and Setup

With the integration of these features, AFS can be performed. For the statistical soundness of the analysis, a permutation test will be performed to assess the significance of the results. The following sections outline the testing procedure and the specific components of the SBDT that will be evaluated.

Permutation Testing for Statistical Significance

As laid out in the methodology chapter **??**, a permutation test will be conducted.

The null hypothesis (H_0) claims that the SBDT accurately represents the real system with respect to the features $\mathcal{F}_c$, meaning the data distributions are indistinguishable:

$$H_0 : \mathcal{D}_{real}(\mathbf{X}|\mathcal{F}_c) = \mathcal{D}_{sim}(\mathbf{X}|\mathcal{F}_c) \tag{5.2}$$

The alternative hypothesis (H_1) claims that the SBDT does *not* accurately represent the real

system, and the distributions are statistically distinguishable using the features $\mathcal{F}_c$:

$$H_1 : \mathcal{D}_{real}(\mathbf{X}|\mathcal{F}_c) \neq \mathcal{D}_{sim}(\mathbf{X}|\mathcal{F}_c) \tag{5.3}$$

Under this framework, if a classifier trained on the feature set $\mathcal{F}_c$ achieves performance significantly better than chance at distinguishing between real ($y = 1$) and simulated ($y = 0$) data, it provides evidence to reject H_0 in favour of H_1. This would imply that the SBDT component c has *not* been learned accurately, as detectable discrepancies exist. Conversely, if the classifier performs poorly (close to random chance), it fails to reject H_0, suggesting that, based on the features $\mathcal{F}_c$, the simulated data is consistent with the real data for that component.

Testing the SBDT Components

To rigorously evaluate the SBDTs fidelity concerning different process aspects, the thesis implements the hypothesis testing framework outlined previously (testing H_0: **??** against H_1: **??**). This involved using machine learning classifiers to determine if statistically significant differences exist between the real process data ($\mathcal{D}_{real}$, labelled $y = 1$) and the simulated data ($\mathcal{D}_{sim}$, labelled $y = 0$) based on specific feature subsets ($\mathcal{F}_c$) corresponding to key SBDT model components.

The analysis was performed separately for distinct feature subsets $\mathcal{F}_c$, each clustered to reflect the behaviour exhibited by specific SBDT components. Based on the feature engineering described in **??**, several targeted feature sets were defined. The `time_model` subset encompasses temporal patterns and cyclical time representations, including `duration`, `sequence_number`, cyclical encodings (`hour_of_day_cos`, `hour_of_day_day_sin`, `day_of_week_cos`, `day_of_week_sin`), and temporal status indicators (`is_break`, `is_not_weekday`). For operational resource allocation, the `resource_model` subset includes categorical identifiers (`resource_id`, `part_id`, `process_id`). To capture product transformations, the `transformation_model` subset contains `part_id`, `process_id`, and `sequence_number`. Movement patterns are represented in the `transition_model` subset through `part_id`, `resource_id`, `sequence_number`, and `duration`. Process execution characteristics are encapsulated in the `process_model` subset via `process_id`, `duration`, and `sequence_number`. Additionally, the `kpi_based` subset includes performance indicators (`throughput`, `cycle_time_sec`, `lead_time_sec`, `setup_time_sec`). Finally, an `all_features` set combined all available engineered features to provide a comprehensive perspective.

Permutation Testing Procedure

The practical implementation of the test described in **??** followed these steps for each feature subset $\mathcal{F}_c$:

Algorithm 1 Multi-Run Permutation Testing

Require:

Concatenated Dataset $\mathcal{D}_{final}$ (Features $\mathbf{X}$, Labels $y \in \{0, 1\}$),
Feature Subsets for model components $\{\mathcal{F}_c\}_{c=1}^{C}$,
Runs n_{runs}, Permutations N, Significance level α, Model Type M_{type} ▷ DTree, BiLSTM

Ensure:

Results $R : \{c \mapsto (\bar{S}_{obs}, \sigma_{S_{obs}}, \bar{p}, RR)\}$ ▷ Map component to aggregated stats

$R := \emptyset$ ▷ Initialize results map
for all feature subset $\mathcal{F}_c$ for component c **do**
  $S_{obs\_list} := [\,]$; $p_{list} := [\,]$ ▷ Initialize arrays for run results
  for $run = 1$ to n_{runs} **do**
    $seed \leftarrow$ RandomSeed()
    $(\mathcal{D}_{train}, \mathcal{D}_{test}) \leftarrow$ Split$(\mathcal{D}_{final}, \mathcal{F}_c, seed)$ ▷ Split stratified on y
    $M \leftarrow$ Train$(M_{type}, \mathcal{D}_{train})$ ▷ Train on $(\mathbf{X}_{train}, \mathbf{y}_{train})$ using features $\mathcal{F}_c$
    $\hat{\mathbf{p}}_{test} \leftarrow M(\mathbf{X}_{test})$ ▷ Predict $P(y = 1|\mathbf{X}_{test})$ using trained model M
    $S_{obs} \leftarrow$ ROC_AUC$(\mathbf{y}_{test}, \hat{\mathbf{p}}_{test})$ ▷ Observed score
    $S_{perm\_list} := [\,]$
    for $i = 1$ to N **do**
      $\mathbf{y}^{(i)}_{test,perm} \leftarrow$ Permute$(\mathbf{y}_{test})$ ▷ Randomly shuffle test labels
      $S_{perm,i} \leftarrow$ ROC_AUC$(\mathbf{y}^{(i)}_{test,perm}, \hat{\mathbf{p}}_{test})$ ▷ Score vs fixed $\hat{\mathbf{p}}_{test}$
      Append $S_{perm,i}$ to S_{perm_list}
    end for
    $count_{ge} \leftarrow \sum_{i=1}^{N} \mathbb{I}(S_{perm\_list}[i] \geq S_{obs})$
    $p_{run} \leftarrow count_{ge}/N$ ▷ p-value for this run
    Append S_{obs} to S_{obs_list}; Append p_{run} to p_{list} ▷ Store run results
  end for ▷ End runs
  $\bar{S}_{obs} \leftarrow$ Mean$(S_{obs_list}|$not NaN$)$ ▷ Calculate mean of valid observed scores
  $\sigma_{S_{obs}} \leftarrow$ StdDev$(S_{obs_list}|$not NaN$)$ ▷ Calculate σ of valid observed scores
  $\bar{p} \leftarrow$ Mean$(p_{list}|$not NaN$)$ ▷ Calculate mean of valid p-values
  $RR \leftarrow$ Mean$(\mathbb{I}(p < \alpha) \mid p \in p_{list}, p \neq$ NaN$)$ ▷ Rejection Rate on valid p-values
  $R[c] \leftarrow (\bar{S}_{obs}, \sigma_{S_{obs}}, \bar{p}, RR)$ ▷ Store aggregated results for component c
end for ▷ End feature subsets
return R

1. **Data Splitting:** The combined dataset (`final_data`) was split into stratified training ($\mathcal{D}_{train}$) and testing ($\mathcal{D}_{test}$) sets, ensuring proportional representation of real ($y = 1$) and simulated ($y = 0$) data in both sets. A random seed was used for reproducibility within a single run.
2. **Model Training:** The chosen classifier (DTree or BiLSTM) was trained on $\mathcal{D}_{train}$ using only the features in $\mathcal{F}_c$.
3. **Observed Statistic Calculation:** The trained model was evaluated on the original test set $\mathcal{D}_{test}$. The performance metric, specifically the ROC AUC score **??**, was calculated and recorded as the observed statistic, S_{obs}.
4. **Null Distribution Generation (Permutation):** To generate the null distribution, N permutations were performed. Critically, an efficient approach was used, particularly

for the computationally intensive BiLSTM model:

- The trained model generated predictions (binary $\hat{\mathbf{y}}_{test}$) and class probabilities (specifically for the positive class, $\hat{\mathbf{p}}_{test}$) on the original test set $\mathcal{D}_{test}$ *once*.
- For each permutation $i = 1...N$: The true labels $\mathbf{y}_{test}$ of the test set were randomly shuffled, creating $\mathbf{y}^{(i)}_{test,perm}$.
- The permuted statistic $S_{perm,i}$ was calculated by comparing the shuffled labels $\mathbf{y}^{(i)}_{test,perm}$ against the fixed probabilities $\hat{\mathbf{p}}_{test}$ (i.e., calculating ROC AUC between $\mathbf{y}^{(i)}_{test,perm}$ and $\hat{\mathbf{p}}_{test}$). This avoids retraining the model for each permutation. (A similar principle was applied for the Decision Tree, comparing permuted labels against fixed predictions/probabilities).

5. **P-value Calculation:** The p-value was computed as the proportion of permutation statistics greater than or equal to the observed statistic, following **??**.

$$p = \frac{\sum_{i=1}^{N} \mathbb{I}(S_{perm,i} \geq S_{obs})}{N}$$

At the end of the runs, the mean observed statistic $\bar{S}_{obs}$, standard deviation $\sigma_{S_{obs}}$, mean p-value $\bar{p}$, and rejection rate RR were calculated. The mean p-value $\bar{p}$ was used to assess the significance of the results, with a threshold of $\alpha = 0.05$ for rejecting the null hypothesis in case of the DTree and $\alpha = 0.01$ for the BiLSTM model. Averaging p-values across runs ($\bar{p}$) is presented here as a naïve summary metric. This approach is generally *not* considered good statistical practice because it fails to formally control the overall Type I error[2] probability (especially when results from different runs are dependent) and leads to a loss of information compared to established p-value combination methods. Despite these drawbacks, it was computed here as a simple, preliminary indicator alongside the rejection rate. For a statistically rigorous analysis using appropriate methods, specifically the Cauchy Combination Test (CCT) for combining the individual run p-values and the Binomial test for formally evaluating the significance of the observed Rejection Rate (RR), please refer to Appendix **??**. Additional statistical tests have been performed there to support the findings presented in this chapter.

5.5 Results and Interpretation

The feature subsets are equal to the model components, as noted. The following UpSet plot summarizes the feature subsets used for the SBDT components.

The plot highlights that the time model has the largest number of unique features (6 features), consisting primarily of derived temporal sinusoidal and cosine encodings and status dummy variables. The KPI category also contains a distinct set of 4 unique features, representing specific performance metrics. These exclusive memberships indicate that temporal representation and performance indicators introduce the most diverse information into the overall feature space. Several other features show significant overlap, signifying their role as core concepts linking multiple perspectives. Key identifiers like process_id are shared across re-

[2]The error of wrongly rejecting the null hypothesis in favour of H_1, although H_0 is true.

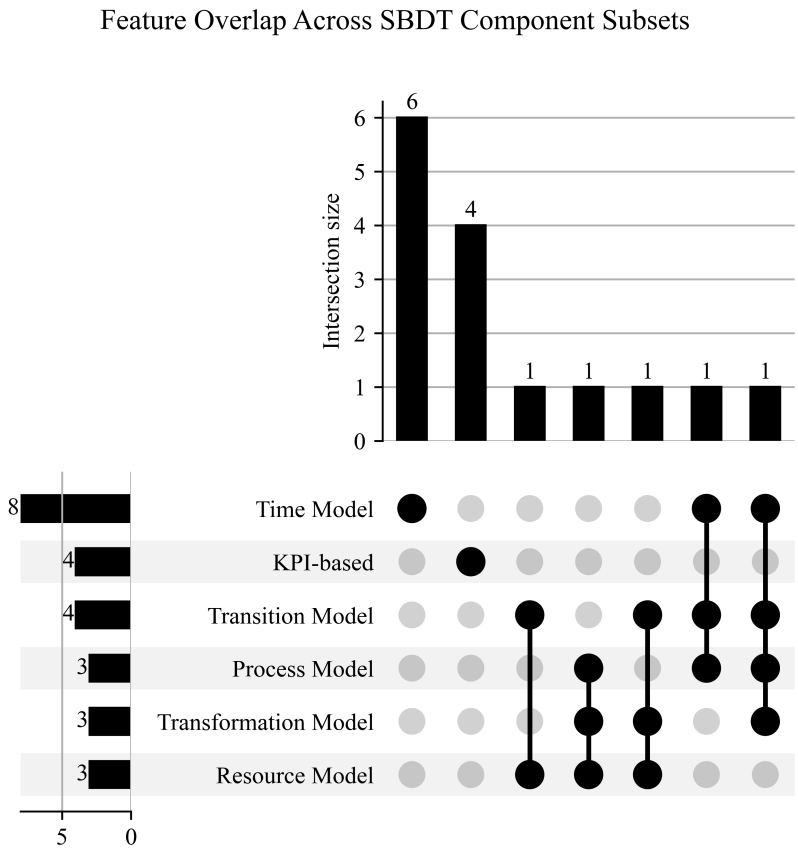

Figure 5.5: UpSet plot visualizing feature overlap across categories. Left bars show total features per category; top bars show counts for specific intersections defined by the dot matrix below.

Source: Own illustration.

source model, transformation model, and process model, while part_id links resource model, transformation model, and transition model. The feature sequence_number exhibits the highest degree of intersection among the detailed combinations, connecting time model, transformation model, transition model, and process model, underscoring its importance in relating temporal sequence to various operational views. Similarly, duration connects time model, transition model, and process model, and resource_id connects resource model and transition model.

Based on the aggregated results, the fidelity of the SBDT component corresponding to the feature set $\mathcal{F}_c$ was assessed. A high *rejection rate* RR (e.g., consistently above 0.5 across the 10 runs) was interpreted as strong evidence against the null hypothesis H_0. This indicates that the classifier could reliably distinguish between real and simulated data based on the features $\mathcal{F}_c$, suggesting that the corresponding SBDT component was *not learned accurately*. Conversely, a low rejection rate suggests insufficient evidence to reject H_0, implying the SBDT component might be *adequately learned*, or at least its potential inaccuracies were not detectable by the classifier using the given features.

Results of Whitebox Model

The whitebox validation was performed using a DTree classifier (**Scikit-Learn**). Key hyperparameters included limiting the tree complexity with a max tree depth of five using the

default gini criterion for node impurity and split evaluation. The Gini impurity $G(S)$ for a set of samples S at a node measures the probability of incorrectly classifying a randomly chosen element if it were randomly labelled according to the distribution of labels in the set. For K classes ($K = 2$ here), with p_k being the proportion of samples belonging to class k in S, it is defined as:

$$G(S) = 1 - \sum_{k=1}^{K} p_k^2 \tag{5.4}$$

A lower Gini impurity indicates a purer node. The quality of a potential split, which divides the set S into subsets S_{left} and S_{right}, is then evaluated based on the weighted average impurity of the child nodes, often referred to as the Gini split index:

$$\text{Gini}_{\text{split}}(S) = \frac{|S_{left}|}{|S|} G(S_{left}) + \frac{|S_{right}|}{|S|} G(S_{right}) \tag{5.5}$$

The decision tree algorithm seeks splits that minimize this value (**breiman1984classification**). Permutation testing was conducted with $N = 1000$ permutations per run over $n_{runs} = 10$ runs, using a significance level of $\alpha = 0.05$. Because inference costs with the DTree are low, label shuffling was performed on training and test set.

The aggregated results, including the mean ROC AUC score ($\bar{S}_{obs}$) with standard deviation ($\sigma_{S_{obs}}$), mean p-value ($\bar{p}$), and the Rejection Rate RR, are presented in Table **??**. The mean ROC AUC scores are also visualized in Figure **??**.

Table 5.2: Whitebox DTree validation results across 10 runs (N=1000, $\alpha = 0.05$), using paragraph-based layout.

Component ($\mathcal{F}_c$)	$\overline{\text{ROC AUC}}$	$\sigma_{\text{ROC AUC}}$	$\bar{p}$-value	RR	Assessment
`time_model`	1.0000	0.0000	0.0000	1.00	INACCURATE
`resource_model`	0.9832	0.0104	0.0000	1.00	INACCURATE
`transformation_model`	1.0000	0.0000	0.0000	1.00	INACCURATE
`transition_model`	1.0000	0.0000	0.0000	1.00	INACCURATE
`process_model`	1.0000	0.0000	0.0000	1.00	INACCURATE
`kpi_based`	1.0000	0.0000	0.0000	1.00	INACCURATE
`all_features`	1.0000	0.0000	0.0000	1.00	INACCURATE

Source: Own tabulation.

The results from the whitebox model are highly significant. For almost all feature subsets, the DTree achieved a perfect mean ROC AUC of 1.0000, with the exception of the `resource_model` subset which scored slightly lower but still extremely high ($\bar{S}_{obs} = 0.9832 \pm 0.0104$). Correspondingly, the mean p-values were effectively zero (< 0.0001, reported as 0.0000) and the rejection rate was 1.00 for all components across all 10 runs. This indicates that the DTree could easily and consistently distinguish the simulated data from the real data based on the features associated with every tested SBDT component. According to the interpretation framework, this implies that, from the perspective of the whitebox model, all

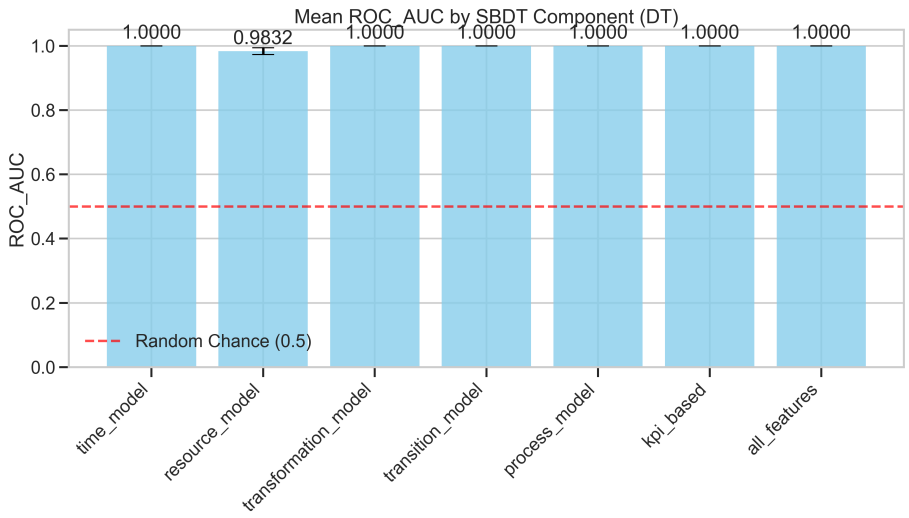

Figure 5.6: Mean ROC AUC scores achieved by the DTree classifier when distinguishing between real and simulated data, using feature subsets corresponding to different SBDT components. Scores averaged over 10 runs. The dashed red line indicates random chance (AUC = 0.5).

Source: Own illustration.

tested SBDT components were learned inaccurately. Given that the mean p-value for every feature subset is far below the significance level ($\bar{p} \approx 0.0000 < \alpha = 0.05$), and the rejection rate RR is 1.00 for all components, it rejects the null hypothesis (H_0) for every tested SBDT component based on the DTree. This provides statistically significant evidence that the model can reliably distinguish between the real process data and the simulated data using the features associated with the time model, resource model, transformation model, transition model, process model, and KPI-based perspectives, as well as when using all features combined.

The conclusion drawn solely from this whitebox analysis is therefore that the SBDT exhibits detectable discrepancies compared to the real system across all evaluated aspects. Consequently, based on the stringent criteria of the Decision Tree's ability to find differentiating patterns (even limited to $max_depth = 5$), all tested components of the SBDT are deemed inaccurate. This suggests that, at the level of detail captured by the engineered features and marked by the DTree model, the simulation does not adequately replicate the observed behaviour of the real IoT factory. The analysis with the blackbox model in the next section will explore whether a more complex model reaches similar conclusions.

Results of Blackbox Model

The blackbox validation employed the BiLSTM-based classifier described earlier. Training this model is computationally more intensive than the DTree. Therefore, while the number of permutations ($N = 1000$) and runs ($n_{runs} = 10$) were kept, the label shuffling for generating the null distribution was performed only on the test set labels for efficiency, comparing against the fixed predictions of the trained model. Furthermore, a stricter significance level of $\alpha = 0.01$ was chosen for this analysis. GPU acceleration via CUDA (**NVIDIA_CUDA**) was utilized to manage the computational cost.

The aggregated results from the blackbox models permutation tests are summarized in **??**. The mean AUC scores $\overline{\text{ROC AUC}}$ are visualized in **??**.

Table 5.3: Blackbox BiLSTM validation results across 10 runs (N=1000, $\alpha = 0.01$).

Component ($\mathcal{F}_c$)	**$\overline{\text{ROC AUC}}$**	**$\sigma_{\text{ROC AUC}}$**	**$\bar{p}$-value**	**RR**	**Assessment**
`time_model`	0.9998	0.0005	0.0000	1.00	INACCURATE
`resource_model`	0.9087	0.1618	0.0492	0.90	INACCURATE
`transformation_model`	0.8705	0.2094	0.1879	0.80	INACCURATE
`transition_model`	1.0000	0.0000	0.0000	1.00	INACCURATE
`process_model`	0.9788	0.0635	0.0000	1.00	INACCURATE
`kpi_based`	0.9989	0.0019	0.0000	1.00	INACCURATE
`all_features`	1.0000	0.0000	0.0000	1.00	INACCURATE

Source: Own tabulation.

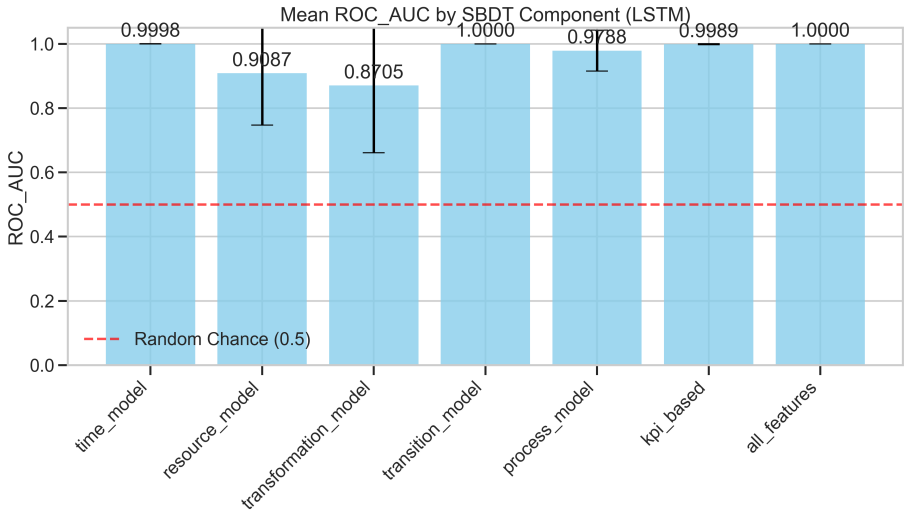

Figure 5.7: Mean ROC AUC scores achieved by the classifier when distinguishing between real and simulated data, using feature subsets corresponding to different SBDT components. Scores averaged over 10 runs. The dashed red line indicates random chance (AUC = 0.5).

Source: Own illustration.

The blackbox model results largely align with the whitebox findings, indicating significant discrepancies between the simulated and real data across various SBDT components. Focusing initially on the BiLSTM classifier (using $\alpha = 0.01$), it achieved very high mean ROC AUC scores $\overline{\text{ROC AUC}}$ for most feature subsets, particularly for `time_model` (0.9998), `transition_model` (1.0000), `process_model` (0.9788), `kpi_based` (0.9989), and `all_features` (1.0000). For these components, the mean p-values were effectively zero, and the rejection rate RR was 1.00, strongly indicating rejection of the null hypothesis (H_0).

For the `resource_model` and `transformation_model`, the mean ROC AUC scores were lower ($\bar{S}_{obs} = 0.9087$ and 0.8705) with higher variability ($\sigma_{S_{obs}} = 0.1618$ and 0.2094). While the *mean* p-values (0.0492 and 0.1879) exceeded $\alpha = 0.01$, the Rejection Rates (RR) remained very high ($RR = 0.90$ and $RR = 0.80$). This initial assessment, prioritizing the high RR, was subsequently confirmed by more rigorous statistical analyses detailed in appendix

??.[3] Specifically for the BiLSTM, both the CCT and the formal Binomial test applied to the rejection rates yielded highly significant results, supporting the rejection of H_0 for these components as well.

Furthermore, **??** also details the results for the DTree model (using $\alpha = 0.05$). The findings for the DT were even more uniformly decisive. **For every single tested component (`time_model`, `resource_model`, `transformation_model`, `transition_model`, `process_model`, `kpi_based`, and `all_features`), the Cauchy Combination Test yielded a p-value of 0.0000. Similarly, the Binomial test assessing the rejection rate was significant for all components, also yielding a p-value of 0.0000 and confirming a rejection rate of 1.00 across the board.**

Consequently, considering the strong performance metrics and consistent rejection of H_0 across runs for both models – BiLSTM (confirmed via CCT/Binomial tests in Appendix **??**) and Decision Tree (with universally significant CCT and Binomial test p-values of 0.0000 for all components, also detailed in Appendix **??**) – the analysis robustly concludes that statistically significant differences exist between the real and simulated data for all tested aspects.

Similar to the whitebox analysis, the conclusion drawn from the blackbox model is that the SBDT exhibits detectable inaccuracies compared to the real system across all evaluated components. The combination of high ROC AUC scores and high rejection rates, supported by the confirming significant results from the CCT and Binomial test analyses for both BiLSTM and DT models (Appendix **??**), provide robust evidence against H_0 for every component. Therefore, based on the consistent ability of both classifiers to distinguish the data, validated through multiple statistical approaches, all tested components of the SBDT are assessed as inaccurate. This supports the finding that the current SBDT configuration, as evaluated through the engineered features, does not sufficiently replicate the behaviour observed in the real IoT factory dataset.

5.6 Sanity Check

Given the high, near-perfect ROC AUC scores observed in the main experiments (**??**, **??**), this control experiment was performed to rule out the possibility that these scores resulted from data artifacts rather than genuine SBDT fidelity issues. This 'Identical Data Control Experiment" tests if classifiers can distinguish between identical datasets solely based on assigned labels ($y = 1$ versus $y = 0$), using the same methodology as the main experiments but on duplicated real and simulated data separately (**adebayo2018sanity**).

The results are summarized in **??**.

[3] The Cauchy Combination Test (CCT) combines the $k = 10$ p-values $(p_1, ..., p_k)$ from the individual runs. It tests the global null hypothesis $H_{0,Global}$: All individual null hypotheses $H_{0,i}$ (defined as $\mathcal{D}_{real}(\mathbf{X}|\mathcal{F}_c) = \mathcal{D}_{sim}(\mathbf{X}|\mathcal{F}_c)$ for run i) are true. A small P_{CCT} provides evidence against $H_{0,Global}$. The Binomial test assesses the observed rejection rate $RR = k_{obs}/k$. It tests the null hypothesis $H_{0,Binom}$: The true probability of rejecting $H_{0,i}$ in a single run is equal to α. The alternative is $H_{1,Binom}$: True rejection probability $> \alpha$. A small P_{Binom} indicates the observed RR is significantly higher than expected by chance under the null. See **??** and **??** for formal details.

Table 5.4: Results of Identical Data Control Experiments (Mean ROC AUC over 10 runs)

Classifier	Feature Set ($\mathcal{F}_c$)	Original AUC (Real vs. Sim)	Identical Real Data AUC (Real vs. Real-Copy)	Identical Simulated Data AUC (Sim vs. Sim-Copy)	Expected AUC (Random Chance)
Decision Tree	`time_model`	1.0000	0.3302	0.3545	≈ 0.5
Decision Tree	`resource_model`	0.9832	0.3916	0.4140	≈ 0.5
Decision Tree	`transformation_model`	1.0000	0.3903	0.4072	≈ 0.5
Decision Tree	`transition_model`	1.0000	0.3389	0.4079	≈ 0.5
Decision Tree	`process_model`	1.0000	0.3279	0.4107	≈ 0.5
Decision Tree	`kpi_based`	1.0000	0.3278	0.3788	≈ 0.5
Decision Tree	`all_features`	1.0000	0.3471	0.3763	≈ 0.5
BiLSTM	`time_model`	0.9998	0.4992	0.5008	≈ 0.5
BiLSTM	`resource_model`	0.9087	0.4989	0.5003	≈ 0.5
BiLSTM	`transformation_model`	0.8705	0.5015	0.4980	≈ 0.5
BiLSTM	`transition_model`	1.0000	0.4998	0.5007	≈ 0.5
BiLSTM	`process_model`	0.9788	0.4752	0.4977	≈ 0.5
BiLSTM	`kpi_based`	0.9989	0.4938	0.5027	≈ 0.5
BiLSTM	`all_features`	1.0000	0.5000	0.5000	≈ 0.5

Source: Own tabulation, incorporating results from provided summary and **??**, **??**.

As shown, both classifiers achieved ROC AUC scores near the random chance baseline of 0.5 when tested on identical data in contrast to the near-perfect scores achieved on the original real vs. simulated data comparison. This successful outcome strongly suggests the original high AUC scores were not primarily driven by artifacts. It validates the framework's core interpretation: the classifiers detected genuine, statistically significant differences between the real and simulated data. Therefore, the high AUC and RR values reported in **??** and **??** can be confidently interpreted as indicators of lower fidelity in the corresponding SBDT components, providing a reliable basis for identifying areas for model improvement.

To conclude, this chapter empirically validated the proposed VVUQ framework using the IoT Factory case study (**??**, **??**). The machine learning-based approach, employing both Decision Tree and BiLSTM classifiers with permutation testing (**??**), successfully identified statistically significant discrepancies between the SBDT and real-world data across all tested components (**??**, **??**). Crucially, identical data control experiments (**??**) confirmed these findings were not due to artifacts (**??**). The validation demonstrated the frameworks effectiveness in automatically assessing SBDT fidelity and providing component-specific feedback for improvement.

Chapter 6: Discussion

This chapter critically examines the findings from the empirical validation of the framework presented in **??**, connecting these results to the broader theoretical foundations established earlier in the thesis, see **??**. The IoT Factory case study provided valuable insights into the effectiveness of the machine learning-based approach to VVUQ of simulation-based digital twins, revealing both strengths and limitations of the proposed methodology.

6.1 Comparison with Traditional VVUQ Methods

Traditional VVUQ practices, as discussed earlier, distinguish between verification (ensuring the model is built correctly according to specifications) and validation (ensuring the model is an accurate representation of the real system for the intended purpose). Key techniques mentioned included code inspection, debugging, unit tests, statistical tests and sensitivity analysis, see **??**. While essential, applying these techniques comprehensively to complex SBDTs faces several challenges, also highlighted previously.

The proposed ML-based validation framework, utilizing classifiers and permutation testing on event log data, primarily addresses the validation aspect by comparing simulated outputs against real-world operational data. It aims to overcome some of the traditional challenges:

- **Effort and Scalability:** Traditional validation relying on expert consultation, manual log comparison, or setting up numerous simple experiments is often highly labour-intensive, see **??**. This manual effort negates some efficiency gains expected from SBDTs, especially if they are automatically generated or frequently updated. The ML approach, while requiring initial setup (data pipeline, feature engineering, model training), automates the core comparison task. Re-running the validation on new data or model versions requires mainly computational resources, addressing the identified key challenge scalability better than manual review which is hard to scale up.
- **Scope, Depth, and Objectivity:** Comparing aggregate KPIs via statistical tests or assessing face validity through expert consultation may fail to capture subtle discrepancies in process dynamics or interactions within the SBDT. The ML approach analyses event data (**??**) and potentially complex feature interactions (**??**), offering a deeper insight into the models dynamic fidelity. The whitebox results (**??**), where the DTree identified statistically significant discrepancies ($RR = 1.00$) across all components based on different feature sets, provide empirical evidence (**??**) for this potential depth. Furthermore, the use of permutation testing (**??**) yields objective p-values, contrasting with the subjectivity in expert consultation.
- **Applicability to Black-Box Models:** Techniques like code inspection are difficult for complex simulation platforms. The ML-based approach treats the SBDT primarily as a generator of output data. As long as comparable event logs can be extracted from the

simulation and the real system, the validation method can be applied, mitigating the challenge of limited applicability to black-box models. This meets the key challenge of applicability.

- **Continuous Validation:** Manual validation activities are often performed periodically. The automated nature of the ML pipeline lends itself better to more frequent or even continuous validation as new real-world data becomes available or the SBDT is updated, addressing a key limitation of traditional batch validation processes. The key requirement of concurrent validation is fulfilled.
- **Integration:** While integrating any V&V process presents challenges, embedding an automated data pipeline and ML testing framework might face different integration challenges compared to scheduling manual reviews or setting up traditional statistical reporting. However, once integrated, it can potentially offer more seamless feedback.

It is crucial to note that the proposed ML-based validation method primarily focuses on assessing the SBDT's fidelity against real-world data (validation). It does not replace the need for traditional *verification* techniques. Methods like code inspection, unit testing, and debugging remain essential for ensuring that the simulation model itself is implemented correctly and is free from bugs. Verification ensures the model is logically correct while the ML-based validation helps determine if it is the right model relative to observed reality.

6.2 Discussion in Context of Research Questions

The subsequent discussion evaluates the extent to which this objective was met by critically analysing the results of the case study (**??**) in the context of the research questions posed in **??**:

- **RQ1:** How can automated validation and verification processes for DTs be efficiently implemented to maintain accuracy?
- **RQ2:** Which data-driven approaches are best suited to identify discrepancies between simulated behaviour and real operational data in discrete material flow systems?
- **RQ3:** To what extent does the developed framework improve the quality and reliability of DTs compared to traditional V&V methods?

6.2.1 RQ1: Efficiency of Automated VVUQ Implementation

The practical efficiency of the given VVUQ framework depends on the given data quality, system architecture and modelling choices. In the specific IoT Factory use case, data was available and documented. In companies which are not data-driven or where gathering data is difficult, this framework might need time to be created and implemented. The data pipeline needed a lot of fine tuning and data expertise for it to work. Removing processes, false entries, and other data issues took a lot of time. The setup effort was higher than the execution effort of the model. Once the data pipeline was established, model training and inference was not time consuming. The largest amount of time was consumed in creating the hypothesis testing framework and assessing model quality. Referring back to the goal conflicts **??** in **??**, the framework was able to fulfil the scalability requirements with relatively moderate com-

putational costs and accuracy. Regarding the key challenges and requirements laid out in **??**, the problems of UQ and dynamic model adaption can be solved through the blackbox model. The p-values offer a quantitative measurement for UQ, while it is able to process online data (dynamic adaption). Model opacity was a trade-off with the BiLSMT structure, but may be solved if the modeller chooses the whitebox model. The key challenge of data dependency persists. During the development phase, nearly 50 hours were invested in creating the data pipeline and preprocessing. Importantly, the whitebox model could be used as a debugging tool for data pipeline evaluation.

To conclude and refer to **??** (also depicted above), the framework tackles the key challenges by providing two models that can be used based on validation goals. The whitebox model is able to provide a high level of interpretability and transparency, while the blackbox model is able to process large amounts of data and adapt to changing data. The effort of *verification* still has to be carried out manually. Regarding the outlined **??**, the given framework is interpretable through the hypothesis testing and AFS as well as the whitebox model, integratable, scalable and can be continuously validated. The key requirement of upholding data quality was not met, as a lot of effort has been spent on manual implementation and debugging. The two-fold model approach is able to provide a good trade-off between interpretability vs. performance/accuracy. The framework balances the outlined goals, but the data quality and data dependency are still to be addressed.

6.2.2 RQ2: Suitability of Data-Driven Approaches for Discrepancy Detection

A lot of work has been done on data-driven VVUQ approaches for all kinds of models, see **??** for example. The model choice of DTree and BiLSTM seemed arbitrary and inappropriate at first, but the quantitative results indicate that both models complement each other nicely by weighing out the weaknesses of the counterpart. The DTree was not able to capture the sequential patterns in the data quite well, as it splits the data into different branches. The BiLSTM on the other side interpreted the processes with a sequence length of 19 bidirectionally (past and future) and thus offers deeper insights. The OCEL format developed was able to capture complex dependencies in the data as it is able to store the data in a hierarchical structure and to model AND or OR processes. Different views on the processes are possible because of its object-centricness. Feature engineering highly improved the fitting of a discriminant rule of the DTree and increased the performance of the BiLSTM as well. During the engineering phase, the whitebox model can be plotted [1] and the feature importance can be calculated. The feature importance of the DTree was able to show which features were important for the model and thus helped to understand the model better. It also facilitated the design of the BiLSTM architecture by showing that temporal features regarding the time model and KPIs were the best features to split the data. The DTree can thus be used as a V%V tool by itself during the construction phase of the blackbox model by iteratively checking feature importance and its decision rules. This highly data-driven approach is a valuable tool for the modeller to perform a kind of sequential feature selection during feature engineering (**pudil1994floating**). By iteratively adding or removing features, AFS can be

[1] Scikit-Learn provides a method `plot_tree` which takes the DTree object and plots its decision rules

performed. Also, it performed as a debugging tool. For example, complete gini impurity could be achieved between the two classes by splitting the data for an order date, an artificial value for the KPI throughput, or processes with ID ≤ 27 which were not present in the simulated data. This indicates that the model was able to detect a bug in the simulation model. The BiLSTM on the other side was of course able to utilize these splits, but is opaque. A solution can be the application of SHAP or LIME.

DTree and BiLSTM both served as appropriate choices for a data-driven VVUQ framework for SBDTs. Permutation testing complements the framework by providing a statistical significance test for the model. The p-values are able to quantify the uncertainty of the prediction. Permutation testing is particularly useful here because the underlying distribution of the OCEL and decision boundaries are unknown and had to be approximated. It was thus perfectly suited for the given data complexity. For non-sequential data, the DTree might be sufficient for VVUQ. Using a whitebox model made it easier to debug and to perform VVUQ. OCED was a suitable data structure to capture the patterns in the data. While some information might not be translatable into the OCEL standard, it serves as a good starting point for modelling efforts. Later on, the framework can be enhanced.

6.2.3 RQ3: Improvement Compared to Traditional Methods

Mainly the validation process improves when using automated VVUQ approaches. The framework is able to ingest and process more information than domain experts. This offers a more comprehensive analysis than traditional checks focusing on aggregate KPIs or limited manual comparisons. The use of permutation testing (**??**) provides quantitative, objective measures of model fidelity, being in conflict with potentially subjective traditional assessments like face validity. Furthermore, the automated nature of the validation execution addresses scalability issues common in manual methods (**??**). However, this framework primarily enhances validation and supports rather than replaces essential verification techniques outlined in traditional approaches (**??**). While it effectively identifies that discrepancies exist, root cause analysis may require further investigation, and the initial setup effort (**??**) must be acknowledged.

In essence, the framework offers significant advancements in validation effectiveness, providing targeted, data-driven feedback to improve SBDT quality and reliability, particularly when integrated alongside traditional verification practices. AFS offers insights in which model component may be underrepresented in the SBDT and thus needs to be improved.

6.3 Implications of the Findings

The findings from this research carry several implications for both the theory and practice of developing and validating SBDTs, particularly those generated automatically.

6.3.1 Methodological and Theoretical Insights

This work contributes to the evolving field of VVUQ for complex simulation models and DTs. The successful use of a simplified OCEL format (**??**) demonstrates its potential beyond PM as a structure for validation data, proving effective for feature engineering and model input al-

though not fully complying to the OCED standard (**??**). The core theoretical contribution lies in the novel validation paradigm proposed and tested: a supervised classification approach where simulated and real data are distinctly labelled. In contrast to unsupervised methods measuring discrepancy magnitude, this framework trains a classifier to distinguish the two data sources. The innovation is the reversed interpretation of its performance (**??**): Low classifier accuracy (for example AUC ≤ 0.5) means high SBDT fidelity for the tested component relative to the feature subset, as the distributions are indistinguishable. Conversely, high accuracy indicates low fidelity—the twin failed to replicate the real complexity, making the distributions easily separable, leading to the rejection of the null hypothesis (H_0). This counter-intuitive interpretation, where high performance signals an inaccurately learned component, underpins the framework. It assumes that a valuable SBDT should mirror complex data patterns, more than simpler abstractions like DM or DS. While any virtual model is an abstraction, the primary abstraction considered here is the OCEL format, which is assumed to be complete for encoding reality in this context. Furthermore, the integration of permutation testing provides statistical rigor, yielding p-values and rejection rates RR that quantify the confidence in the validation assessment and offer inherent UQ. Hyperparameters N (permutation count) and α (significance) allow tuning the VVUQ process to specific domain needs, enabling stricter thresholds where required (e.g., healthcare). This component-based fidelity assessment using AFS provides more actionable feedback than simple static scores.

Overall, this framework refines V&V theory for automated SBDTs by changing focus from code verification to validating behaviour against data distributions, demonstrating the feasibility of the layered conceptual model (**??**) developed in **??** through its implementation (**??**).

6.3.2 Recommendations for Practical Application

The findings offer several recommendations for practitioners. Firstly, prioritizing data infrastructure and quality is important. Implementing automated ML-based validation requires robust data pipelines, standardized formats like OCEL, and rigorous quality checks (**??**), as significant setup effort may be involved (**??**). Secondly, leveraging domain knowledge remains crucial for effective feature engineering (**??**) to capture relevant process aspects aligned with SBDT components (**??**). Thirdly, employing a hybrid model approach is advantageous: Use interpretable whitebox models like DTree for initial insights, debugging, and guiding feature selection (**??**), supplemented by more complex blackbox models to potentially capture more intricate dynamics. Fourthly, utilize the objective fidelity metrics derived from the classification approach and permutation testing (ROC AUC, p-value, RR) (**??**) to enhance trust and credibility over subjective assessments (**??**). Fifthly, adopt component-wise validation using relevant feature subsets to gain targeted feedback for SBDT improvement. Sixthly, plan for integration with existing manufacturing systems (MES, PPC, SCADA), leveraging the framework's modular design and standard protocols (**??**) to enable continuous validation cycles. Finally, remember to complement this automated validation framework with traditional verification techniques (code review, unit testing) to ensure the SBDT is implemented correctly (**??**).

6.4 Limitations of the Study

Despite promising results, this study has limitations. Generalizability is constrained by the single case study. The IoT Factory (**??**) using the OFacT platform (**??**) is a laboratory framework designed for students to assess their work. Results may differ in other domains or with other SBDT tools. Data dependency is significant as well. The framework requires quality data, and the substantial preprocessing effort noted (**??**, **??**), including filling missing values with zeros (**??**), presents a practical challenge and potential point of influence on results. Methodological choices, such as the specific classifiers (DTree vs. BiLSTM), engineered features (**??**), and permutation test parameters (N, α) (**??**, **??**), impact outcomes and represent specific points in a larger space.

A key aspect is that manual verification remains the main limitation of this framework. Furthermore, interpretability of discrepancies poses a challenge; while the framework identifies inaccurate components (**??**), determining the root cause, especially with the BiLSTM, requires further investigation, potentially using XAI tools or expert analysis. Lastly, the reliance on a simplified OCEL format means the validation is performed on an abstraction of reality (**??**); detected differences could partially reflect format limitations, or the format might obscure other discrepancies.

6.5 Discussion Summary

In summary, this chapter critically discussed the thesis findings, connecting the empirical results from the IoT Factory case study to the theoretical foundations. The ML-based VVUQ framework demonstrated potential for improving the validation of SBDTs. The discussion contextualized the answers to the research questions, affirming the potential for efficient implementation despite initial setup costs **??**, validating supervised classification with a novel reversed interpretation as a suitable data-driven approach **??**, and confirming improvements in validation scope, depth, and objectivity over traditional methods **??**.

Chapter 7: Conclusion and Future Work

7.1 Summary of Key Findings and Contributions

This thesis addressed the critical challenge of efficiently validating and verifying automatically generated SBDTs for DMFS. The research was guided by three core questions **??**, which this work aimed to answer:

Regarding RQ1 **??**, the study demonstrated that while significant effort at the beginning is required for data pipeline setup and feature engineering, also concerning data quality, the operational execution of the proposed VVUQ framework can be highly efficient and scalable. The framework utilizes a two-model approach (whitebox DTree and blackbox BiLSTM) to balance interpretability and performance, automating the core validation and uncertainty quantification task once established. The efficiency stems from leveraging readily available simulation and real-world data, processed into an OCEL format.

In response to RQ2 **??**, the empirical validation confirmed that data-driven supervised classification is possible for identifying discrepancies between simulated SBDT behaviour and real operational data. By labelling simulated data differently from real data, a binary classifier is trained to distinguish between them. The innovation lies in the reversed interpretation of the classifier's performance: *Low* performance (AUC ≤ 0.5) implies that the SBDT component has been learned accurately relative to the tested features, as the simulated and real distributions are statistically indistinguishable. Conversely, *high* classifier performance indicates that the component was *not* learned accurately, as the distributions are easily separable. This contrasts with unsupervised approaches like **dos2024digital<empty citation>** which measure discrepancy magnitude with p-charts, and also runs counter to the intuition that high performance always signals success. Here, high performance signals high distinguishability, meaning the twin failed to replicate the complex manifold of the real data for that component. Permutation testing (**??**) provided a robust method for assessing the statistical significance (p-value) and consistency (Rejection Rate, RR) of this distinguishability. This approach makes the assumption that a successful SBDT must accurately reflect complex underlying data patterns, going further than simpler abstractions like DM or DS, and that the chosen OCEL format is suitable to encode the relevant aspects of reality.

Addressing RQ3 **??**, the developed ML framework demonstrated improvements over traditional V&V approaches (**??**). The case study (**??**) showed enhanced context and objectivity, systematically identifying inaccuracies across all tested SBDT components with high statistical significance (**??**, **??**). This automated VVUQ capability offers advantages in scalability and objectivity. The significance level (α) and number of permutations (N) can be tuned based on domain-specific requirements (e.g., lower α, higher N in safety-critical applications like healthcare).

The main empirical results from the IoT Factory case study (**??**) showed that both classifiers achieved high performance for most components, leading to the rejection of the null hypothesis (H_0) and the conclusion that these components were inaccurate in the current SBDT configuration. This highlights the frameworks sensitivity in detecting deviations and provides specific clues for targeted SBDT model improvement or recalibration

The primary contributions of this thesis are:

1. The development and conceptualization of a multi-layered, automated VVUQ framework (**??**) leveraging supervised classification for SBDT fidelity assessment.
2. A novel methodological interpretation where low classifier performance indicates high SBDT fidelity for specific components/features, combined with permutation testing for statistical rigor (validation) and uncertainty quantification (p-values, RR).
3. Empirical demonstration of the frameworks ability to target inaccuracies in an industrial SBDT case study (**??**).

However, a main limitation stays: The framework automates validation and uncertainty quantification but relies on *manual verification* to ensure the SBDT model is built correctly according to its conceptual description and specifications.

7.2 Concluding Remarks

The increasing adoption of SBDTs necessitates robust and efficient VVUQ methods to ensure trust and reliability. Traditional approaches face significant hurdles in scalability and objectivity when applied to complex, automatically generated twins. This thesis proposed and validated a novel, data-driven framework using supervised classification with a reversed interpretation of performance metrics, complemented by permutation testing. This approach offers a scalable, objective, and statistically grounded method for assessing SBDT fidelity component-wise. The empirical success in identifying specific areas of inaccuracy underscores the value of data-driven validation. While verification remains a manual task and data quality is paramount, this work contributes a significant step towards continuous, automated, and trustworthy validation of SBDTs, essential for their successful deployment in demanding Industry 4.0 applications.

7.3 Future Research Directions

Based on the findings and limitations identified (**??**), several avenues for future research emerge:

- **Enhancing Interpretability (Root Cause Analysis):** Integrate XAI techniques (SHAP, LIME) to move beyond identifying that a component is inaccurate (high classifier performance) to explaining why, thereby facilitating targeted model correction.
- **Refining Uncertainty Quantification:** Further explore methods like BNNs or MCD to provide richer UQ measures beyond the p-value/RR, potentially offering confidence intervals on the fidelity assessment itself.
- **Automated Recalibration and Verification Support:** Develop mechanisms for auto-

mated SBDT recalibration triggered by validation results. Explore how the frameworks outputs could potentially support or partially automate aspects of manual verification, addressing the main limitation.

- **Improving Online Capabilities and Integration:** Enhance the framework for robust real-time operation, optimizing data stream processing and developing standardized interfaces for seamless integration into industrial IoT environments.
- **Addressing Data Quality and Pipeline Automation:** Research more automated techniques for ensuring input data quality and streamlining the data pipeline setup, potentially leveraging AutoML or anomaly detection on the input data itself.
- **Investigating Foundational Assumptions:** Conduct studies on the impact of the OCEL format's completeness and the assumption regarding required SBDT complexity across different application domains.
- **Generalizability and Broader Application:** Validate the framework's effectiveness and the reversed interpretation hypothesis across diverse industrial domains, SBDT platforms, and levels of system complexity.
- **Exploring Alternative ML Approaches:** Investigate other promising ML techniques not yet applied in this specific concurrent VVUQ context. The Local Outlier Factor (LOF), a distance-based approach, could identify outliers potentially linked to rare events or concept drift (**alghushairy2020review**). Density-based methods like DBSCAN identify anomalies in low-density regions (**ccelik2011anomaly**), distinguishing normal clusters from deviations. Isolation Forest offers efficiency for high-dimensional data by isolating anomalies quickly through random partitioning (**xu2017improved**). One-Class SVMs (**li2003improving**) could be useful when only normal data is available, learning a boundary for normal operations. Autoencoders (AE) (**zhou2017anomaly**) trained on normal data can detect anomalies via high reconstruction errors, capturing complex non-linear patterns. Furthermore, semi-supervised techniques could leverage small amounts of labelled anomaly data alongside larger normal datasets to enhance detection.

Addressing these directions will further mature automated VVUQ methodologies, making SBDTs more reliable, trustworthy, and impactful in practice. ■

Eigenständigkeitserklärung

Hiermit versichere ich

Name, Vorname _____

Matrikelnummer _____

Studiengang _____

dass ich die vorliegende mit dem Thema

Automatic Verification and Validation of Automatically Generated Simulation-Based Digital Twins for Discrete Material Flow Systems

selbstständig und ohne die Benutzung anderer als der angegebenen Hilfsmittel angefertigt habe.
Alle Stellen – einschließlich Tabellen, Karten, Abbildungen etc. –, die wörtlich oder sinngemäß aus veröffentlichten und nicht veröffentlichten Werken und Quellen (dazu zählen auch Internetquellen) entnommen wurden, sind in jedem einzelnen Fall mit exakter Quellenangabe kenntlich gemacht worden.

Zusätzlich versichere ich, dass ich beim Einsatz von generativen IT-/KI-Werkzeugen (z. B. ChatGPT, BARD, Dall-E oder Stable Diffusion) diese Werkzeuge in einer Rubrik Übersicht verwendeter Hilfsmittel mit ihrem Produktnamen, der Zugriffsquelle (z. B. URL) und Angaben zu genutzten Funktionen der Software sowie Nutzungsumfang vollständig angeführt habe. Wörtliche sowie paraphrasierende Übernahmen aus Ergebnissen dieser Werkzeuge habe ich analog zu anderen Quellenangaben gekennzeichnet.

Mir ist bekannt, dass es sich bei einem Plagiat um eine Täuschung handelt, die gemäß der Prüfungsordnung sanktioniert wird.

Ich versichere, dass ich die vorliegende Arbeit oder Teile daraus nicht bereits anderweitig innerhalb oder außerhalb der Hochschule als Prüfungsleistung eingereicht habe.

Ort, Datum _____

Unterschrift _____

Übersicht verwendeter Hilfsmittel

Generative KI-Werkzeuge

Tool	Zugriffsquelle	Verwendungszweck
GitHub Copilot	https://github.com/features/copilot	Unterstützung bei der Programmierung von Python-Code für Datenverarbeitung und Modellimplementierung. Vorschläge für Codeergänzungen und Lösungsansätze.
ChatGPT	https://chat.openai.com	Formulierungshilfe für englische Textpassagen, Korrekturvorschläge für Grammatik, Umformulierung komplexer Sätze.
Grammarly	https://app.grammarly.com	Überprüfung der englischen Grammatik und Rechtschreibung im gesamten Dokument.

Verwendungsumfang

Die generativen KI-Werkzeuge wurden wie folgt eingesetzt:

- **GitHub Copilot**: Hauptsächlich zur Unterstützung bei Implementierungsdetails im Python-Code für die Datenverarbeitung und Modellierung. Alle generierten Vorschläge wurden manuell überprüft, verstanden und bei Bedarf angepasst. Verwendung in ca. 35% der gesamten Codeimplementierung.
- **ChatGPT**: Zur sprachlichen Verbesserung komplexer wissenschaftlicher Formulierungen, insbesondere bei methodologischen Beschreibungen und Diskussionen. Alle Formulierungsvorschläge wurden kritisch geprüft und in den Kontext der eigenen Argumentation integriert. Unterstützung bei ca. 30% des geschriebenen Texts.
- **Grammarly**: Überprüfung der finalen Version auf grammatikalische und stilistische Fehler. Vorgeschlagene Korrekturen wurden manuell überprüft und nur bei Erhaltung des beabsichtigten wissenschaftlichen Inhalts übernommen.

Kennzeichnung generierter Inhalte

Alle durch KI-Werkzeuge vorgeschlagenen Formulierungen, die substantiell in den Text übernommen wurden, sind durch Fußnoten gekennzeichnet. Geringfügige grammatikalische oder stilistische Anpassungen wurden nicht separat markiert, da sie den Inhalt nicht wesentlich verändert haben.

Inhalte aus wissenschaftlichen Quellen und eigene Gedanken stellen den überwiegenden Teil der vorliegenden Arbeit dar. Die Verwendung generativer Werkzeuge diente ausschließlich

der sprachlichen und technischen Unterstützung und hat die eigene kritische Auseinandersetzung mit dem Forschungsthema nicht ersetzt.

Appendices

Chapter A: Mathematical Appendix

A.1 Mathematical Notation

This section provides a reference for the mathematical notation used throughout this thesis.

Table A.1: Mathematical Notation

Notation	Description
Basic Notation	
$\boldsymbol{x}$, $\boldsymbol{y}$, $\boldsymbol{z}$	Bold lowercase letters denote vectors
W, A, Q	Uppercase letters denote matrices
x_i or $(\boldsymbol{x})_i$	Subscript i denotes the i-th element of vector $\boldsymbol{x}$
W_{ij}	Subscripts i, j denote the element at row i, column j of matrix W
$\boldsymbol{h}_t$	Subscript t typically denotes time step or sequence position
$\mathbb{R}^n$	n-dimensional Euclidean space
$\hat{y}$	Predicted value (in contrast to true value y)
Mathematical Operations	
$\odot$	Hadamard (element-wise) product
$[\boldsymbol{a}; \boldsymbol{b}]$	Vertical concatenation of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
$\|\|\boldsymbol{x}\|\|_p$	L_p norm of vector $\boldsymbol{x}$
$\nabla J(\theta)$	Gradient of function J with respect to parameters θ
$d_{\text{Euclidean}}(\boldsymbol{a}, \boldsymbol{b})$	Euclidean distance between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
CosineSimilarity$(\boldsymbol{a}, \boldsymbol{b})$	Cosine similarity between vectors $\boldsymbol{a}$ and $\boldsymbol{b}$
Neural Network Components	
$\sigma(\cdot)$	Sigmoid activation function
$\tanh(\cdot)$	Hyperbolic tangent activation function
ReLU$(\cdot)$	Rectified Linear Unit activation function
softmax$(\boldsymbol{z})$	Softmax function applied to vector $\boldsymbol{z}$
$\vec{\boldsymbol{h}}_t$, $\boldsymbol{h}_t$	Forward and backward hidden states in Bi-LSTM
$\boldsymbol{h}_{\text{mean\_pool}}$	Result of mean pooling operation on sequence
$\boldsymbol{h}_{\text{max\_pool}}$	Result of max pooling operation on sequence
$\boldsymbol{\gamma}$, $\boldsymbol{\beta}$	Scale and shift parameters in normalization layers
Optimization	
η	Learning rate in optimization algorithms

Continued on next page

Table A.1 – continued from previous page

Notation	Description
λ	Regularization strength parameter
ϵ	Small constant added for numerical stability
$\boldsymbol{m}_t$, $\boldsymbol{v}_t$	First and second moment estimates in Adam optimizer
$\hat{\boldsymbol{m}}_t$, $\hat{\boldsymbol{v}}_t$	Bias-corrected moment estimates in Adam optimizer
Loss Functions and Performance Metrics	
BCE	Binary Cross-Entropy loss function
MSE	Mean Squared Error metric
MAE	Mean Absolute Error metric
MAPE	Mean Absolute Percentage Error metric
Precision	Precision metric in binary classification
Recall	Recall (sensitivity) metric in binary classification
F1	F1-score, harmonic mean of precision and recall
AUC	Area Under the Curve metric for binary classification
ROC	Receiver Operating Characteristic curve
TPR, FPR	True Positive Rate and False Positive Rate
TP, TN, FP, FN	True Positive, True Negative, False Positive, False Negative counts
Statistical Concepts	
μ, σ^2	Mean and variance of a distribution
$\mathcal{N}(\mu, \sigma^2)$	Normal (Gaussian) distribution with mean μ and variance σ^2
UCL, LCL	Upper and Lower Control Limits in statistical process control
p	Proportion (typically of anomalies) in statistical process control
H_0	Null hypothesis in statistical testing
α	Significance level in statistical testing
S_{obs}	Observed test statistic in permutation testing
N	Number of permutations in permutation testing
p-value	Probability of observing test statistic at least as extreme as S_{obs} under H_0
RR	Rejection rate of null hypothesis
Feature Engineering	
x_{sin}, x_{cos}	Sine and cosine transformations of cyclical feature x
P	Period of cyclical feature in time encoding
$\mathcal{F}_c$	Feature subset in feature selection methods
Attention Mechanism	
Q, K, V	Query, Key, and Value matrices in attention mechanism

Continued on next page

Table A.1 – continued from previous page

Notation	Description
d_k	Dimension of key vectors in attention mechanism
Dataset Notation	
$\mathcal{D}$	Dataset or distribution
$\mathcal{D}_{train}$, $\mathcal{D}_{test}$	Training and testing datasets
$\mathcal{D}_{real}$, $\mathcal{D}_{sim}$	Real-world and simulation datasets
Statistical Tests and Combinations	
p_i	Individual p-value from the i-th statistical test
k	Number of independent or related tests being combined
t_i	Cauchy-transformed p-value in the CCT method
T_{CCT}	Combined test statistic in the Cauchy Combination Test
P_{CCT}	Combined p-value from the Cauchy Combination Test
k_{obs}	Observed number of rejections across k tests
P_{Binom}	P-value from the Binomial test for rejection rates
$B(k, \alpha)$	Binomial distribution with k trials and success probability α

Source: Own tabulation.

A.2 Mathematical Foundations

This appendix provides formal definitions and illustrations for the core mathematical functions and operations referenced in the theoretical foundations chapter (Section **??** onwards), as well as other relevant mathematical concepts and techniques commonly encountered in machine learning, deep learning, optimization, and data handling that are pertinent to the methods employed in this thesis.

A.2.1 Basic Notation and Operations

As established at the beginning of the chapter, vectors are denoted by bold lowercase letters (e.g., $\boldsymbol{z}$) and matrices by uppercase letters (e.g., W). Vectors are assumed to be column vectors unless otherwise specified.

Matrix-Vector Multiplication:

Given a matrix $W \in \mathbb{R}^{m \times n}$ and a vector $\boldsymbol{x} \in \mathbb{R}^n$, their product is a vector $\boldsymbol{y} = W\boldsymbol{x} \in \mathbb{R}^m$, where the i-th element is calculated as:

$$y_i = \sum_{j=1}^{n} W_{ij} x_j \tag{A.1}$$

Vector Addition:

Given two vectors $\boldsymbol{y}, \boldsymbol{b} \in \mathbb{R}^m$, their sum is a vector $\boldsymbol{z} = \boldsymbol{y} + \boldsymbol{b} \in \mathbb{R}^m$, computed element-wise:

$$z_i = y_i + b_i \tag{A.2}$$

Element-wise (Hadamard) Product:

Given two vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^m$, their Hadamard product is a vector $\boldsymbol{c} = \boldsymbol{a} \odot \boldsymbol{b} \in \mathbb{R}^m$, computed element-wise:

$$c_i = a_i \cdot b_i \tag{A.3}$$

This operation is notably used in LSTM cells to apply gate activations (Section **??**).

Vector Concatenation:

Given two vectors $\boldsymbol{a} \in \mathbb{R}^{d_a}$ and $\boldsymbol{b} \in \mathbb{R}^{d_b}$, their concatenation $\boldsymbol{c} = [\boldsymbol{a}; \boldsymbol{b}]$ results in a vector $\boldsymbol{c} \in \mathbb{R}^{d_a + d_b}$ formed by stacking the elements of $\boldsymbol{b}$ below the elements of $\boldsymbol{a}$. This is used in Bi-LSTMs (Equation **??**) and Multi-Head Attention (Equation **??**).

Matrix Transpose:

The transpose of a matrix $A \in \mathbb{R}^{m \times n}$, denoted $A^T \in \mathbb{R}^{n \times m}$, is obtained by swapping its rows and columns:

$$(A^T)_{ij} = A_{ji} \tag{A.4}$$

This is used in the Scaled Dot-Product Attention formula (Equation **??**).

A.2.2 Activation Functions

Activation functions introduce non-linearity into neural network models, allowing them to learn complex patterns. They are typically applied element-wise to the output of a linear transformation $\boldsymbol{z} = W\boldsymbol{x} + \boldsymbol{b}$.

Sigmoid Function (σ):

The standard sigmoid function maps any real input to the range (0, 1). It is defined as:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{A.5}$$

Due to its output range, it is commonly used for gating mechanisms in LSTMs (Equations **??**, **??**, **??**) and for producing probabilities in binary classification outputs. A plot is shown in Figure **??**.

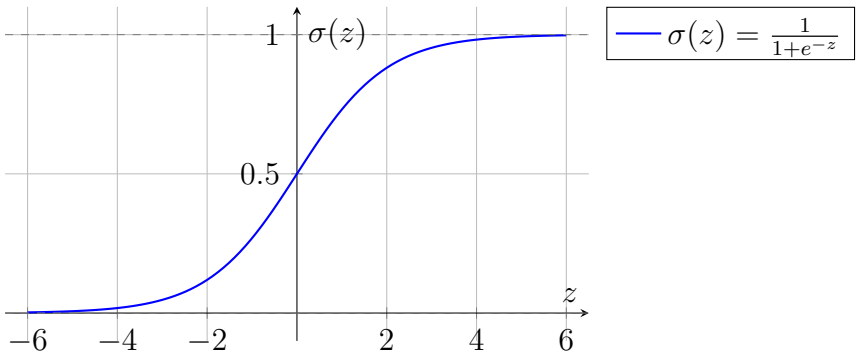

Figure A.1: The Sigmoid activation function.

Own illustration.

Hyperbolic Tangent Function (tanh):

The hyperbolic tangent function maps any real input to the range (-1, 1). It is defined as:

$$\tanh(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}} = 2\sigma(2z) - 1 \tag{A.6}$$

It is frequently used as the main activation function for hidden states in RNNs and LSTMs (e.g., Equations **??**, **??**, **??**). See Figure **??**.

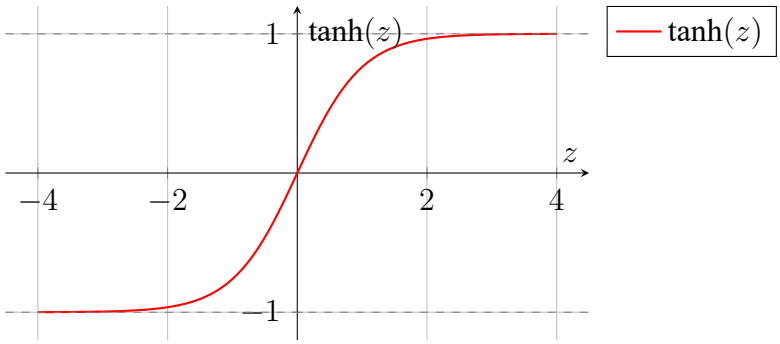

Figure A.2: The Hyperbolic Tangent (tanh) activation function.

Own illustration.

Rectified Linear Unit (ReLU):

The ReLU function outputs the input directly if it is positive, and zero otherwise. It is defined as:

$$\text{ReLU}(z) = \max(0, z) \tag{A.7}$$

ReLU is widely used in deep learning due to its simplicity and effectiveness in mitigating the vanishing gradient problem for positive inputs. It is used within the model presented in this thesis (Section **??** refers to the code using it). See Figure **??**.

Softmax Function:

The softmax function converts a vector of K real numbers $\boldsymbol{z} = (z_1, ..., z_K)$ into a probability distribution consisting of K probabilities. The function is applied to the entire vector and the

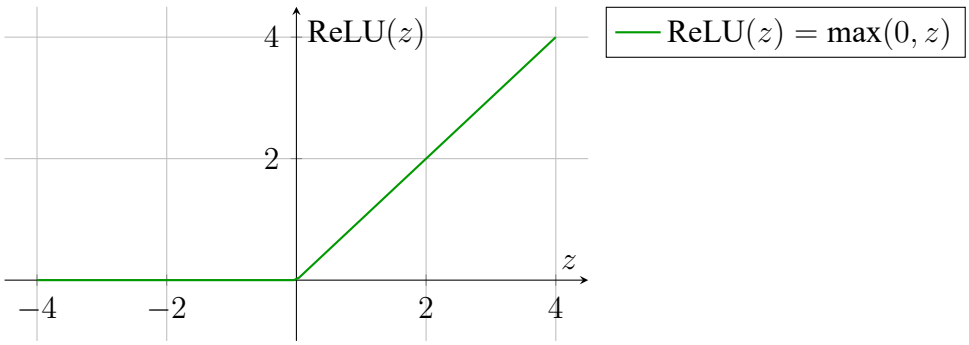

Figure A.3: The Rectified Linear Unit (ReLU) activation function.
Own illustration.

i-th element of the output vector is calculated as:

$$\text{softmax}(\boldsymbol{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, ..., K \tag{A.8}$$

The outputs are non-negative and sum to 1 ($\sum_{i=1}^{K} \text{softmax}(\boldsymbol{z})_i = 1$). Softmax is commonly used in the output layer of multi-class classification models and plays a crucial role in normalizing scores into attention weights in the Attention mechanism (Equation **??**).

A.2.3 Distance and Similarity Metrics

These metrics quantify the difference or similarity between vectors, which is fundamental in many machine learning tasks like clustering, nearest neighbour search, and evaluating embedding spaces. Let $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{R}^d$ be two vectors of dimension d.

Euclidean Distance (L2 Distance):

The most common distance measure, representing the straight-line distance between two points in Euclidean space. It is the L2 norm of the difference between the vectors:

$$d_{\text{Euclidean}}(\boldsymbol{a}, \boldsymbol{b}) = ||\boldsymbol{a} - \boldsymbol{b}||_2 = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2} \tag{A.9}$$

Manhattan Distance (L1 Distance):

Measures the distance by summing the absolute differences of the vector components. It corresponds to the distance travelled along axes at right angles (like navigating city blocks). It is the L1 norm of the difference:

$$d_{\text{Manhattan}}(\boldsymbol{a}, \boldsymbol{b}) = ||\boldsymbol{a} - \boldsymbol{b}||_1 = \sum_{i=1}^{d} |a_i - b_i| \tag{A.10}$$

Cosine Similarity:

Measures the cosine of the angle between two non-zero vectors, indicating their orientation similarity irrespective of their magnitude. It ranges from -1 (exactly opposite) through 0 (orthogonal) to 1 (exactly the same direction). It is calculated using the dot product and vector magnitudes (L2 norms):

$$\text{CosineSimilarity}(\boldsymbol{a}, \boldsymbol{b}) = \frac{\boldsymbol{a} \cdot \boldsymbol{b}}{||\boldsymbol{a}||_2||\boldsymbol{b}||_2} = \frac{\sum_{i=1}^{d} a_i b_i}{\sqrt{\sum_{i=1}^{d} a_i^2}\sqrt{\sum_{i=1}^{d} b_i^2}} \tag{A.11}$$

Cosine similarity is widely used for comparing high-dimensional vectors, such as text document embeddings or feature vectors, where magnitude might be less important than orientation. Note that while similarity increases as the value approaches 1, Cosine Distance is sometimes defined as $1 - \text{CosineSimilarity}(\boldsymbol{a}, \boldsymbol{b})$.

A.2.4 Attention Mechanism Components

The Scaled Dot-Product Attention mechanism (Equation **??**) relies on several fundamental operations beyond the Softmax function:

Dot Product Similarity:

The compatibility between a query $\boldsymbol{q}$ and a key $\boldsymbol{k}$ (both typically vectors of the same dimension d_k) is often computed using the dot product $\boldsymbol{q} \cdot \boldsymbol{k} = \boldsymbol{q}^T \boldsymbol{k}$. As noted above, this is closely related to Cosine Similarity but does not normalize for vector magnitudes. For matrices Q and K containing multiple queries and keys as rows, the matrix product QK^T computes all pairwise dot products efficiently.

Scaling:

To counteract the effect of large dot product values when the dimension d_k is high, the scores QK^T are scaled down by dividing by $\sqrt{d_k}$ before applying the softmax function. This helps maintain a stable gradient flow during training.

A.2.5 Normalization Techniques

Normalization layers help stabilize training, speed up convergence, and sometimes improve generalization by standardizing layer inputs.

Layer Normalization (LayerNorm):

Layer Normalization normalizes the inputs across the features for *each individual data sample* in a batch, making its computation independent of the batch size. For an input vector $\boldsymbol{x}$ representing the features for one sample at a specific layer, LayerNorm computes:

$$\text{LayerNorm}(\boldsymbol{x}) = \boldsymbol{\gamma} \odot \frac{\boldsymbol{x} - \mu_{\text{sample}}}{\sqrt{\sigma_{\text{sample}}^2 + \epsilon}} + \boldsymbol{\beta} \tag{A.12}$$

Here, μ_{sample} and σ^2_{sample} are the mean and variance calculated across the feature dimension(s) of the single input sample $\boldsymbol{x}$. $\boldsymbol{\gamma}$ (scale) and $\boldsymbol{\beta}$ (shift) are learnable affine transformation parameters of the same dimension as $\boldsymbol{x}$, and ϵ is a small constant added for numerical stability. LayerNorm is frequently used in RNNs and Transformers (including the model implemented in this work) where batch statistics might be less stable or meaningful.

Batch Normalization (BatchNorm):

(Included for contrast, delete if not relevant) Batch Normalization normalizes inputs across the *batch dimension* for each feature separately. It calculates the mean μ_{batch} and variance σ^2_{batch} for each feature across all samples in the current mini-batch. While highly effective in CNNs, its dependence on batch statistics can be less suitable for sequence models with variable lengths or small batch sizes compared to LayerNorm.

A.2.6 Pooling Strategies for Sequences

Pooling layers are used to aggregate information across the time or sequence dimension, often to produce a fixed-size representation from variable-length sequence outputs for downstream tasks like classification. Given a sequence of hidden states $H = [\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_T]$, where each $\boldsymbol{h}_t \in \mathbb{R}^d$:

Mean Pooling:

Calculates the element-wise average of the hidden state vectors across the sequence dimension:

$$\boldsymbol{h}_{\text{mean\_pool}} = \frac{1}{T} \sum_{t=1}^{T} \boldsymbol{h}_t \tag{A.13}$$

The resulting vector $\boldsymbol{h}_{\text{mean\_pool}} \in \mathbb{R}^d$ represents the average activation over the sequence. This strategy is used in the implemented model to summarize the output sequence before the final classification layer.

Max Pooling:

Calculates the element-wise maximum of the hidden state vectors across the sequence dimension:

$$(\boldsymbol{h}_{\text{max\_pool}})_j = \max_{t=1...T} (\boldsymbol{h}_t)_j \quad \text{for } j = 1, ..., d \tag{A.14}$$

The resulting vector $\boldsymbol{h}_{\text{max\_pool}} \in \mathbb{R}^d$ captures the strongest activation for each feature dimension across the sequence.

A.2.7 Weight Initialization

Initializing the weight parameters of a neural network appropriately is crucial for effective training, helping to prevent issues like vanishing or exploding gradients.

Kaiming (He) Initialization:

Proposed by He et al. (**he2015delving**), this method is primarily designed for layers followed by Rectified Linear Unit (ReLU) activations. It accounts for the non-linearity of ReLU. For Kaiming Normal initialization (used in the implemented model via 'kaiming$_n ormal\cdot), weights W$ are drawn from a normal distribution $\mathcal{N}(0, \text{std}^2)$, where:

$$\text{std} = \sqrt{\frac{2}{\text{fan\_in}}} \tag{A.15}$$

Here, 'fan$_i n$'$is the number of input units to the weight tensor. A variant considers the non-linearity slope a$ of leaky ReLU (where $a = 0$ for standard ReLU).

Xavier (Glorot) Initialization:

Proposed by Glorot and Bengio (**glorot2010understanding**), this method aims to keep the variance of activations and gradients roughly constant across layers, particularly effective for symmetric activations like tanh or sigmoid. For Xavier Normal initialization, weights W are drawn from $\mathcal{N}(0, \text{std}^2)$, where:

$$\text{std} = \sqrt{\frac{2}{\text{fan\_in} + \text{fan\_out}}} \tag{A.16}$$

'fan$_o ut$'$is the number of output units. Uniform versions also exist for both Kaiming and Xavier initializ$

A.2.8 Optimization Algorithms and Refinements

Optimization algorithms iteratively update model parameters θ to minimize a loss function $J(\theta)$.

Stochastic Gradient Descent (SGD):

A fundamental algorithm that updates parameters based on the gradient of the loss computed on a small batch (or single sample) of data at iteration k.

$$\theta_{k+1} = \theta_k - \eta \nabla J(\theta_k; \boldsymbol{x}^{(i)}; \boldsymbol{y}^{(i)}) \tag{A.17}$$

where η is the learning rate and $\nabla J(...)$ is the gradient computed on a mini-batch $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$. Variants include momentum or Nesterov momentum.

Adam Optimizer:

An adaptive learning rate optimization algorithm that computes individual adaptive learning rates for different parameters using estimates of first and second moments of the gradients (**kingma2014adam**). It often converges faster than basic SGD. The update rules involve computing biased moment estimates $(\boldsymbol{m}_t, \boldsymbol{v}_t)$, bias-corrected estimates $(\hat{\boldsymbol{m}}_t, \hat{\boldsymbol{v}}_t)$, and then

updating parameters:

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{\boldsymbol{v}}_t} + \epsilon}\hat{\boldsymbol{m}}_t \tag{A.18}$$

Adam is used for training the model in this work.

Learning Rate Scheduling:

Techniques used to adjust the learning rate η during training, which can improve convergence and final model performance.

- *ReduceLROnPlateau:* Monitors a specified metric (e.g., validation loss). If the metric does not improve for a defined number of 'patience' epochs, the learning rate is reduced by a multiplicative 'factor'. This strategy is employed in the training procedure of this thesis.
- *Step Decay:* Reduces the learning rate by a fixed factor every specified number of epochs.
- *Cosine Annealing:* Gradually decreases the learning rate following a cosine curve shape over a specified number of epochs or iterations.

A.2.9 Loss Functions

Loss functions quantify the difference between the models predictions and the true target values, guiding the optimization process.

Binary Cross-Entropy (BCE):

Used for binary classification tasks where the model outputs a probability $\hat{p}_i$ for the positive class (true label $y_i \in \{0, 1\}$). It is the loss function used in the model training for this thesis.

$$\text{BCE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{p}_i) + (1 - y_i)\log(1 - \hat{p}_i)] \tag{A.19}$$

A.2.10 Regularization Techniques

Regularization techniques are used to prevent overfitting by adding constraints or penalties to the model or its training process.

L2 Regularization (Weight Decay):

Adds a penalty to the loss function proportional to the squared magnitude of the model weights θ.

$$J_{reg}(\theta) = J(\theta) + \frac{\lambda}{2}||\theta||_2^2 = J(\theta) + \frac{\lambda}{2}\sum_i \theta_i^2 \tag{A.20}$$

where λ is the regularization strength. This encourages smaller weights.

Dropout:

During training, randomly sets a fraction of neuron activations (outputs) to zero at each forward pass before the subsequent layer (**srivastava2014dropout**). This prevents units from overly co-adapting and can be interpreted as training an ensemble of thinned networks. The dropout rate (e.g., 0.3 in the implemented model) specifies the probability of an element being zeroed out. At test time, dropout is turned off, and sometimes the outputs of the kept units are scaled down by the dropout rate (though often handled implicitly by libraries or absorbed into subsequent layers). Dropout is used in both the LSTM layers and the final fully connected block in the implemented model.

A.2.11 Data Handling for Sequences

Sequence Padding:

Neural networks typically require inputs within a batch to be tensors of uniform shape. Since sequential data (like manufacturing process steps or natural language sentences) often has varying lengths, techniques are needed to handle this during batch processing. Padding involves augmenting shorter sequences within a batch with special padding values (often zero) until they reach the length of the longest sequence in that batch. This results in rectangular tensors suitable for processing. The ‘collate_fn‘ used in the data loading pipeline for this thesis employs padding (via ‘pad_sequence‘). It is important that subsequent computations (e.g., loss calculation, attention mechanisms) are designed to ignore or mask these padded values to avoid introducing noise.

Chapter B: Miscellaneous

B.1 Design Science Research Methodology

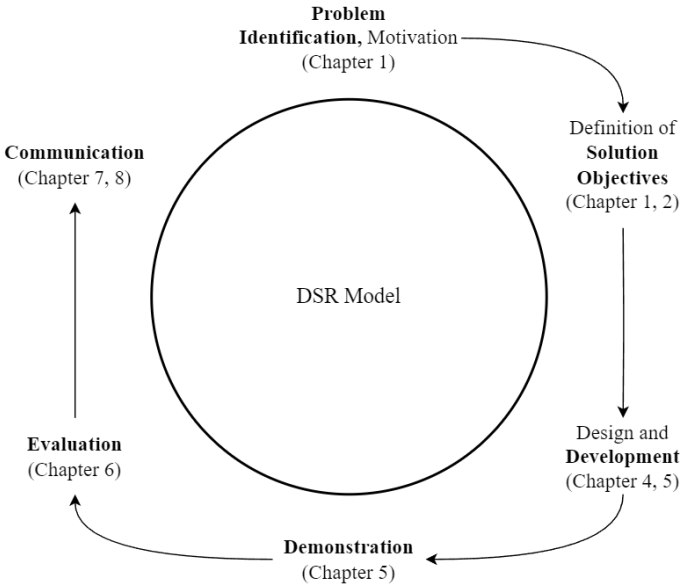

Figure B.1: The cyclical design science research model. The model consists of six steps. The problem identification (1) refers to the research gap in automated VVUQ of SBDT. Defining the solution objectives (2) specifies the research gap by formulating questions and hypotheses based on the theoretical foundations. The design and development (3) phase includes the development of the framework. The demonstration (4) phase shows the application of the framework in a case study. The evaluation (5) phase assesses the effectiveness of the framework. The communication (6) phase concludes the research by presenting the results.

Own illustration inspired by **peffers2007design<empty citation>**

B.2 Time Encoding on Unit Circle

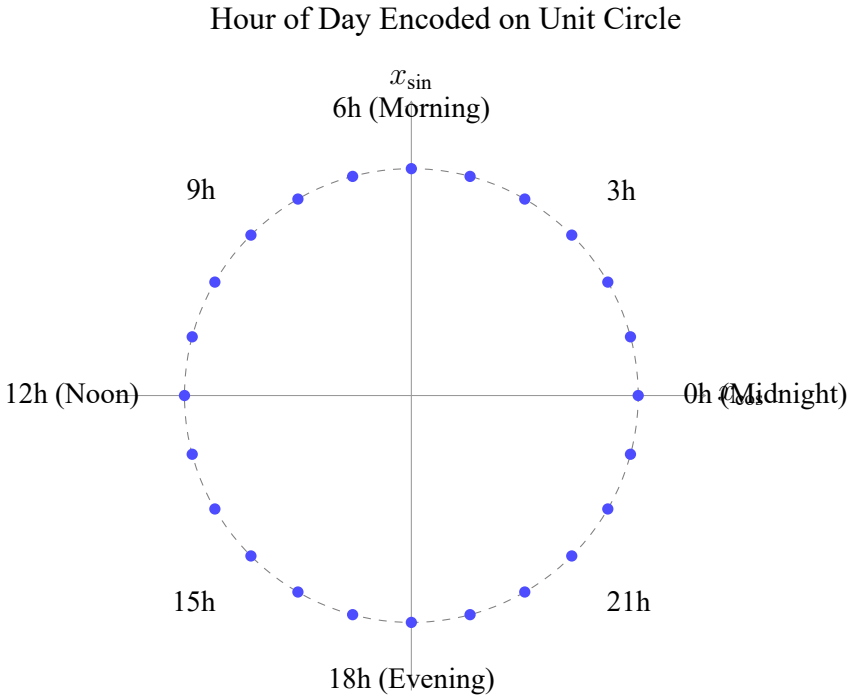

Figure B.2: Sine and Cosine transformation of the hour of the day (0-23). Each hour is mapped to a point $(x_{\cos}, x_{\sin})$ on the unit circle (blue dots). Key hours are labelled, demonstrating how the transformation preserves cyclical continuity (hour 23 is near hour 0).

Source: Own illustration

Chapter C: Additional Statistical Validation Methods

This chapter details the methodology for the Cauchy Combination Test (CCT) and the Binomial test for rejection rates, which were used to provide a more robust statistical assessment of the significance observed across multiple runs of the permutation tests described in Section **??** (referring to the main permutation testing section, adjust label if needed). These methods address the limitations of simply averaging p-values, particularly when dealing with potentially dependent results from repeated analyses on the same or related datasets.

C.1 Cauchy Combination Test (CCT)

The Cauchy Combination Test (**liu2020cauchy**) provides a powerful method for combining k individual p-values, $p_1, p_2, ..., p_k$, obtained from multiple statistical tests (in this case, the $k = 10$ runs of the permutation test for each feature subset) into a single overall p-value. A key advantage of the CCT is its robustness under arbitrary dependence structures among the individual p-values, which is particularly relevant when tests are performed on resampled versions of the same underlying dataset.

Methodology:

The CCT works by transforming each individual p-value p_i using the inverse cumulative distribution function (CDF) of the standard Cauchy distribution ($C(0, 1)$), which is equivalent to using the tangent function. Specifically, each p_i is transformed into a Cauchy-distributed variable t_i:

$$t_i = \tan\left((0.5 - p_i)\pi\right) \tag{C.1}$$

where π is the mathematical constant pi. Note that $p_i = 0.5$ transforms to 0, $p_i \to 0$ transforms to $t_i \to \infty$, and $p_i \to 1$ transforms to $t_i \to -\infty$.

The combined test statistic, T_{CCT}, is then calculated as the mean (or a weighted mean, though equal weights $w_i = 1/k$ are typically used) of these transformed values:

$$T_{CCT} = \frac{1}{k}\sum_{i=1}^{k} t_i = \frac{1}{k}\sum_{i=1}^{k} \tan\left((0.5 - p_i)\pi\right) \tag{C.2}$$

Null Distribution and Combined P-value:

A remarkable property of the Cauchy distribution is that the average of k independent standard Cauchy variables is itself a standard Cauchy variable. The CCT leverages the fact that

this holds approximately even under dependence. Therefore, under the global null hypothesis (that all individual null hypotheses corresponding to $p_1, ..., p_k$ are true), the test statistic T_{CCT} follows a standard Cauchy distribution, $C(0, 1)$.

The final combined p-value, P_{CCT}, is computed as the upper tail probability of the standard Cauchy distribution evaluated at the observed test statistic T_{CCT}:

$$P_{CCT} = P(C(0,1) \geq T_{CCT}) = 1 - F_{C(0,1)}(T_{CCT}) = 0.5 - \frac{\arctan(T_{CCT})}{\pi} \tag{C.3}$$

where $F_{C(0,1)}$ is the CDF of the standard Cauchy distribution. A small P_{CCT} indicates strong evidence against the global null hypothesis.

Advantages:

The CCT is computationally simple and does not require estimating dependence structures. It is particularly powerful when the signal against the null hypothesis is sparse (i.e., present in only a subset of the individual tests) and maintains good performance under various dependency scenarios.

C.2 Binomial Test for Rejection Rate

While the CCT combines the magnitude of the p-values themselves, the Binomial test provides a complementary perspective by formally assessing the significance of the observed *Rejection Rate* (RR). The RR is defined as the proportion of the k individual runs for which the null hypothesis H_0 was rejected at a pre-defined significance level α.

Purpose:

This test addresses the question: ”Is the number of times we rejected H_0 across the k runs significantly greater than what we would expect purely by chance if H_0 were true for all runs?”

Setup and Null Hypothesis:

Consider the k runs of the permutation test performed for a specific feature subset. Let α be the significance level used for each individual run (e.g., $\alpha = 0.05$ for DT, $\alpha = 0.01$ for BiLSTM). We define a ”success” for a single run if its p-value p_{run} is less than α ($p_{run} < \alpha$). Let k_{obs} be the observed number of successful runs (i.e., number of rejections) out of the total k runs. The Rejection Rate is $RR = k_{obs}/k$.

The null hypothesis (H_0) for the Binomial test is that the true underlying probability of success (rejecting H_0 in any single run) is equal to the significance level α. This assumes that under the global null, each run represents an independent Bernoulli trial with success probability α.

Methodology:

Under H_0, the random variable X representing the number of successful runs follows a Binomial distribution with parameters k (number of trials, i.e., runs) and α (probability of success):

$$X \sim B(k, \alpha) \tag{C.4}$$

The probability of observing exactly i successes in k trials is given by the probability mass function (PMF) of the Binomial distribution:

$$P(X = i|k, \alpha) = \binom{k}{i} \alpha^i (1 - \alpha)^{k-i} \tag{C.5}$$

where $\binom{k}{i} = \frac{k!}{i!(k-i)!}$ is the binomial coefficient.

Test P-value:

To assess whether the observed number of rejections k_{obs} is significantly high, we compute the probability of observing k_{obs} or more successes under H_0. This corresponds to a one-sided Binomial test p-value:

$$P_{Binom} = P(X \geq k_{obs}|H_0) = \sum_{i=k_{obs}}^{k} P(X = i|k, \alpha) = \sum_{i=k_{obs}}^{k} \binom{k}{i} \alpha^i (1 - \alpha)^{k-i} \tag{C.6}$$

A small P_{Binom} suggests that observing k_{obs} or more rejections is unlikely if the true rejection probability were only α, providing evidence that the observed RR is significantly higher than expected by chance and supporting the overall rejection of H_0 for that feature subset.

Utility:

This test directly quantifies the statistical significance of the consistency of rejecting H_0 across multiple independent or near-independent analyses, complementing methods like CCT that focus on the magnitude of the p-values.

C.3 Detailed Results for Decision Tree Model

The following table presents the detailed hypothesis testing results obtained using the Decision Tree (DT) classifier as the blackbox model for distinguishing between real and simulated data. The evaluation metric was the ROC AUC score. The results are based on $k = 10$ independent runs, each employing $N = 1000$ permutations for p-value calculation. The significance level for individual run rejection and the Binomial test was set at $\alpha = 0.05$. The Cauchy Combination Test (CCT) was used to combine p-values across runs (yielding P_{CCT}), and the Binomial test assessed the significance of the observed Rejection Rate (RR, yielding P_{Binom}).

Table C.1: Detailed Decision Tree validation results across 10 runs (N=1000, $\alpha = 0.05$).

Component ($\mathcal{F}_c$)	**ROC AUC**	$\sigma_{\textbf{ROC AUC}}$	P_{CCT}	**RR**	P_{Binom}	**Assessment**
time_model	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE
resource_model	0.9817	0.0088	0.0000	1.00	0.0000	INACCURATE
transformation_model	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE
transition_model	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE
process_model	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE
kpi_based	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE
all_features	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE

Source: Own tabulation based on permutation test results.

Summary of DT Results:

As evidenced by the results in Table **??**, both the Cauchy Combination Test ($P_{CCT} = 0.0000$) and the Binomial test ($P_{Binom} = 0.0000$) yielded extremely significant p-values for all feature subsets when using the Decision Tree model at $\alpha = 0.05$. The perfect rejection rate ($RR = 1.00$) across all components further underscores this. This indicates a highly consistent and statistically robust ability of the DT model to distinguish between the real and simulated data across all evaluated components based on the engineered features, leading to the assessment "INACCURATE" for all components according to the defined validation framework.

C.4 Detailed Results for BiLSTM Model

The following table presents the detailed hypothesis testing results obtained using the BiLSTM classifier as the blackbox model for distinguishing between real and simulated data. The evaluation metric was the ROC AUC score. The results are based on $k = 10$ independent runs, each employing $N = 1000$ permutations for p-value calculation. The significance level for individual run rejection and the Binomial test was set at $\alpha = 0.01$. The Cauchy Combination Test (CCT) was used to combine p-values across runs (yielding P_{CCT}), and the Binomial test assessed the significance of the observed Rejection Rate (RR, yielding P_{Binom}).

Table C.2: Detailed BiLSTM validation results across 10 runs (N=1000, $\alpha = 0.01$).

Component ($\mathcal{F}_c$)	**ROC AUC**	$\sigma_{\textbf{ROC AUC}}$	P_{CCT}	**RR**	P_{Binom}	**Assessment**
time_model	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE
resource_model	0.9945	0.0071	0.0000	1.00	0.0000	INACCURATE
transformation_model	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE
transition_model	0.9644	0.1067	0.0000	1.00	0.0000	INACCURATE
process_model	0.9714	0.0858	0.0000	1.00	0.0000	INACCURATE
kpi_based	0.9975	0.0054	0.0000	1.00	0.0000	INACCURATE
all_features	1.0000	0.0000	0.0000	1.00	0.0000	INACCURATE

Source: Own tabulation based on permutation test results.

Summary of BiLSTM Results:

As shown in Table **??**, the results for the BiLSTM model mirror the decisiveness observed with the Decision Tree. Using the stricter significance level of $\alpha = 0.01$, both the Cauchy Combination Test ($P_{CCT} = 0.0000$) and the Binomial test ($P_{Binom} = 0.0000$) yielded extremely significant p-values for all feature subsets. The rejection rate was also perfect ($RR = 1.00$) across all components. This demonstrates a consistent and statistically robust capability of the BiLSTM model to distinguish between the real and simulated data across all SBDT components evaluated, leading to the assessment ”INACCURATE” for all components within the validation framework.