Technische Hochschule Deggendorf
Fakultät Angewandte Informatik
Studiengang Master Kunstliche Intelligenz und Data Science

Generierung einer synthetischen Trainingsdatenquelle für ML-basierte Prozess-Mining-Tools

Generating a synthetic training data source for ML-based process mining tools

Masterarbeit zur Erlangung des akademischen Grades:
Master of Science (M.Sc.)
an der Technischen Hochschule Deggendorf

Vorgelegt von:
Anjali Singh
Matrikelnummer: 12203896

Am: 1. March 2024

Prüfungsleitung:
Prof. Dr. Andreas Fischer

Ergänzende Prüfende:
Zineddine Bettouche

Erklärung

Name des Studierenden: Anjali Singh

Name des Betreuenden: Prof. Dr. Andreas Fischer

Thema der Abschlussarbeit:

Generierung einer synthetischen Trainingsdatenquelle für ML-basierte Prozess-Mining-Tools

. . . .

. . . .

1. Ich erkläre hiermit, dass ich die Abschlussarbeit gemäß § 35 Abs. 7 RaPO (Rahmenprüfungsordnung für die Fachhochschulen in Bayern, BayRS 2210-4-1-4-1-WFK) selbständig verfasst, noch nicht anderweitig für Prüfungszwecke vorgelegt, keine anderen als die angegebenen Quellen oder Hilfsmittel benutzt sowie wörtliche und sinngemäße Zitate als solche gekennzeichnet habe.

   Deggendorf, ...
   Datum Unterschrift des Studierenden

2. Ich bin damit einverstanden, dass die von mir angefertigte Abschlussarbeit über die Bibliothek der Hochschule einer breiteren Öffentlichkeit zugänglich gemacht wird:

   ◯ Nein

   ◯ Ja, nach Abschluss des Prüfungsverfahrens

   ◯ Ja, nach Ablauf einer Sperrfrist von . . . Jahren.

   Deggendorf, ...
   Datum Unterschrift des Studierenden

Bei Einverständnis des Verfassenden vom Betreuenden auszufüllen:

Eine Aufnahme eines Exemplars der Abschlussarbeit in den Bestand der Bibliothek und die Ausleihe des Exemplars wird:

◯ Befürwortet

◯ Nicht befürwortet

Deggendorf, ...
Datum Unterschrift des Betreuenden

Abstract

The rapid expansion of process mining tools based on machine learning has transformed the capacity to derive valuable insights from intricate business processes. Nonetheless, the effectiveness of these tools heavily relies on the availability of high-quality training data, which often presents notable obstacles such as privacy restrictions, data scarcity, and data heterogeneity. This study investigates approaches to tackle the issue of data scarcity by suggesting the creation of synthetic training data through generative models. The proposed technique integrates a Generative Adversarial Network to explicitly capture the underlying distributional patterns of authentic process data. This strategy concentrates on generating new process instances that closely mirror the intricate characteristics and attributes of the real data. To comprehensively evaluate the effectiveness of this method, a comparative study is conducted to assess the performance of a Long Short-Term Memory (LSTM) model in contrast to the suggested Generative Adversarial Network (GAN) model, utilizing artificially generated process sequence data and authentic data. This comparative analysis is meticulously carried out on two widely used datasets in process mining: the BPI Challenge 2012 dataset and the Helpdesk dataset.

The study involves the selection and preprocessing of authentic process event logs, the development and training of a specialized GAN for generating process data, the assessment of the quality of synthetic data using statistical metrics through comprehensive experiments, and the examination of process workflow diagrams derived from both LSTM and GAN-generated data to evaluate their originality and accuracy. The results indicate that while the LSTM model accurately reproduces the initial data structure, the GAN introduces more variability, providing a wider range of training scenarios. This highlights the potential of utilizing GAN-generated data as a training resource for process mining tools based on machine learning, potentially enhancing their effectiveness and reliability by exposing them to various and realistic process patterns. Subsequent research could investigate the application of this method to process mining activities like analyzing customer journeys or detecting anomalies, exploring alternative generative models and evaluation techniques, and integrating domain-specific constraints and expert knowledge into the GAN framework to enhance the quality and usability of the generated data for advancements in process mining capabilities.

Contents

1 Introduction

Process mining is essential for enhancing business processes by extracting insights from event logs that document the sequence of activities in a process. It serves as a link between data science and process management, utilizing event logs to discover, monitor, and enhance real-world processes. In the dynamic realm of process mining, the capacity to analyze and enhance intricate processes heavily depends on the availability and quality of the underlying data. However, a significant challenge in this field is the restricted access to practical datasets that are comprehensive, diverse, and adhere to privacy laws.This shortage significantly impedes the advancement and validation of sophisticated process mining tools, especially those employing machine learning (ML) methodologies.
Synthetic data generation emerges as a promising solution, enabling the creation of realistic and controlled event logs without these limitations. Synthetic data generation techniques have been extensively explored in areas like computer vision; however, their application in the domain of process mining signifies an emerging and rapidly growing area of study. Synthetic data generated not only improves the quality of training data but also provides a solution to data privacy concerns, effectively addressing the issues posed by data scarcity.

1.1 Motivation

The application of machine learning in process mining has led to a new era of insights and optimizations for companies and organizations. The foundational need for process mining is the vast amount of data that detail every step, decision, and action within a business process. However, despite the critical importance of data in this field, its acquisition poses significant challenges, ranging from privacy concerns to data scarcity, which present research endeavors to address. Unlike domains such as image processing or natural language understanding, which have benefited from large public datasets like ImageNet and GLUE, process mining lacks such expansive, standardized datasets. The scarcity of data presents a significant obstacle to conducting process mining studies in various fields. Existing methodologies often do not capture the multifaceted nature of business processes and fail to adequately represent the variability and complexity inherent in real-world operations.

1.2 Objective

This thesis addresses the critical gap in the current landscape of process mining research by proposing an innovative approach to generate high-quality, realistic synthetic training data tailored for ML-based process mining tools. At its core, this study explores the creation of

synthetic datasets that not only accurately reflect the distribution patterns of actual process data, but also expand the training dataset with unique and diverse sequences. The fundamental strategy employed in this research involves the utilization of long-short-term memory (LSTM) networks and generative adversarial networks (GAN) using a transformer encoder as both a generator and a discriminator.
The generated synthetic data is expected to have the same distributions as real data. In addition, the size of the process data can be increased to include novel process sequences, eventually adding more diversity to the original data set. The research question addressed in this work is "*Can Generative Adversarial Networks and traditional LSTM models be used to generate synthetic process data, effectively and efficiently, while preserving the underlying distribution and patterns of the original data*".
The purpose of this thesis is to design, implement, and test a synthetic data generator based on Generative Adversarial Networks and LSTM. For a proof-of-concept, sample datasets are used to train the models of the proposed solution that subsequently will generate new synthetic data. The result is an analysis that compares both the LSTM and the GAN-based framework to conclude with the advantages and disadvantages of using Generative Adversarial Networks for Synthetic Data Generation.

1.3 Thesis Structure

This section outlines the organization of the chapters in this thesis. The background is covered in *Chapter 2* providing a thorough description of the main concepts and technologies relevant to this work, including Process Models, Synthetic data generation techniques using deep learning, and generative models such as LSTM, GAN, and VAE. *Chapter 3* discusses previous related work in this thesis. Methodology is the main focus of *Chapter 4*, which also explores the technical details of Generative Adversarial Networks (GAN) and clarifies the steps involved in data collection and preprocessing. In *Chapter 5*, the implementation steps for the generation of synthetic data using LSTM and GAN are described. *Chapter 6* investigates and compares the quality of the data generated by both models based on statistical similarity metrics followed by the analysis of the synthetic data generated using process workflow models.*Chapter 7* summarizes key findings and contributions and suggests potential avenues for future research in this innovative domain. In *Chapter 8*, the thesis provides its conclusions and outlines future research directions.

2 Background

2.1 Process Mining and Event Logs

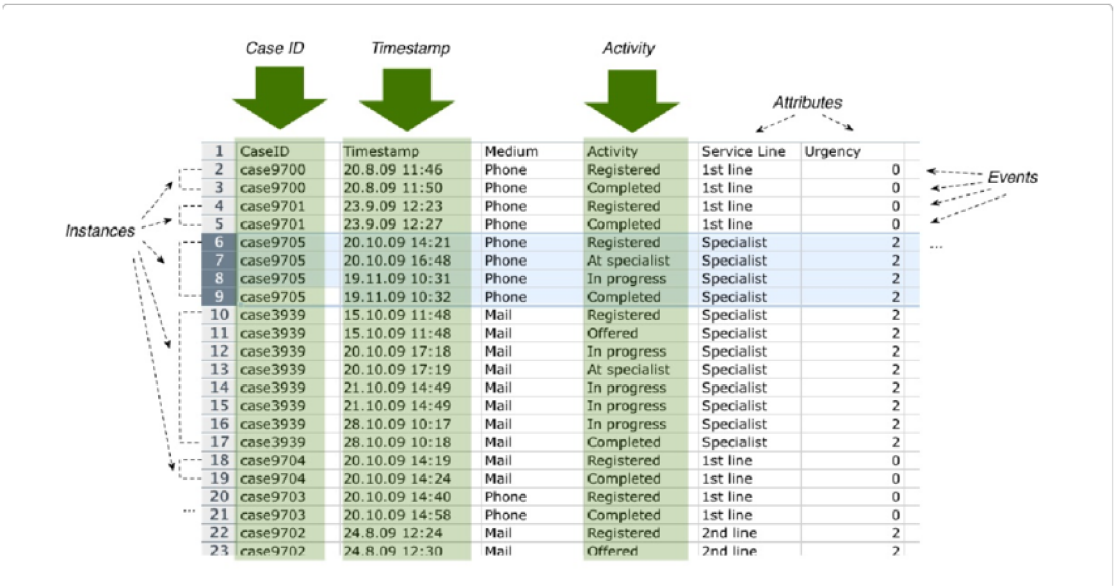
Process mining is a family of techniques that allow users to interactively analyze data extracted from enterprise information systems to derive insights to improve one or more business processes. Introduced by Wil van der Aalst[1] in the late 1990s, it has since evolved to become a key tool in understanding and improving business processes. At its core, process mining revolves around event logs, process models, and discovery algorithms. Event logs are sequences of events captured by IT systems; process models represent the expected flow of these events; and discovery algorithms are used to extract process-related information from the event logs. This synthesis provides unprecedented insights into the actual workings of business processes. Process mining tools extract business process execution data from an enterprise system and consolidate them into the form of an event log.

Process mining can be broadly categorized into three types:

- Discovery: Identify the actual processes by analyzing the event logs.
- Conformance: Check if the real-life processes conform to the predefined models.
- Enhancement: Enhancing existing process models based on insights gained from event logs.

An event log is a hierarchically structured file with historical information about the execution of a business process generated by a process-oriented information system. Cases organize attribute values within an event log. A case contains a group of events that belong to the same business process execution. An event is made up of properties or attributes. The typical attributes of an event are the name, timestamp, and resource of the activity. Depending on the granularity required in the event log and the type of process in which this event log will be used, other attributes can be included. In this direction, the sequence of events related to a case is known as a trace. Figure 2.1 illustrates an example of an event log. Event logs are typically used in the analysis of process-oriented systems and are central to the field of process mining. Each row in the log represents an event that has occurred within a system or process. In the following sections, Each characteristic will be thoroughly examined in the demonstration of the event log.

Some useful concepts for understanding the basis of event logs in the context of process mining are discussed next.



	CaseID	Timestamp	Medium	Activity	Service Line	Urgency
1	case9700	20.8.09 11:46	Phone	Registered	1st line	0
2	case9700	20.8.09 11:50	Phone	Completed	1st line	0
3	case9701	23.9.09 12:23	Phone	Registered	1st line	0
4	case9701	23.9.09 12:27	Phone	Completed	1st line	0
5	case9705	20.10.09 14:21	Phone	Registered	Specialist	2
6	case9705	20.10.09 16:48	Phone	At specialist	Specialist	2
7	case9705	19.11.09 10:31	Phone	In progress	Specialist	2
8	case9705	19.11.09 10:32	Phone	Completed	Specialist	2
9	case3939	15.10.09 11:48	Mail	Registered	Specialist	2
10	case3939	15.10.09 11:48	Mail	Offered	Specialist	2
11	case3939	20.10.09 17:18	Mail	In progress	Specialist	2
12	case3939	20.10.09 17:19	Mail	At specialist	Specialist	2
13	case3939	21.10.09 14:49	Mail	In progress	Specialist	2
14	case3939	21.10.09 14:49	Mail	In progress	Specialist	2
15	case3939	28.10.09 10:17	Mail	In progress	Specialist	2
16	case3939	28.10.09 10:18	Mail	Completed	Specialist	2
17	case9704	20.10.09 14:19	Mail	Registered	1st line	0
18	case9704	20.10.09 14:24	Mail	Completed	1st line	0
19	case9703	20.10.09 14:40	Phone	Registered	1st line	0
20	case9703	20.10.09 14:58	Phone	Completed	1st line	0
21	case9702	24.8.09 12:24	Mail	Registered	2nd line	2
22	case9702	24.8.09 12:30	Mail	Offered	2nd line	2

Figure 2.1: Sample Event log file

Definition 1. An event refers to a case, an activity, and a point in time. The event is characterized by a set of attributes such as ID, timestamp, cost, and resource, among others.

Definition 2. A trace can be seen as a case, i.e., a finite sequence of events z is the element of E*, such that each event appears only once.

Definition 3. An event log consists of a set of cases, and cases consist of events, such that each event appears, at most, once in the entire log.

The events for a case are represented in the form of a trace, i.e., a sequence of unique events. Moreover, cases, such as events, can have attributes. The structure of an event log is made up of the following elements.

- An event log consists of cases.
- A case consists of events such that each event relates to precisely one case.
- Events within a case are ordered.
- Events can have attributes. Examples of typical attributes are activity name, time, costs, and resources.

Definition 4. A Business process model is the graphical and analytic representation used to capture the behavior of an organization's business processes. A business process model is usually expressed through different graphic methods or notation languages, such as flowcharts, UML, workflows, Petri nets, and BPMN, among others.

2.2 Synthetic Data and Generation Techniques

Real-world data has a long history in artificial intelligence (AI). Collecting, processing, or distributing real-world datasets is often associated with data collection costs, quality problems, and privacy concerns. Synthetic data generation is the process of creating artificial data that mimic the statistical patterns and properties of real-life data. Synthetic data is generated using algorithms, models, or other techniques.

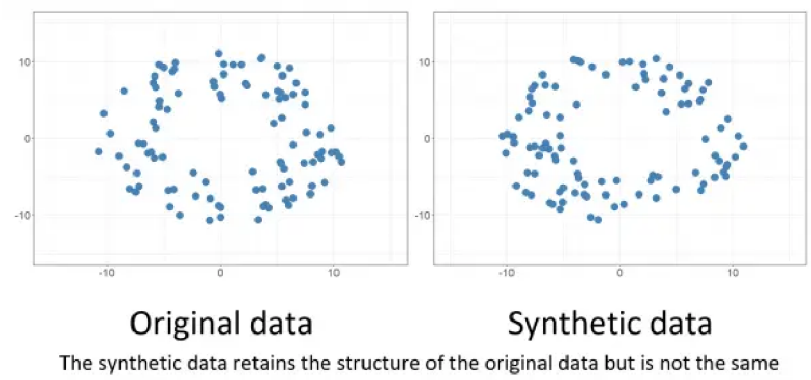


Figure 2.2: Synthetic Data vs Real Data

Synthetic data is randomly generated with the intent of hiding sensitive private information and retaining statistical information on features of the original data. Synthetic data are broadly classified into three categories:

- **Fully Synthetic Data**: Entirely artificial, this data type does not incorporate any element from the original dataset. It involves estimating the distribution functions of features in the original data and generating privacy-safe series based on these estimations. Techniques such as bootstrapping are commonly used. Its strength lies in robust privacy protection, though the data accuracy might be less reliable.

- **Partially Synthetic Data**: This approach involves substituting sensitive attributes in the original data set with synthetic values, particularly for data points at high risk of revealing private information. It is a method that seeks to balance data privacy with fidelity to the original dataset. Techniques like multiple imputation and model-based approaches are employed, which is also useful for addressing missing data issues.

- **Hybrid Synthetic Data**: Combining elements of both real and synthetic datasets, this method involves pairing each random real data record with a closely related synthetic one. The resultant hybrid data benefits from the advantages of both full and partial synthesis, offering improved privacy and utility, but at the cost of increased memory and processing requirements.

In a report on synthetic data, Gartner predicted by 2030 most of the data used in AI will be artificially generated by rules, statistical models, simulations, or other techniques. Figure 2.3 shows the trend for synthetic data to overshadow real data by 2030.

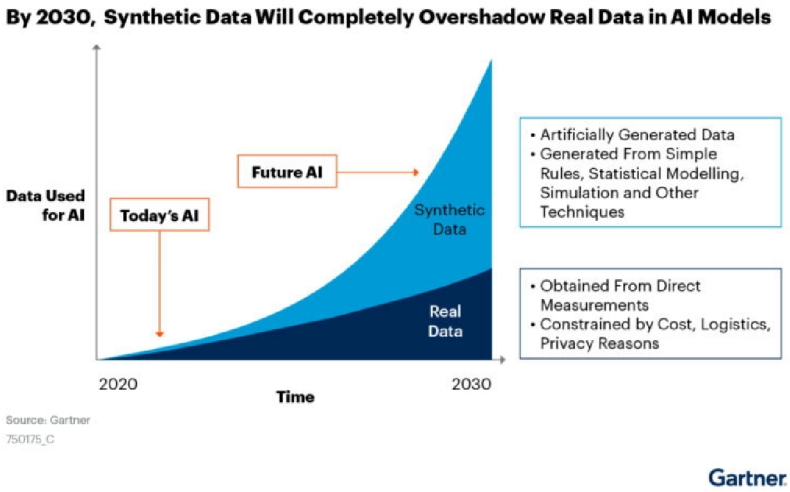


Figure 2.3: Synthetic data will become the main form of data used in AI. Source: Gartner, "Maverick Research: Forget About Your Real Data – Synthetic Data Is the Future of AI," Leinar Ramos, Jitendra Subramanyam, 24 June 2021[Image Source / NVIDIA]

The three most popular approaches to generate synthetic data are discussed below:

1. **Rule-Governed Synthetic Data Creation**: This technique uses explicit guidelines, constraints, and mathematical expressions to craft synthetic data that adhere to preestablished patterns. It is especially effective for data with known structures or links, such as the creation of synthetic time sequences or simulated sensor data.

2. **Synthetic Data Generation through Statistical Modeling**: This approach uses statistical models to produce artificial data. These models are trained on actual datasets to assimilate the inherent patterns and distributions, allowing the creation of new data instances that mirror the original data. Typical examples include the use of Gaussian mixture models, autoencoders, and generative adversarial networks (GANs).

3. **Combined Methodologies for Synthetic Data Generation**: This strategy merges rule-based and model-based techniques for synthetic data creation. By blending deterministic rules with statistical models, combined methods offer the versatility to produce varied and intricate data sets while upholding certain specific features and constraints.

In this work, GAN and LSTM networks will be used as primary methods for data synthesis. This specialized approach will allow us to explore the ability to generate realistic and complex data sets, tailored to enhance machine learning applications in process mining.

2.3 Synthetic Logs Generation Techniques

A diverse array of machine learning techniques have been employed to create artificial sequential event log data, increasing the availability of process data to train and evaluate process mining algorithms. These techniques encompass a range of approaches including models like recurrent neural networks (RNNs), the Transformer network, generative adversarial networks (GANs), and others.

2.3.1 Deep Learning Models

Deep learning encompasses a broad range of machine learning algorithms that employ artificial neural networks with multiple layers to learn intricate patterns from data. These models excel at tasks like classification, regression, and anomaly detection, where the goal is to map the input data to a corresponding output label or value. Deep learning has revolutionized various fields, including computer vision, natural language processing, and speech recognition.

Recurrent Neural Networks (RNNs)

Recurrent neural networks (RNNs) are a type of deep learning architecture specifically designed to handle sequential data. They excel at capturing long-range dependencies, meaning that they can effectively model the relationships between events that are far apart in an event log. This makes RNNs well-suited for generating realistic event sequences that reflect the intricate patterns of real-world processes.

LSTM (Long Short-Term Memory)

Long short-term memory (LSTM) is a type of recurrent neural network (RNN) architecture that is specifically designed to overcome the vanishing gradient problem, which is a common issue in RNNs that makes it difficult for them to learn long-range dependencies. LSTMs achieve this by introducing a memory cell that can maintain its state over time, allowing it to capture long-range dependencies between events in a sequence. Figure 2.4 shows the basic structure of the LSTM network.

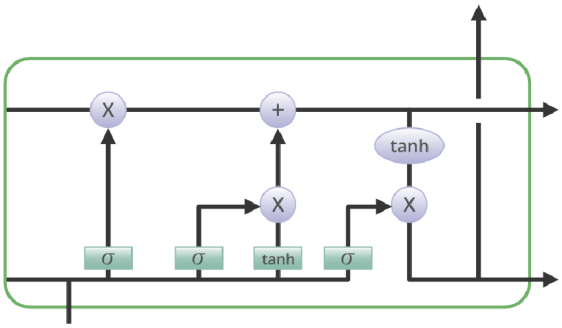


Figure 2.4: Structure of LSTM Network

Gated Recurrent Unit (GRU)

Gated recurrent unit (GRU) is another type of RNN architecture that is similar to LSTM but has a simpler structure. GRUs also introduce gating mechanisms to control the flow of information through the network but do not have a separate memory cell like LSTMs. This makes GRUs more computationally efficient than LSTMs, but they may not be as effective at capturing long-range dependencies. Figure 2.5 shows the simple architecture of the GRU network.

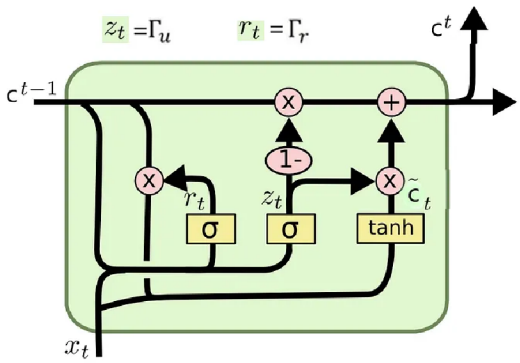


Figure 2.5: GRU Basic Architecture

Transformers

The Transformer, introduced in "Attention is All You Need" (Vaswani et al.)[2], has become a cornerstone in Natural Language Processing (NLP). Its encoder-decoder design leverages a self-attention mechanism, enabling parallel processing and better modeling of long-range dependencies within the language. The encoder creates continuous high-dimensional representations of input sequences for the decoder (see Figure 2.6). The decoder integrates these representations with previous outputs to generate an output sequence, producing state-of-the-art results for various NLP tasks.

2.3.2 Deep Generative Models

Deep generative models focus on generating new data instances that resemble the underlying distribution of the training data. They aim to capture the statistical structure of the data and produce samples that are indistinguishable from the real data points. This ability to generate realistic data makes deep-generative models particularly useful for tasks like data augmentation, synthetic data generation, and creative applications like art and music generation.

Variational Autoencoders

Variational autoencoder was proposed in 2013 by Diederik P. Kingma [3] and Max Welling at Google and Qualcomm. VAE provides a probabilistic way of describing an observation in latent space. Thus, rather than building an encoder that outputs a single value to describe each latent

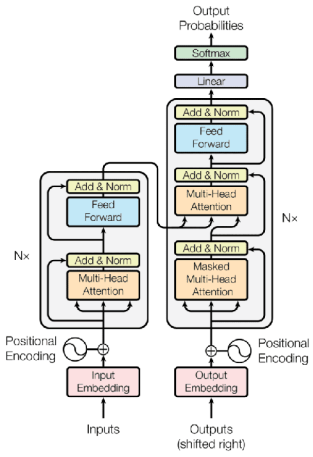


Figure 2.6: Encoder-decoder structure of the Transformer architecture taken from "Attention is all you need"

state attribute, the encoder is formulated to describe a probability distribution for each latent attribute. It has many applications, such as data compression, synthetic data creation, etc. It is different from an autoencoder in the sense that it provides a statistical way to describe the samples of the dataset in latent space. Therefore, the encoder outputs a probability distribution in the bottleneck layer instead of a single output value. However, VAEs also have limitations such as smooth data generation, limited data distribution representation, and mode collapse. Figure 2.7 shows the general architecture of VAE.

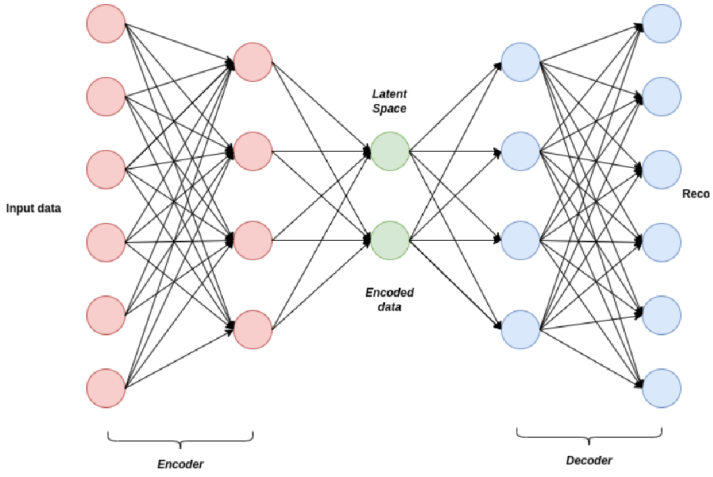


Figure 2.7: Variational Autoencoder

Generative Adversarial Network

GAN can generate new data instances based on a given training dataset. This is achieved through the interplay of two submodels: a generator and a discriminator. The generator creates data fabricated from random input, while the discriminator attempts to distinguish real data from its fabrications. Through a competitive process, these sub-models continuously improve their performance over time. The intricate architecture of GANs will be explored in the next section. Additionally, the focus of this thesis will be on the use of GANs to generate synthetic event log data.

2.4 Architectural Overview of GAN

Generative Adversarial Networks (GANs) were developed in 2014 by Ian Goodfellow [4] and his teammates. GANs have two main blocks that compete with each other and can capture, copy, and analyze variations in a dataset. The two models are usually called Generator and Discriminator(Figure 2.8). The generator network takes random input (typically noise) and generates samples, such as images, text, sequences, or audio, that resemble the training data it was trained on. The goal of the generator is to produce samples that are indistinguishable from the real data.

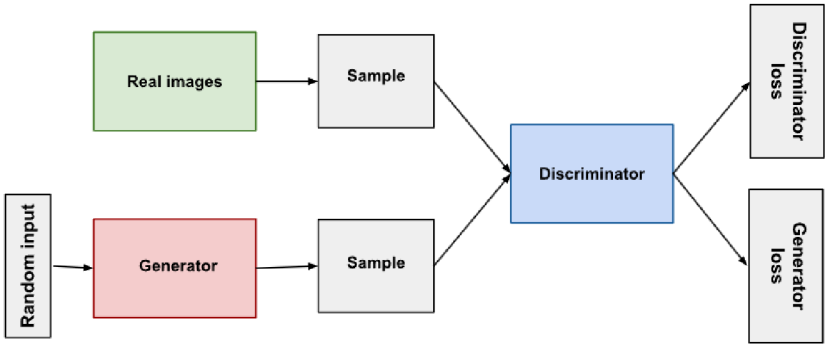


Figure 2.8: Generative Adversarial Network Architecture

As training progresses, the generator becomes more adept at producing realistic samples, while the discriminator becomes more skilled at differentiating between real and generated data. Ideally, this process converges to a point where the generator is capable of generating high-quality samples that are difficult for the discriminator to distinguish from real data.

The adversarial training implies that only one of the two networks will finally succeed.

- Generator: This will succeed if the discriminator, acting as a binary classifier, achieves a 50 percent accuracy in classifying real and synthetic data. This is the accuracy of a

random guess and means that the generator has learned a data distribution that generates data that are indistinguishable from the discriminator.

- Discriminator: This will succeed if it achieves 100 percent accuracy, being able to correctly classify between the real and the synthetic data generated by the generator.

Generator Model

A key element responsible for creating fresh, accurate data in a Generative Adversarial Network (GAN) is the generator model. The generator takes random noise as input and converts it into complex data samples, such as text or images. The underlying distribution of the training data is captured by layers of learnable parameters in its design through training. The generator adjusts its output to produce samples that closely mimic real data, as it is being trained by using backpropagation to fine-tune its parameters(Figure 2.9). The generator's ability to generate high-quality, varied samples that can fool the discriminator is what makes it successful.

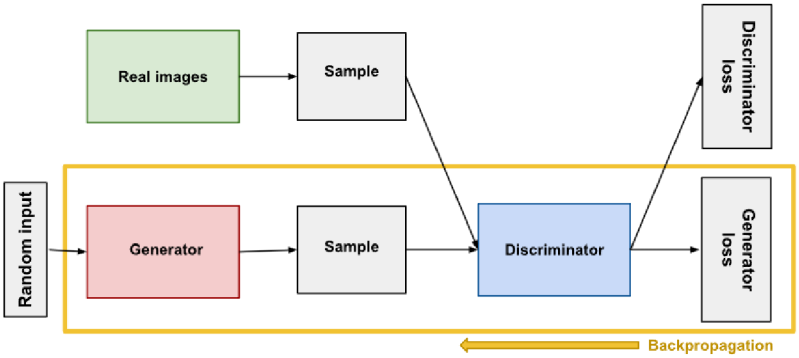


Figure 2.9: Backpropogation in Generator Training

Generator Loss(J_G)

For generated samples, the generator minimizes the log likelihood that the discriminator is right. Due to this loss, the generator is incentivized to generate samples that the discriminator is likely to classify as real (logD(G(z_i)) close to 1).

$$J_G = -\frac{1}{m}\Sigma_{i=1}^{m} log D(G(z_i))$$

where

- J_G measures how well the generator fools the discriminator.
- log D(G(z_i)) represents the log probability that the discriminator is correct for generated samples.
- The generator aims to minimize this loss, encouraging the production of samples that the discriminator classifies as real (log D(G(z_i)) close to 1.

Discriminator Model

An artificial neural network called a discriminator model is used in Generative Adversarial Networks (GAN) to differentiate between generated and actual input. By evaluating input samples and allocating the probability of authenticity, the discriminator functions as a binary classifier. Over time, the discriminator learns to differentiate between the genuine data of the dataset and the artificial samples created by the generator. This allows it to progressively hone its parameters and increase its level of proficiency. Maximizing the discriminator's ability to accurately identify generated samples as fraudulent and real samples as authentic is the objective of the adversarial training procedure. The discriminator updates its weights through backpropagation of the discriminator loss through the discriminator network(Figure 2.10). The discriminator becomes increasingly discriminating as a result of the interaction between the generator and the discriminator, helping the GAN to produce realistic synthetic data in general.

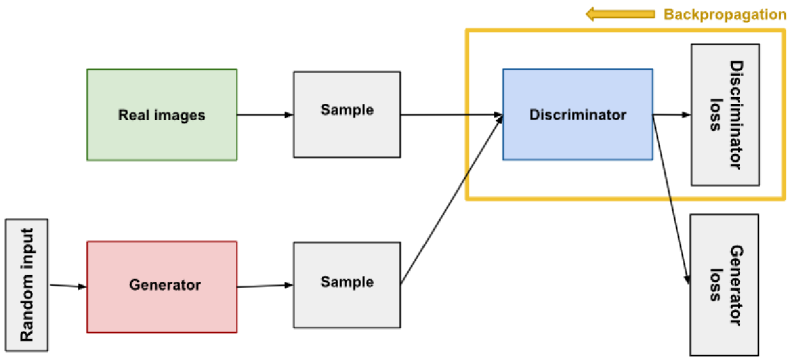


Figure 2.10: Backpropogation in Discriminator Training

Discriminator Loss(J_D)

The discriminator reduces the negative log-likelihood of correctly classifying both produced and real samples.

$$J_D = -\frac{1}{m}\sum_{i=1}^{m}\log\left(D\left(x_i\right)\right) - \frac{1}{m}\sum_{i=1}^{m}\log\left(1 - D\left(G\left(x_i\right)\right)\right)$$

MinMax Loss

In a Generative Adversarial Network (GAN), the minimax loss formula is provided by:

$$\min_G \max_D V(D,G) = Ex \sim p_{data}(x)[\log D(x)] + Ez \sim p_z(z)[\log(1 - D(G(z)))]$$

where

- G is the generator network and is D is the discriminator network

- Actual data samples obtained from true data distribution $p_{data}(x)$ are represented by x.
- Random noise sampled from previous distribution $pz(z)$ (usually a normal or uniform distribution) is represented by z.
- D (x) represents the probability that the discriminator will correctly identify the actual data as real.
- D(G(z)) is the likelihood that the discriminator will identify generated data from the generator as authentic.

2.5 ML Based Process Mining Tools

Gartner defines process mining tools as tools that are designed to discover, monitor, and improve processes by extracting knowledge from events captured in information systems to continuously deliver visibility and insights. Process mining includes automated process discovery (i.e., extracting process models from an event log), conformance checking (i.e., monitoring deviations by comparing model and log), social network/organizational mining, automated construction of simulation models, model extension, model repair, case prediction, and history-based recommendations. The magic quadrant for ML-based process mining tools from Gartner is shown in Figure 2.11. Several tools leverage ML to provide predictive insights, automation, and enhanced analytics. Here are some notable ML-based process mining tools:

- **Celonis** offers a comprehensive suite of machine learning-powered process and task mining capabilities, providing analytics, customization tools, and automation features to streamline business processes.
- **Disco** by Fluxicon is a user-friendly tool that provides detailed process analysis and visualization, making it accessible to business users.
- **EverFlow** utilizes cutting-edge technologies in Big Data and Machine Learning to analyze large volumes of events, offering simple and intuitive design for process insights.
- **LANA** Process Mining by Lana Labs includes an algorithm that enables prediction of future process behavior and automated compliance checks.
- **MEHRWERK** Process Mining (MPM) combines self-service process mining, visual analytics, and associative analytics on the Qlik Sense BI platform.
- **Minit** features advanced process improvement functionalities such as hierarchical visualization, simulation of ”what-if” scenarios, and interactive dashboards.
- **myInvenio** provides a comprehensive solution for process mining with functionalities such as simulation, decision rule mining, task mining, and analysis of multistage business processes.
- **PAFnow** by Process Analytics Factory is built on Microsoft Power BI, integrating process mining with business intelligence capabilities.

Figure 2.11: Magic Quadrant for ML based Process Mining Tools

- **ProDiscovery** from Puzzle Data offers a suite of widgets for process discovery, statistical analysis, and organizational charts, designed for Big Data processing.
- **QPR** ProcessAnalyzer offers advanced analytics to identify case clusters and root causes, customizable dashboards, and process prediction.
- **Signavio** Process Intelligence is part of Signavio's Business Transformation Suite, facilitating seamless integration between mining, modeling, and automation.
- **UiPath** The mining process part of an end-to-end automation platform combines RPA with AI and cloud technologies to enhance digital business operations.

The integration of synthetic data into these tools presents a forward-thinking approach, addressing challenges related to data scarcity, privacy constraints, and the need for comprehensive testing environments. As summarized by the notable tools discussed, synthetic data not only catalyze the refinement of process mining algorithms, but also fortifies the robustness and scalability of these systems. It represents a strategic asset, ensuring that the development of ML-based process mining tools continues to evolve, driven by efficiency, accuracy, and a deepened understanding of the underlying process dynamics.

3 Related Work

In March 2017, a study in the paper *"A Review Of Synthetic Data Generation Methods For Privacy-Preserving Data Publishing"* by *Surendra et. al* conducted a review of various methods for generating synthetic data. The findings of this paper suggest that a significant drawback of the majority of the methods analyzed is their dependency on extensive user involvement and expertise. Many of the synthetic data generation techniques described necessitate the establishment of a comprehensive set of rules and constraints before the generation process can begin. Furthermore, these methods demand that users possess a thorough knowledge of the data's domain.

In the process mining domain, collecting and sharing process data can be challenging due to its intricate nature and the presence of sensitive information [5] The hidden Markov model has been employed to expand process datasets by capturing the sequential relationships between activities and generating artificial process data [6]. In 2019, Zisgen et al.[7] in their study, implemented an advanced algorithmic approach to generate synthetic sensor event logs, improving the efficiency of the process extraction. This algorithm skillfully mimicked the complexity and variability of real-world operational data, creating authentic and varied event logs reflective of typical business process anomalies. The efficacy of their method, demonstrated through the nuanced and realistic nature of the produced logs, was pivotal in refining process mining algorithms. The generated logs provided a robust testing ground, ensuring the precision and reliability of the algorithm. The findings underscored the value of such sophisticated simulation techniques in augmenting process mining tools, particularly in their accuracy and adaptability to real-world data dynamics.

As shown in the work by Sommers et al. (2021)[8] GNNs have also been used for process mining tasks including process discovery and prediction. Although it does not directly address synthetic data generation for process mining, it is relevant due to its exploration of advanced process discovery techniques using GNNs. The paper provides insights into data preparation, process representation, and modeling, indirectly impacting the quality of data used in process mining. Although synthetic data generation is not its central theme, it contributes to the broader field by introducing advanced methods that can inspire improvements in synthetic data quality for process mining applications. Most of these approaches were not based on generative machine learning models, they were based on simpler algorithms for the development of synthetic data generators.

In recent years, deep generative models like RNN-based models, the Transformer network, and GANs have been used in the process mining domain for process event prediction[9][10][11]. These models can also be used to generate process data. There have also been some studies

focused on different types of synthetic data using Generative Adversarial Networks. To begin with, Esteban et al.[12] experimented with replacing the multilayer perceptron in the original GAN model with a Recurrent Neural Network (RNN) to generate real-valued time series medical data in a conditional setting. In the same line and motivated by privacy concerns, Choi et al.[13] proposed a new model called the Medical Generative Adversarial Network (medGAN) to generate realistic synthetic patient records, including discrete high-dimensional variables, through a combination of an auto-encoder and a GAN. There are some efforts in the direction of generating privacy-preserving process data, as discussed in paper by Keyi Li et al. [14] using traditional deep learning models. The paper by Yu et al.[15] claims that their Sequence Generative Adversarial Nets with Policy Gradient (SeqGAN) is the first work to extend GANs to generate discrete tokens of data. An approach that involves the use of reinforcement learning techniques in the generator.

In summary, the context of the studies presented in this section suggests that the idea of generating synthetic data using deep generative models for the data specific to the process mining domain requires more research and hence is the foundation of this thesis.

4 Methodology

This chapter covers a detailed overview of the event log data set, along with the exploratory data analysis and the preprocessing steps required, ensuring the suitability of the data for our study. Furthermore, the steps in the data generation process are defined that are utilized in this thesis. In the last section, the evaluation methods used for the synthetic data are discussed in detail.

4.1 Dataset Exploration and Pre-Processing

In this thesis, two datasets are used, namely BPI 2012 Loan applications and the helpdesk dataset.

4.1.1 Helpdesk Dataset

This data set captures activities related to the help desk ticketing process of an Italian software company. It has a total of 9 distinct activities, with each case beginning with the creation of a new ticket within the ticketing management system and concluding upon issue resolution and ticket closure. The data set comprises 3,804 process instances (or "cases") and 13,710 individual events. The data set used is already in the anonymized version, with activity names being replaced by their IDs [16]. Since the data is already preprocessed and encoded, the detailed explanation of the BPI2012 dataset is given in the next subsection to get more information about the Process mining data format and activities.

4.1.2 BPI Challenge 2012 Dataset

The BPI 2012 dataset represents a series of loan applications within a Dutch financial institute, starting with the submission by a customer for a loan and ending in the rejection, cancellation, or acceptance of the loan. Figure 4.1 shows the sample event log from the BPI 2012 dataset.

	org:resource	lifecycle:transition	concept:name	time:timestamp	case:REG_DATE	case:concept:name	case:AMOUNT_REQ
0	112	COMPLETE	A_SUBMITTED	2011-09-30 22:38:44.546000+00:00	2011-09-30 22:38:44.546000+00:00	173688	20000
1	112	COMPLETE	A_PARTLYSUBMITTED	2011-09-30 22:38:44.880000+00:00	2011-09-30 22:38:44.546000+00:00	173688	20000
2	112	COMPLETE	A_PREACCEPTED	2011-09-30 22:39:37.906000+00:00	2011-09-30 22:38:44.546000+00:00	173688	20000
3	112	SCHEDULE	W_Completeren aanvraag	2011-09-30 22:39:38.875000+00:00	2011-09-30 22:38:44.546000+00:00	173688	20000
4	NaN	START	W_Completeren aanvraag	2011-10-01 09:36:46.437000+00:00	2011-09-30 22:38:44.546000+00:00	173688	20000
5	10862	COMPLETE	A_ACCEPTED	2011-10-01 09:42:43.308000+00:00	2011-09-30 22:38:44.546000+00:00	173688	20000
6	10862	COMPLETE	O_SELECTED	2011-10-01 09:45:09.243000+00:00	2011-09-30 22:38:44.546000+00:00	173688	20000

Figure 4.1: Sample Event Log from the BPI 2012 Dataset

In the dataset, there are 13087 different cases of loan applications in the timeframe between October 2011 and March 2012. These 13087 cases lead to the entry of 262,200 different events, of which 164,505 signify the completion of an activity, while the other entries signify the scheduling of the start of the activities[17]. A typical event log gives an organized list of different case instances for a particular type of process. In the BPI 2012 dataset, all distinct case instances can be identified with the 'Case ID' identifier. Each row in the data set can then be interpreted as an individual event that takes place in any of these case instances. Each distinct event has an associated 'Activity Name' label such as A_SUBMITTED, A_DECLINED, etc. An event log will also store the timestamp at which a particular event occurs, which is denoted by the 'Complete Timestamp'. For our research, the Case ID and Activity attributes are the only necessary attributes among others, as our interest is primarily on the dependencies of the activities.

Key Columns in BPI 2012 Dataset

In our analysis of the BPI 2012 dataset, key statistical insights were derived for the columns **Case ID**, **Concept Name**, and **Timestamp** (Figure 4.2), offering a deeper understanding of the process flow:

CaseId	ActivityName	CompleteTimestamp
173688	A_SUBMITTED	2011-09-30 22:38:44.546000+00:00
173688	A_PARTLYSUBMITTED	2011-09-30 22:38:44.880000+00:00
173688	A_PREACCEPTED	2011-09-30 22:39:37.906000+00:00
173688	W_Completeren aanvraag	2011-09-30 22:39:38.875000+00:00
173691	A_SUBMITTED	2011-10-01 06:08:58.256000+00:00
173691	A_PARTLYSUBMITTED	2011-10-01 06:09:02.195000+00:00
173691	A_PREACCEPTED	2011-10-01 06:09:56.648000+00:00
173691	W_Completeren aanvraag	2011-10-01 06:09:59.578000+00:00
173694	A_SUBMITTED	2011-10-01 06:10:30.287000+00:00
173694	A_PARTLYSUBMITTED	2011-10-01 06:10:30.591000+00:00
173697	A_SUBMITTED	2011-10-01 06:11:08.866000+00:00
173697	A_PARTLYSUBMITTED	2011-10-01 06:11:09.035000+00:00
173694	A_PREACCEPTED	2011-10-01 06:11:13.026000+00:00
173694	W_Completeren aanvraag	2011-10-01 06:11:13.390000+00:00

Figure 4.2: Entries in the event logs with Case ID, Concept Name, and Timestamp columns

1. **Case ID**: Serves as a unique identifier for each process instance (e.g., a loan application). It is crucial for grouping events related to the same process, allowing an end-to-end view of each instance. The data set contains 13,087 unique case IDs.

2. **Concept Name**: Represents the specific activities or steps within the process. This column is key for identifying and analyzing the various stages and actions in the process flow. There are a total of 36 unique activities captured in the data set. The start activity is 'A_SUBMITTED', occurring 13087 times, while 'A_DECLINED' is the end activity with 3429 occurrences.

3. **Timestamp**: Indicates when each event occurred. Essential for time-based analysis, it helps in understanding the duration and sequencing of process steps and in identifying delays or bottlenecks.

Relevance to the Research

The BPI 2012 dataset is of significant relevance to the study on synthetic data generation for process mining, underscored by the following considerations:

1. **Depth and Diversity**: Provides a comprehensive log of real-world financial processes, offering a rich canvas to model synthetic data with a diverse range of process variations and event types.
2. **Benchmark Standard**: Its role as a reference within the process mining community ensures compatibility and comparability with a wide range of existing studies, enriching the continuity and context of research efforts.
3. **Data Quality**: The granularity and detail within the dataset present an opportunity to create high-fidelity synthetic data, which is essential for rigorous process mining analysis.
4. **Ethical Research**: Data anonymization addresses privacy concerns, meeting the ethical standards necessary for responsible research practices.
5. **Educational Resource**: Beyond research, the data set serves as an excellent learning tool for process extraction techniques, benefiting educational initiatives.
6. **Community Adoption**: Widespread use by the research community provides a supportive backdrop for problem-solving and innovation in process mining methodologies.

4.1.3 Data Exploration and Pre-processing

This section begins with Exploratory Data Analysis (EDA) and Preprocessing of the BPI Challenge 2012 dataset, laying the groundwork for accurate and insightful analysis. The generative models will be trained on these preprocessed data, aiming to create synthetic datasets.

Introduction to PM4Py Library

PM4Py is a Python library tailored for process mining, a field focusing on the analysis of business processes based on event logs. It encompasses a wide array of functionalities catering to various aspects of process mining.

1. Process Discovery: Creating process models from raw event logs (Figure 4.3). Using algorithms such as Alpha Miner, PM4Py can discover the underlying process model, revealing the sequence of activities and their relationships.

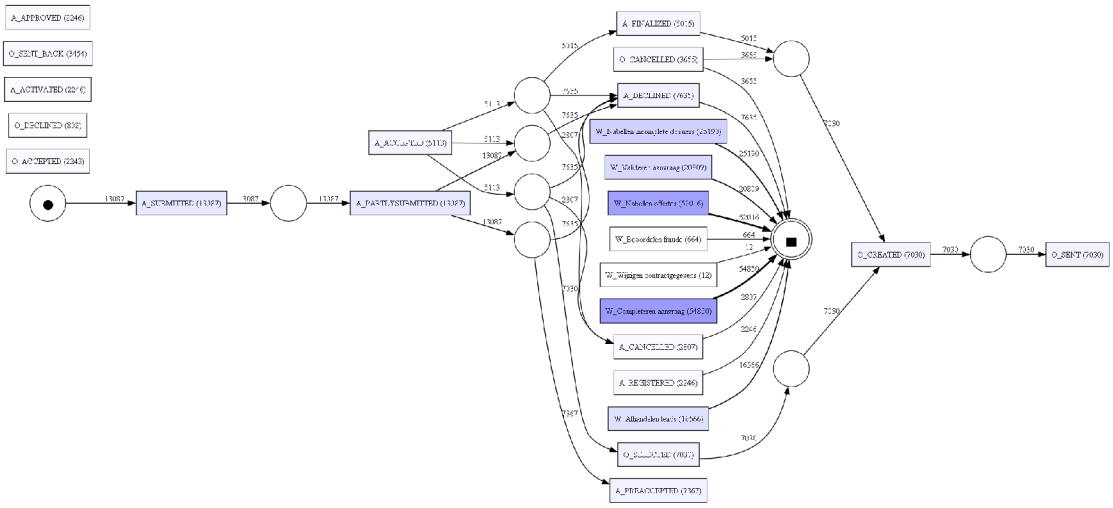


Figure 4.3: A Petri net from Alpha Miner showing business process event frequencies for BPI Dataset using PM4py library

2. Conformance Checking: Comparison of event logs with existing process models to identify deviations.

3. Model Enhancement: Enhancing process models using data derived from actual process execution.

4. Decision Point Analysis: Investigating decision points within processes.

5. Performance and Bottleneck Analysis: Identify and analyze process performance issues and bottlenecks.

In this thesis, PM4Py is particularly used to convert data from XES format, the standard in process mining, to CSV. This conversion is a crucial step in preparing the data for analysis using Python-based tools such as Pandas. For the evaluation of the synthetic data workflow, PM4py is again used to create process models.

The PM4Py library extends its functionality to the generation of process models like Petri nets, which are crucial for visualizing and analyzing business process flows. These models are pivotal in process discovery, a key aspect of process mining focused on deriving models from event log data. Algorithms such as Alpha Miner are instrumental in this regard, translating logs into a set of relationships that inform the creation of Petri nets. These nets serve as a valuable tool in both understanding potential business behaviors and in conformance checking—ensuring adherence to expected process flows, particularly when assessing synthetic data's alignment with real-world processes. Petri nets themselves are a blend of graphical and mathematical modeling, representing the concurrent elements of systems through places (conditions) and transitions (events), thereby providing a comprehensive view of a process's dynamics.

4.1.4 Exploratory Data Analysis

This exploratory data analysis begins with an investigation of the activity counts within the BPI 2012 dataset. The aim is to identify which activities are most prevalent and which are less common, as this information can highlight routine processes as well as pinpoint bottlenecks or rare, but critical, events. By visualizing the activity frequencies as shown in the accompanying bar chart, we can start to discern patterns and irregularities that may warrant a deeper dive to understand the underlying process behavior and efficiency.

Frequency of Activities in Event Logs

As shown in figure 4.4, few activities stand out "W_Completeren aanvraag", "W_Nabellen offertes" and "W_Nabellen incomplete dossiers" and have a lot of actions.The high frequency of activities beginning with "W_" in the BPI 2012 dataset bar chart likely indicates routine work tasks or steps central to the business process. They might represent repeated actions within cases, reflecting standard operational procedures. Alternatively, their prevalence could point to potential bottlenecks where tasks tend to accumulate. The activity occurrence is also used as an evaluation measure of the synthetic data in this research.

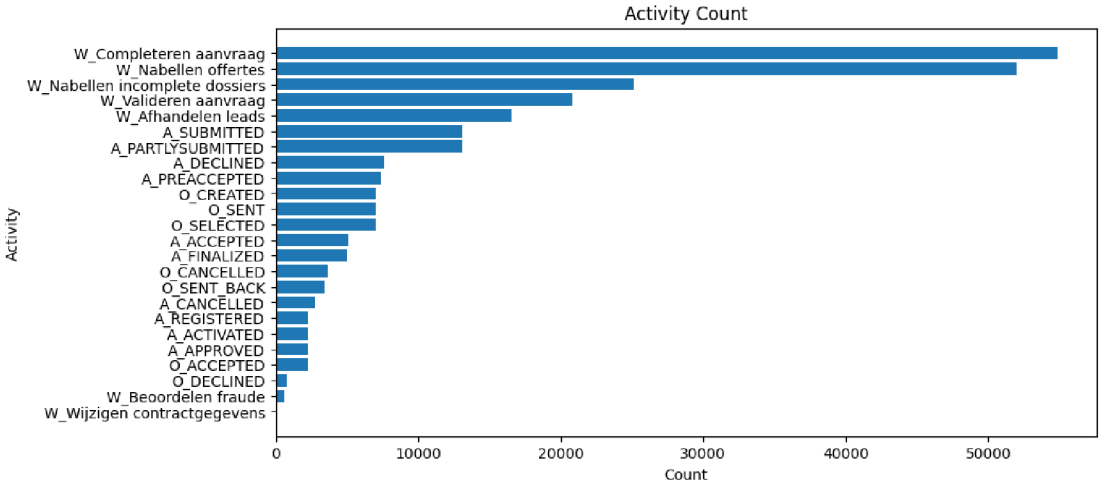


Figure 4.4: Frequency of Activities in Event Logs for BPI 2012 Dataset

Boundary Activities Analysis

An analysis of boundary activities is performed on process instances, as shown in Figure 4.5. The process log commences with a uniform initial activity across all instances, denoted as 'A_SUBMITTED'. This activity is the inception point for the process, occurring 13,087 times, which indicates that every case within the log begins with this step.
The consistent appearance of 'A_SUBMITTED' as the start activity suggests a standardized process entry across the dataset. The terminal activities within the event log are more varied, reflecting the multiple potential endpoints of a process instance. The dataset exhibits

```
Start Activities:  {'A_SUBMITTED': 13087}
End Activities:

{'W_Valideren aanvraag': 2747,
 'W_Wijzigen contractgegevens': 4,
 'A_DECLINED': 3429,
 'W_Completeren aanvraag': 1939,
 'A_CANCELLED': 655,
 'W_Nabellen incomplete dossiers': 452,
 'W_Afhandelen leads': 2234,
 'W_Nabellen offertes': 1290,
 'W_Beoordelen fraude': 57,
 'O_CANCELLED': 279,
 'A_REGISTERED': 1}
```

Figure 4.5: Boundary Activity Analysis

'A_DECLINED' as the most frequent concluding activity with 3,429 instances, followed by 'W_Valideren aanvraag' with 2,747 instances, and 'W_Completeren aanvraag' with 1,939 instances.

Analysis of Process Sequence Variants

Detailed information on the variants is shown in Table 4.1. Of 13087 cases in logs, 3429 of them (i.e. 26%) are in 1 variant. The variants 'A_SUBMITTED, A_PARTLYSUBMITTED, A_DECLINED' emerged as the most frequent, with a count of 3429. This sequence suggests a high occurrence of processes being submitted, partially submitted, and then declined. Other notable variants include 'A_SUBMITTED, A_PARTLYSUBMITTED, W_Afhandelen leads, W_Afhandelen leads, A_DECLINED, W_Afhandelen leads' with a count of 1872, and a more complex sequence 'A_SUBMITTED, A_PARTLYSUBMITTED, W_Afhandelen leads, W_Afhandelen leads, W_Afhandelen leads, W_Afhandelen leads, A_DECLINED, W_Afhandelen leads' recorded 271 times. These variants highlight a series of lead-handling activities interspersed with submissions and declinations. Sequences leading to 'A_PREACCEPTED' and 'A_CANCELLED' stages were also observed, though less frequently. Variants ending in 'A_CANCELLED' suggest processes that started but were not completed, a critical aspect for process efficiency analysis.

4.2 Data Generation Process using LSTM and GAN Models

The overall data generation process in the thesis by both models is designed to be executed in 5 steps (Figure 4.6). The input is the XES file, which is the start file for the event logs containing event details. The output is a collection of synthetic event sequences, where each sequence represents the possible order of events that a process can take.

Variant
A_SUBMITTED, A_PARTLYSUBMITTED, A_DECLINED
A_SUBMITTED, A_PARTLYSUBMITTED, W_Afhandelen leads, W_Afhandelen leads, A_DECLINED, W_Afhandelen leads
A_SUBMITTED, A_PARTLYSUBMITTED, W_Afhandelen leads, W_Afhandelen leads, W_Afhandelen leads, W_Afhandelen leads, A_DECLINED, W_Afhandelen leads
A_SUBMITTED, A_PARTLYSUBMITTED, W_Afhandelen leads, W_Afhandelen leads, A_PREACCEPTED, W_Completeren aanvraag, W_Afhandelen leads, W_Completeren aanvraag, A_DECLINED, W_Completeren aanvraag
A_SUBMITTED, A_PARTLYSUBMITTED, A_PREACCEPTED, W_Completeren aanvraag, W_Completeren aanvraag, A_DECLINED, W_Completeren aanvraag
A_SUBMITTED, A_PARTLYSUBMITTED, A_PREACCEPTED, W_Completeren aanvraag, W_Completeren aanvraag, A_CANCELLED, W_Completeren aanvraag
A_SUBMITTED, A_PARTLYSUBMITTED, W_Afhandelen leads, W_Afhandelen leads, A_PREACCEPTED, W_Completeren aanvraag, W_Afhandelen leads, W_Completeren aanvraag, W_Completeren aanvraag, W_Completeren aanvraag, A_DECLINED, W_Completeren aanvraag
A_SUBMITTED, A_PARTLYSUBMITTED, A_PREACCEPTED, W_Completeren aanvraag, W_Completeren aanvraag, W_Completeren aanvraag, W_Completeren aanvraag, A_DECLINED, W_Completeren aanvraag
A_SUBMITTED, A_PARTLYSUBMITTED, A_PREACCEPTED, W_Completeren aanvraag, W_Completeren aanvraag, W_Completeren aanvraag, W_Completeren aanvraag, A_CANCELLED, W_Completeren aanvraag
A_SUBMITTED, A_PARTLYSUBMITTED, W_Afhandelen leads, W_Afhandelen leads, A_PREACCEPTED, W_Completeren aanvraag, W_Afhandelen leads, W_Completeren aanvraag, W_Completeren aanvraag, W_Completeren aanvraag, W_Completeren aanvraag, W_Completeren aanvraag, A_DECLINED, W_Completeren aanvraag

Table 4.1: Most frequent sequence Variants in BPI 2012 Dataset

4.2.1 Data Acquisition and Conversion

The first stage involves sourcing data from the renowned Business Process Intelligence Challenge (BPI) 2012 dataset. This data set, originally in the XES (eXtensible Event Stream) format, which is a standard format for process mining, is converted to the CSV (Comma Separated Values) format for better analysis.

4.2.2 Pattern Analysis and Feature Engineering

The main idea behind this step is to detect the correlations in the data set and encode and process the data set so that it can be understood and best processed by any machine learning or statistical model.

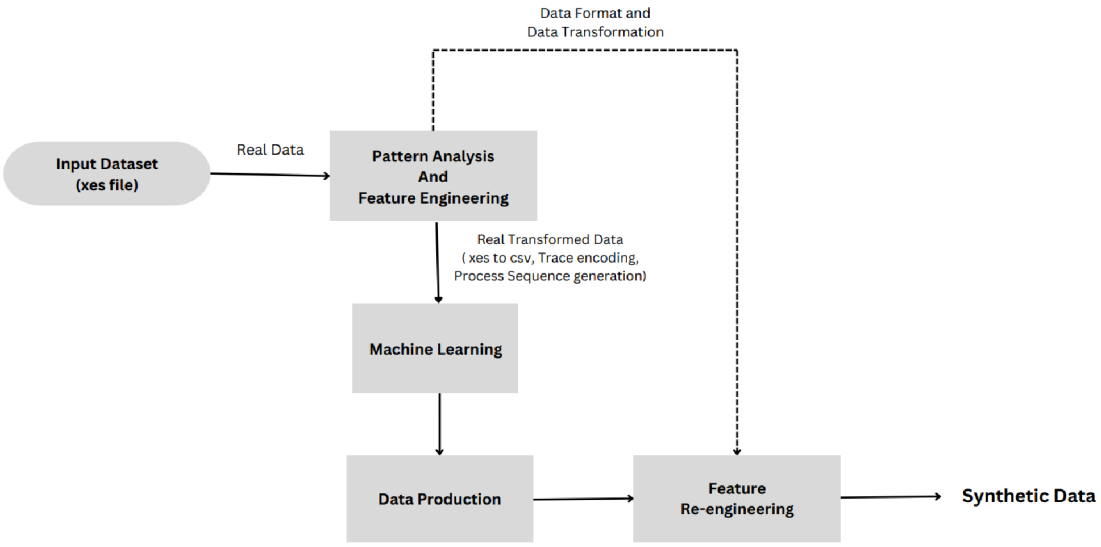


Figure 4.6: Data Generation Process

Chronological Sequence Generation

Following conversion, the next critical step is to generate sequences for each case ID using the timestamp column. This process involved organizing the events of each case in a chronologically coherent sequence that reflected the actual flow of activities within the business process. By aligning these events according to their timestamps, the temporal integrity of the process flow is maintained, which is a vital aspect of accurate process mining.

Threshold Selection for Sequences

To facilitate the generation of synthetic data that accurately reflect the complexity and variability of sequences of real-world processes, it is imperative to establish meaningful thresholds. These thresholds serve as benchmarks for categorizing the sequences according to their length, which is a proxy for their complexity. The histogram of the sequence lengths in Figure 4.7 provides the empirical basis for determining these thresholds. Upon analysis of the histogram:

- **Initial Observations**: The data are heavily skewed towards shorter sequence lengths, with a pronounced peak at the lower end of the sequence-length spectrum. This indicates a predominance of simpler process sequences within the data set.
- **Tail Analysis**: A noticeable long tail extends towards the higher sequence lengths, suggesting the presence of more complex and less frequent process variants.

Using these observations, the threshold values are determined. For the experiments, the threshold of 25 is taken. Situated slightly above the median sequence length of 11, it captures the most frequently occurring sequence lengths. This threshold ensures that the synthetic data represent the bulk of the process instances. In applying these thresholds, the aim is to ensure that

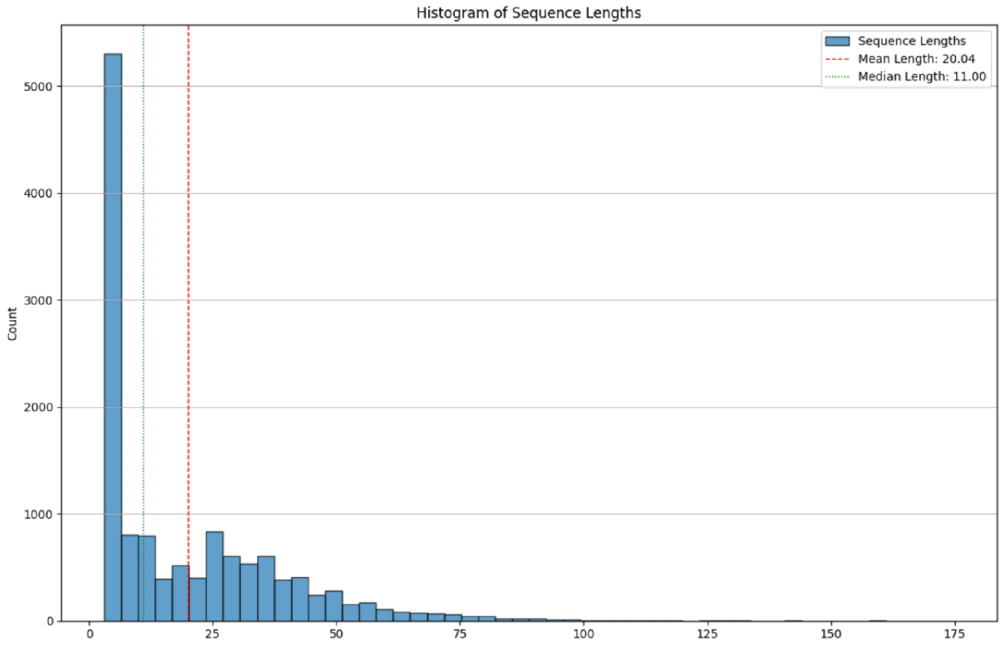


Figure 4.7: Histogram of Sequence length for Bench-marking for BPI 2012 Dataset

the generative models are exposed to a wide range of process sequences during training. This approach is designed to produce a synthetic data set that is not just a replication of the most frequent sequences, but is inclusive of the diversity inherent in real-world process flows.

Index-based Encoding for Machine Learning Readiness

With machine learning as the cornerstone of synthetic data generation, preparing the data set for algorithm compatibility is essential. The activity names are translated into integers, and it is more than just a data transformation technique. For instance, the event sequence ['A_SUBMITTED', 'A_PARTLYSUBMITTED', 'W_Afhandelen leads', 'W_Afhandelen leads', ' A_DECLINED', 'W_Afhandelen leads'] is converted to [10, 7, 18, 18, 5]. It is a critical step in rendering the data suitable for machine learning models, particularly those that are sensitive to non-numerical data. This encoding not only makes the dataset machine learning ready, but can also be used as an anonymization method to hide private information. It also significantly improves the efficiency of the computations, a crucial factor when working with complex generative models.

4.2.3 Machine Learning - Model training for Synthetic data generation

The Machine learning part consists of the model training step where the model is created and trained with either a sample of real data or random noise based on its architecture. In this thesis, as discussed above, the LSTM and GAN model with Transformer encoder-based generator and discriminator are compared for the process sequence generator parameters.

LSTM as generative Model

LSTMs are explicitly designed to overcome the limitations of traditional RNNs in learning long-term dependencies, making them exceptionally good at capturing complex patterns over lengthy sequences. They can generate a wide variety of sequence types, including text, time series data, and event logs, making them versatile tools for synthetic data generation in different domains.

- **Learning Phase**: Initially, the LSTM model is trained on a dataset that contains real sequences. During this phase, the model learns the underlying patterns, structures, and dependencies within the data. This includes learning the probability distribution of the sequence elements and their temporal dependencies.
- **Generation Phase:** Once the model is adequately trained, it generates new sequences. Generation typically starts with a seed (an initial sequence or part of a sequence), and the model predicts the subsequent elements one at a time based on the learned distribution. The output of each step can be fed back into the model as input to generate the next step, continuing until a sequence of the desired length is produced.

Transformer Encoder based GAN Architecture

- **Architecture Setup**: In this setup, both the generator and discriminator are built upon Transformer architectures. The Transformer's encoder component is utilized in both models to handle sequential input data effectively. The generator learns to produce sequences that resemble the training data, while the discriminator learns to distinguish between real data from the training set and fake data produced by the generator.
- **Training Phase**: During training, the generator receives a random noise vector and transforms it into a sequence. The discriminator then evaluates sequences from both the generator and the real dataset, learning to classify them as real or fake. This adversarial process continues iteratively, with the generator improving its ability to produce realistic sequences and the discriminator enhancing its ability to detect synthetic ones. The use of Transformer encoders allows both models to better handle the complexities of sequence data, leveraging attention mechanisms to capture long-range dependencies within the data.

4.2.4 Data Production and Feature Re-engineering

Once the models are trained and in a productive state, it is possible to generate new synthetic sequences for the defined data set. The input for generation of sequences is given based on the model type, and depending on the number of sequences to be generated, the data are generated by models in the index encoded sequences format. All the transformations performed in the step feature engineering can be reversed here to have the same format as the input data to make it usable for analysis and evaluation purposes such as creating workflow diagrams, and more.

4.3 Evaluation Methods

4.3.1 Sequence Length Analysis

To determine whether the length variation found in the real process sequences is accurately reflected by the synthetic sequences. This evaluation involves comparing the lengths of the sequences generated by both models based on the comparison of the sequence length probability distribution with a real dataset. This assessment is crucial for understanding whether the generated sequences follow the sequence-length distribution present in the real data.

4.3.2 Activity type Occurrence Distribution

For each unique activity type, the frequency of occurrence within the sequences of the synthetic and real data sets is calculated. The goal is to measure how closely the distribution of activity types in the synthetic dataset (S_a) matches the distribution in the real dataset (X_a).

4.3.3 Sequence Variance using Sum of Pairwise Normalized Edit Distances

Sequence variance refers to the diversity or spread of process sequences in a data set. The sum of Pairwise Normalized Edit Distance (SPE) is a metric used to measure the heterogeneity or diversity of a set of sequences. It provides a way to quantify this variance by looking at how different each sequence is from every other sequence in the data set. The sum of Pairwise Normalized Edit Distance can be calculated using the following steps:

To quantify the internal diversity of the authentic and generated data set, the sum of pairwise normalized edit distances (SPE) is used. This statistical metric is critical for assessing the variance within a collection of sequences and providing information on the heterogeneity of the data set. The SPE is particularly adept at encapsulating the spread of the sequences, which is essential for ensuring that the synthetic data mirrors the complexity and variability inherent in real-world data. The SPE is computed by following a two-step process.

1. **Edit Distance (ED)**: The ED is employed to measure how dissimilar two sequences are by counting the minimum number of operations needed to transform one sequence into the other. Operations considered include insertions, deletions, and substitutions. The edit distance in question used here is the Levenshtein distance, which accounts for these types of operation.

2. **Normalization**: To normalize the impact of the sequence length on our variance measure, divide the edit distance by the combined length of the two sequences in comparison. This normalization accounts for the possibility that longer sequences naturally have a larger edit distance, not necessarily indicative of greater diversity.

The SPE is then calculated using the formula:

$$SPE = \frac{1}{N^2} \sum_{i=1}^{N} \sum_{j=i+1}^{N} \frac{ED(s_i, s_j)}{(length(s_i) + length(s_j))} \tag{4.1}$$

where:

- N is the total number of sequences in the dataset.
- s_i and s_j are individual sequences of the data set.
- $ED(s_i, s_j)$ is the edit distance (Levenshtein distance) between the sequences s_i and s_j.
- $length(s_i)$ and $length(s_j)$ are the lengths of the sequences s_i and s_j, respectively.

The SPE gives an overall indication of diversity within the dataset. A lower SPE value would suggest that the sequences are quite similar to each other, indicating low variance. A higher SPE value would indicate that there is a wide range of differences between the sequences, suggesting a high variance. High variance is often desirable in synthetic datasets because it implies that the synthetic data encompasses a wide range of possible scenarios, which can be essential for robust machine learning.

4.3.4 KL Divergence

KL Divergence, or Kullback-Leibler Divergence, is a measure of information theory that quantifies how much one probability distribution diverges from a second expected probability distribution. It is often used in various fields, including statistics, data science, and machine learning, to measure the difference or similarity between two distributions. When comparing real (observed) and synthetic (model-generated) sequences, KL Divergence can help assess how well the synthetic data represents the real data.

$$D_{KL}(P \parallel Q) = \sum_{x \in X} P(x) \log \left(\frac{Q(x)}{P(x)} \right) \tag{4.2}$$

where:

- P is the probability distribution derived from the real data.
- Q is the probability distribution derived from the synthetic data.
- X is the set of all events (or unique elements in your sequences).

The result of the KL Divergence is a non-negative value where a result of 0 indicates that the two distributions are identical (in the context of the information contained in the distributions). Higher values indicate a greater divergence.

4.3.5 Unique Sequence Comparisons

To assess how well the synthetic data (generated by the model) mirrors the real-world data, the comparison of unique sequences is crucial. When unique sequences are compared, the diversity and authenticity of the generated data can be evaluated, thereby validating and refining the model performance. It is also important to ensure that synthetic sequences are not mere replicas of real sequences, but are, instead, diverse and representative. Analysis of unique sequences can lead to the identification of novel patterns, structures, or functions that were not evident in the real data alone.

4.3.6 Process Flow Evaluation

The evaluation of synthetic data for process mining must extend beyond statistical similarity and encompass the functional similarity of the process flows. It is imperative to confirm that the synthetic data not only statistically resembles the real data but also preserves the underlying process flows and structures. The PM4Py library, a state-of-the-art process mining toolkit implemented in Python, is used to plot the process models using the Heuristic Miner process discovery algorithm.

The direct comparison of process models from real and synthetic data offers insightful perspectives on the utility of the generated data. A high degree of similarity in visualization and low alignment costs in conformance checking reinforce the validity of synthetic data for use in process mining applications. On the contrary, significant deviations suggest areas where the data generation model may require further tuning to more accurately capture the nuances of real process flows.

5 Implementation

In this chapter, the implementation details related to the research, focusing on two methods: Long Short-Term Memory (LSTM) networks and then a more advanced method using GAN is discussed. The analysis of the results, with a comparison between the performance of GAN and LSTM models, is discussed in the next chapter.

5.1 LSTM Synthetic Sequence Generator

This approach is used for comparison purposes, in which the traditional LSTM model is used for the event log sequence generation. This choice is motivated by the LSTM's ability to learn and retain information across longer periods, effectively modeling the temporal nature of process event logs. For the list of parameters and variables involved in the definition and training of the model, please refer to Table 5.1. For more information on LSTMs, please refer to the original paper[18].

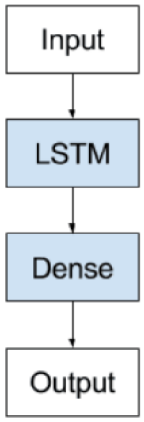


Figure 5.1: LSTM as Generative Model

Event sequences are represented as integer indices, which are first transformed into continuous vector embeddings. These embeddings feed into the LSTM layer, where hidden states capture the context and dependencies between events. At each step, the model predicts the probability of the next event of each sequence based on the current hidden state and the learned patterns.

Training involves feeding actual process event logs into the model. Over time, the model learns

typical event sequences and transitions, updating its internal parameters to minimize prediction errors. This process attempts to replicate the statistical properties of the training data, resulting in synthetic logs that resemble the real ones but avoid direct copying. The early stopping mechanism is also used to terminate training if the validation loss does not show improvement over a certain number of epochs. The model was meticulously configured with a set of hyperparameters optimized for sequence generation tasks.

The model's architecture is designed with an embedding dimension of 4, catering to the compact vocabulary size of 23 unique tokens for the BPI 2012 dataset, and a hidden layer dimension of 16, which provides a balance between capturing the nuances of the input data and maintaining computational efficiency. The training was carried out with each sequence standardized to a maximum length based on the length of the sequence tokens to ensure uniform processing. The training process utilized a batch size of 64 over 200 epochs, employing a learning rate of 0.01 to guide the optimization process toward effective model convergence. Execution on a CUDA-enabled GPU for accelerated computation was strategically chosen to foster robust learning and generalization capabilities of the LSTM generator, thereby enhancing its performance on the designated sequence generation tasks.

Variable	**Value**
Neural network architecture	LSTM
Hidden layers	1 fully connected
Activation function	LogSoftmax
Number of neurons	16
Optimizer	Adam optimizer
Batch size	64
Number of training epochs	200
Learning rate	0.01
Device for computation	CPU/GPU

Table 5.1: Variables and parameters related to the definition and training of LSTM Model. Any parameter not listed in this table was left as default.

5.2 GAN-based synthetic data generator

This thesis primarily employs Generative Adversarial Networks (GANs) as its foundation. The implementation deviates from the traditional GAN architecture, as the data generation is accomplished through the integration of a transformer's encoder and the architecture of a GAN. In contrast to LSTM which generates sequences incrementally from a single initial token, transformers can process an entire random sequence as input all at once. The Transformer network uses the positional encoding technique to inform the relative positions of the tokens. The transformer leverages its multihead self-attention mechanism to effectively capture long-range dependencies within data. Concurrently, the GAN architecture improves the accuracy of the synthesized sequences produced, particularly in scenarios where the available data is limited.

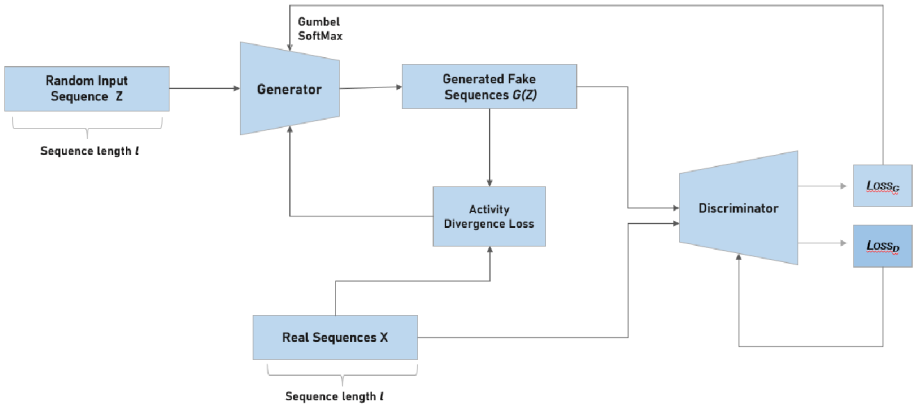


Figure 5.2: Architecture diagram for GAN for generating process data. The random input sequences and actual sequences undergo preprocessing to ensure they are of equal length. The discriminator is taught to differentiate between the real and generated sequences. The generator is fine-tuned using losses calculated by the discriminator and activity divergence.

Model Definition

The first step to train Generative Adversarial Networks is to define the generator and discriminator models. The way these models are designed for the implementation of this thesis slightly differs in various aspects from the way they are commonly defined. The main reason is that most GAN implementations are designed to generate images, while the focus here is the sequence event generation; second, an auxiliary loss function is used for the activity distribution divergence to better train the GAN model. The implementation of Transformer Encoder and GAN is based on Pytorch learning documents [19][20]. Figure 5.2 describes the basic architecture diagram for the Process Data generation using GAN.

The generator initializes by taking a random sequence as input. It then embeds this sequence into a continuous vector representation inspired by word embedding techniques commonly used in Natural Language Processing (NLP). Subsequently, the embedded sequence is fed into the self-attention mechanism of the Transformer architecture, enabling the model to analyze the positional and relational dependencies within the sequence. Finally, the output of the transformer generates a categorical vector, representing the probability distribution over activity classes. An argmax operation is applied to this vector, resulting in a one-hot encoded representation of the generated sequence.

To prevent mode collapse potentially arising from learning rate discrepancies between the generator and discriminator during adversarial training, the Transformer encoder-based discriminator is utilized. By identifying the first end token in generated sequences and padding subsequent tokens, the discriminator effectively processes sequences of one-hot encoded vectors

(both generated and authentic) and outputs a binary classification indicating their authenticity. For the list of parameters and variables involved in the definition and training of the model, please refer to Table 5.2.

Variable	Value
Dropout rate	0.1
Batch size	128
Generator/Discriminator ratio	2 (k value)
Learning rate (generator)	0.0001
Learning rate (discriminator)	0.0001
Total epochs	800
Optimizer	Adam Optimizer
Loss Function	Binary Cross-Entropy Loss
Device for computation	CPU/GPU

Table 5.2: Parameters for the GAN model

Learning Objectives

In a dynamic interplay, the generator and discriminator were iteratively optimized to establish an equilibrium. The discriminator evaluated the generated sequences, providing feedback to the generator. The generator was designed to enhance its performance by maximizing this evaluation score. The generator's last layer outputs sequences of vectors v, where each dimension of v represents the probability of a specific token being generated in that position. During forward propagation, argmax is applied on v to create a sequence in one-hot encoding. This sequence is then input to the discriminator. However, since the argmax operation is not differentiable, it cannot pass gradients back during backpropagation. To address this, the model uses the straight-through Gumbel-Softmax mechanism, a differentiable sampling operation that allows the discriminator to score the generated sequences and pass the gradient back to the generator. Compared to RNNs, random input and adversarial training in GAN help reduce exposure bias. The activity distribution divergence (MSE) is added as an auxiliary loss to the generator. The divergence in each training batch is fed to the generator.

Adversarial Training

During the initial stages of adversarial training, the discriminator might quickly reach convergence if the generator produces sequences that are not plausible. As a result, could potentially reject all sequences generated by Generator, leading to ineffective training. To manage the speed of optimization and prevent rapid convergence, a training strategy is employed. This strategy involves optimizing the generator for a certain number of epochs, specifically two epochs in this study, followed by a single epoch of optimization for the discriminator. The generator and discriminator iteratively strive to reach an equilibrium. The discriminator acts

like a critic and evaluates the process sequences generated by the generator along with assigning a scalar score. The generator aims to maximize this score to produce sequences that are indistinguishable from real data.

Gumbel Softmax Mechanism

Gumbel-Softmax is a technique used in generative models to approximate discrete distributions with continuous ones. This allows the gradients of the model to be backpropagated through the discrete sampling process, which is essential for training the model using stochastic gradient descent (SGD).In this model, the Gumbel-Softmax distribution is used to approximate the discrete distribution over activities that are used to generate process sequences. This allows the model to be trained using SGD and produce more realistic and diverse process sequences. This technique is used to handle the discrete backpropagation from the discriminator.[21]

Loss Functions

In a Generative Adversarial Network, the error between the output of the discriminator and the real labels is determined using the binary cross-entropy loss. This loss ensures that the discriminator provides a clear signal to the generator about the quality of the synthetic data, hence guiding the generator to produce more realistic data. Before the actual adversarial training begins, to improve the generator's performance in the GAN architecture, an activity loss is introduced during a pre-training phase. This loss assesses the discrepancy between the frequency distributions of generated and real sequence tokens. The evaluation of the results is based on the evaluation methods discussed in the methodology section.

5.3 Experimental Setup and Hardware Requirements

The experimental framework for this research was primarily supported by a robust cloud computing infrastructure. The bulk of the experiments, including data preprocessing and visualization tasks, were executed on a Google Colab Pro cloud platform. This environment was chosen due to its balance of computational power and accessibility, which is detailed in Table 5.3.

Specification	Value
Cloud Platform	Google Colab Pro
Processor	2.00 Cores, 1 vCPU
Memory Size	16 GB
GPU	NVIDIA Tesla P100
Languages	Python 3.11.0

Table 5.3: Specifications of the test system used to train LSTM and GAN models and run all the experiments, including data preprocessing and visualization.

The software implemented is written in Python 3.11.0 with the Python libraries mentioned in

table 5.4.

Package Name	Version	Purpose
PyTorch	1.8.1	ML framework and NN library
Pandas	1.1.5	Data manipulation and analysis
NumPy	1.19.5	Numerical computations
Matplotlib	3.3.4	Data visualization
Python	3.11.0	Programming language

Table 5.4: Software packages and libraries used for the experimental setup.

6 Experiments

In the thesis, a series of experiments have been conducted to compare the synthetic data generated using Long Short-Term Memory (LSTM) networks and Generative Adversarial Networks (GANs). This chapter reveals the results of the experiments. To allow a clear comparison, Figure 6.1 presents a statistical analysis of the real data used in the LSTM and GAN models for the Helpdesk and BPI 2012 datasets. This analysis includes key metrics like the number of cases, vocabulary size, and mean and standard deviation of sequence lengths. In this section, there

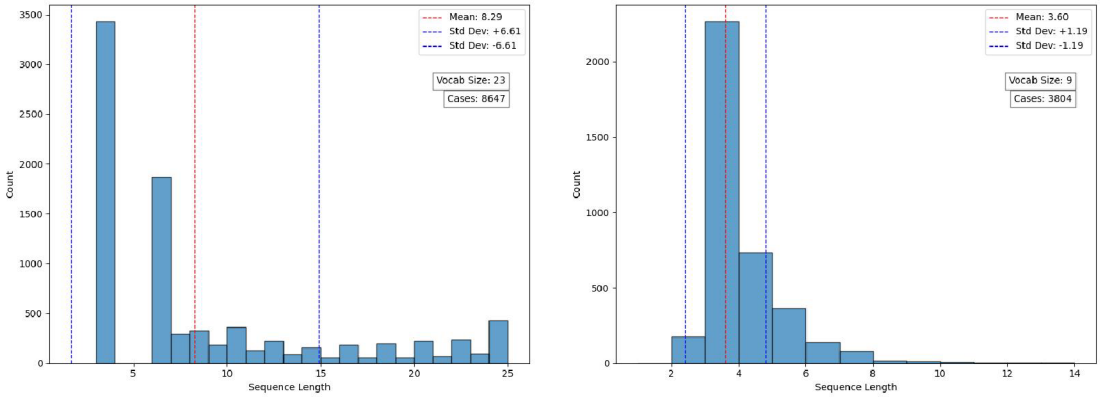


Figure 6.1: Summary of the characteristics of the BPI 2012 (left) and Helpdesk (right) dataset

will be a discussion on the comprehensive analysis of the synthetic data generated based on the metrics discussed in the methodology section. LSTM and GAN models will be compared and analyzed for the BPI 2012 and helpdesk datasets. For the evaluation of each model, 500 synthetic sequences were generated and compared for both data sets.

6.1 Model Training Results

To understand the performance of the GAN model, the plot for the discriminator loss is shown in Figure 6.2. This plot illustrates the trend of discriminator loss during the training of the GAN model aimed at generating synthetic process activity sequences.
Initially, a spike in the discriminator loss is observed, suggesting a period of uncertainty or less effective discrimination between real and synthetic sequences. As training proceeds, a downward trend in the discriminator loss is noticeable, implying an improvement in the discriminator's performance. This is indicative of the adversarial training process, where the discriminator and the generator iteratively improve in response to each other's progress. A decline in the

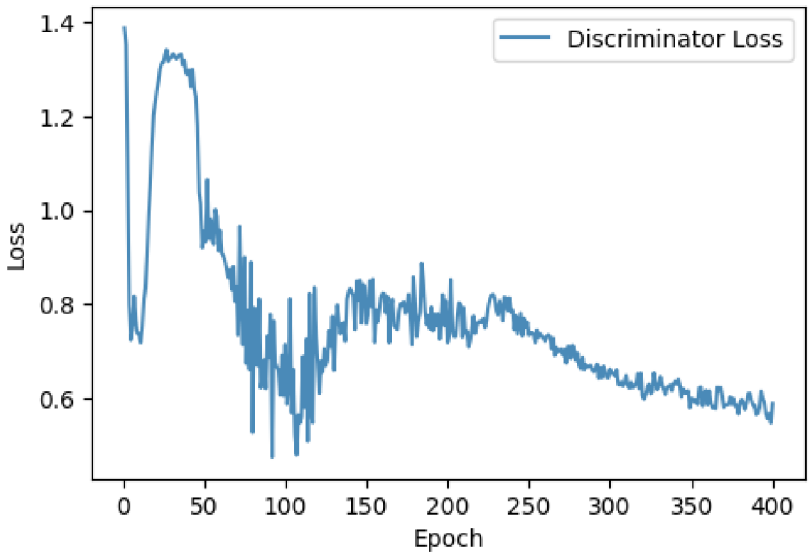


Figure 6.2: Discriminator loss over epochs for BPI 2012 Dataset

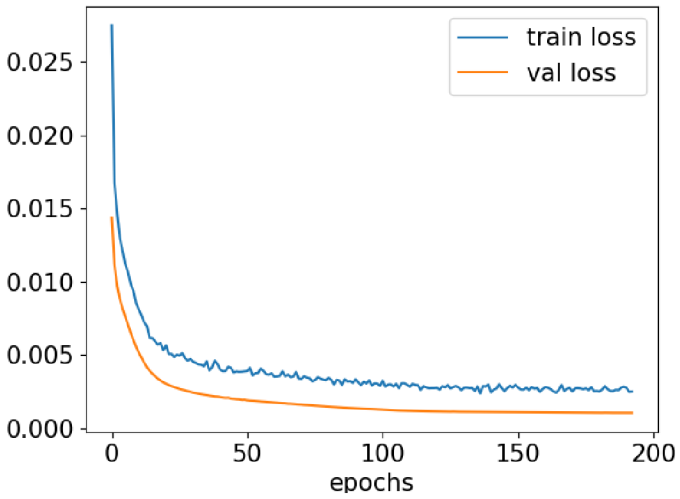


Figure 6.3: Training and Validation Loss for LSTM Model for BPI 2012 Dataset

loss value typically conveys that the discriminator is becoming more proficient at identifying real data as opposed to synthetic data produced by the generator.

Figure 6.3 displays the training progression of an LSTM network, with both training and validation loss depicted over 200 epochs. Initially, there is a steep decline in loss, indicating rapid learning from the sequential data. Losses quickly converge and remain tightly coupled, suggesting an effective generalization without signs of overfitting. As training progresses, the

loss values plateau near zero, which points to the LSTM's capability to capture and predict the patterns in the dataset with high accuracy.

6.2 Sequence Length Comparison

Evaluation of sequential data is not as intuitive as image data. Synthetic images are usually evaluated on the basis of the authenticity of the resolution, which is easy to observe. The aim here is to generate synthetic process data that follow the underlying distribution of the real-world process. In a sequence-length comparison to assess the fidelity of synthetic sequences, this analysis compared the probability distribution of the sequence lengths across both datasets. Figure 6.4 and Figure 6.5 show a bar plot comparing the sequences of Real, LSTM, and GAN for the helpdesk and BPI 2012 datasets respectively.

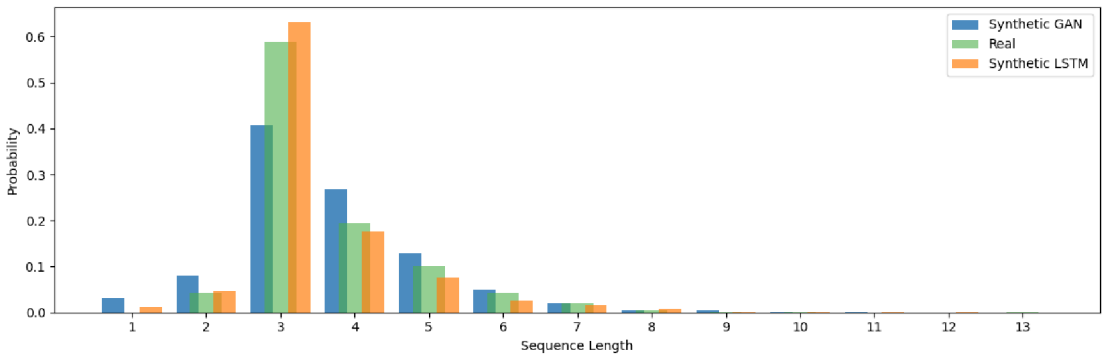


Figure 6.4: Comparison of Probability Distribution of Sequence Length of Real and Synthetic data for Helpdesk Dataset(Sequence Length=13)

In the analysis of LSTM-generated sequences, the outcomes closely align with the real data for both datasets, particularly with shorter sequences, capturing the distribution of the real data effectively. However, when the performance of GAN is assessed against both the BPI2012 and the helpdesk datasets, it reveals a marginally lower similarity to real data compared to LSTM, but it exhibits a greater diversity in sequence lengths. This diversity suggests the ability of GAN to create new sequences. An example of such a sequence is where the GAN-generated sequence incorporated the additional repetitive steps not found in the original data[22].

For instance, a real sequence detailed as "A SUBMITTED → A PARTLYSUBMITTED → A PREACCEPTED → W Completeren aanvraag → W Completeren aanvraag → A DECLINED → W Completeren aanvraag" in BPI2012 dataset is expanded in the GAN-generated sequence to include more "W Completeren aanvraag" steps, leading to a sequence like "A SUBMITTED → A PARTLYSUBMITTED → A PREACCEPTED → W Completeren aanvraag (x4) → A DECLINED → W Completeren aanvraag", which is also a valid sequence. This process showcases GAN's strength in generating new, plausible patterns by repeating specific steps and introducing sequences not originally present in the dataset. The novelty of the generated data needs

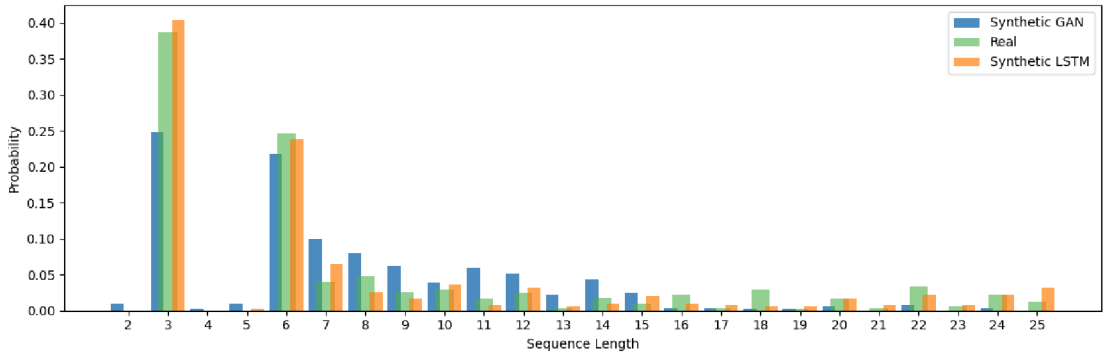


Figure 6.5: Comparison of the Probability Distribution of Sequence Length of Real and Synthetic data for BPI 2012 Dataset(Sequence Length =25)

to be observed with further experiments to understand the usability of these data to train the ML-based process mining tools.

6.3 Sequence Variance Analysis

Sequence variance is calculated using the sum of the pairwise normalized Edit distance (SPE). It is calculated between all pairs of activity sequences for both real and synthetic data using the Levenshtein distance, also called the edit distance. In the analysis (Table 6.1), the LSTM-generated process sequences exhibited variance scores closer to authentic data (0.204 for BPI and 0.156 for Helpdesk) compared to those generated by the GAN model (0.231 for BPI and 0.183 for Helpdesk).

Data Source	BPI 2012	Helpdesk
Authentic	**0.196**	**0.165**
LSTM	0.204	0.156
GAN	0.231	0.183

Table 6.1: Comparison of Sequence variance of Synthetic and Authentic Data.

LSTM model demonstrated a capability to replicate the variance of authentic data, suggesting their effectiveness in capturing the sequential dependencies typical of process sequences. This was particularly evident in the Helpdesk dataset, where the LSTM variance was slightly lower than that of the authentic data. The GAN model, although it generates higher variance scores than the LSTM model and the authentic data, indicates the ability to explore a wider range of process variations. Although this did not meet the initial goal of closely mimicking authentic data variance, it underscores GANs' potential to enrich process mining analyses by providing diverse data coverage, ensuring that the generated sequences reflect the variety of actual process variations.

6.4 Kullback-Leibler (KL) divergence

The KL divergence values indicate how the probability distribution of the synthetic data (generated by the models) diverges from the actual (authentic) data distribution in the given datasets.

Model	BPI2012	Helpdesk
LSTM	2.62	0.80
GAN	3.03	2.76

Table 6.2: Comparison of KL Divergence of Synthetic and Authentic Data

When the probability distribution of the complete sequences of real and synthetic data is compared, LSTM generated sequences show lower KL divergence values (2.62 for BPI2012 and 0.80 for Helpdesk), indicating that its synthetic data is closer to the actual data, particularly for the Helpdesk dataset. This suggests that LSTM is effective in capturing temporal patterns in the processing of data. Synthetic data generated from GAN(3.03 for BPI2012 and 2.76 for Helpdesk) has a higher KL divergence that showcases its effectiveness in balancing the need for similarity with the need for differences. Although a low KL divergence is desirable to ensure that the synthetic data are representative of the real data, a nonzero divergence is expected and beneficial. GAN is known for its ability to generate diverse and novel data samples. This capability can be particularly beneficial in process mining for exploring a wider range of process variations and enhancing model robustness against overfitting. The diversity in GAN results can be crucial for stress testing process mining algorithms or for augmenting datasets where the actual data is limited or lacks variability.

6.5 Activity Type Occurrence Comparison

In this experiment, to understand the distribution of activity type in the data set, a comparison is made between the probability distribution of all activities and the activities generated from the models. For the BPI 2012 Dataset (Figure 6.6), it can be observed that for certain activities, the LSTM and GAN Synthetic datasets closely match the Real dataset's probabilities, while for others, there are discrepancies. For GAN, Activities, such as "W_Completeren aanvraag" and "W_Afhandelen leads" and "A_PREACCEPTED" have been represented with a higher frequency compared to the real dataset and the LSTM synthetic dataset. When comparing the start activity "A_SUBMITTED" for the BPI 2012 dataset, which is common for all sequences, both the LSTM and GAN synthetic datasets replicate this start activity with close probabilities of the real dataset, suggesting that the models effectively capture the initiation patterns of the process. Other activities show notable differences in the probability of activities between the real and synthetic datasets, which could be due to the synthetic models not capturing all the complexities of the data or due to intentional variations introduced in the synthetic data generation process. Both the LSTM and GAN models demonstrate proficiency in replicating rarer activities found within the real dataset, a crucial capability to ensure complete process representation. For the helpdesk data set, both models showed activity similarity to the real

data. This could be due to the shorter sequence length and the small vocab size of 9.

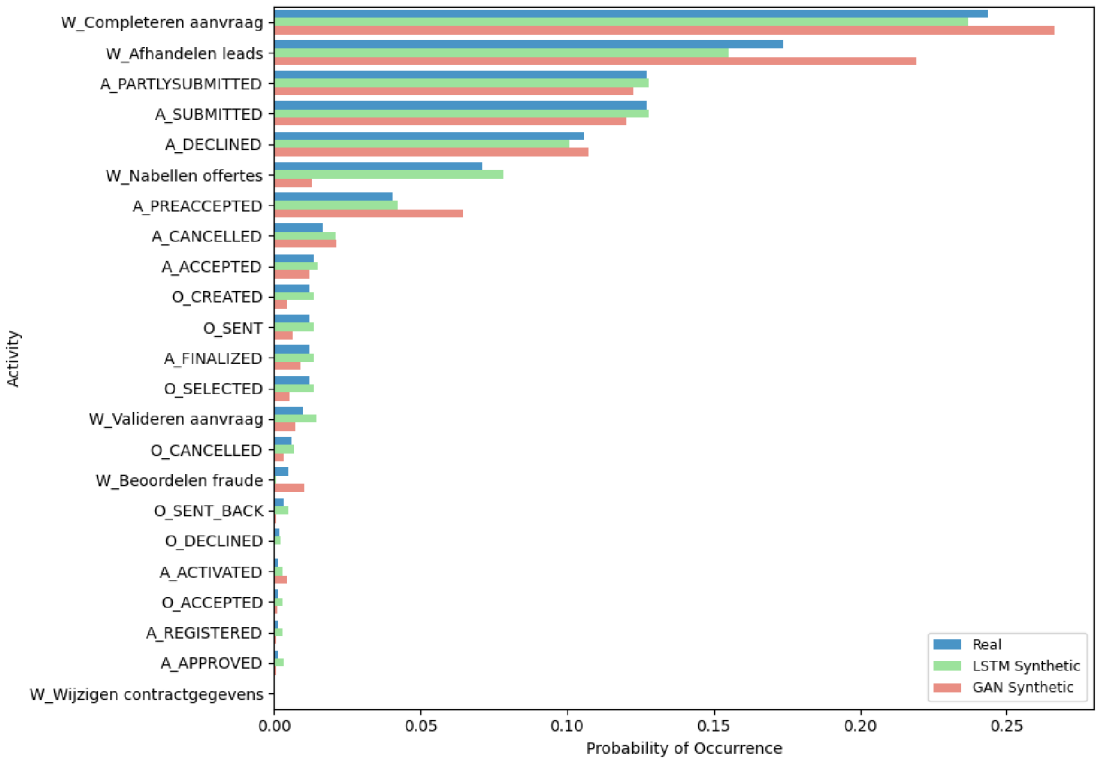


Figure 6.6: BPI 2012 Dataset Activity Type Occurrence

6.6 Unique Data Comparisons

This section analyzes the uniqueness of sequences generated by LSTM and GAN models. Table 6.3 summarizes the comparisons of unique sequences. Each row represents a model, and each column shows the number of sequences for a specific category within a data set.

- Real: Sequences present in the original dataset only.
- Overlap: Sequences found in both original and synthetic datasets.
- Synthetic: Sequences unique to the synthetic dataset and model

The LSTM model appears to be more conservative in generating synthetic sequences, with a higher overlap and fewer generated unique sequences. This could mean that LSTM is better at capturing and replicating the existing patterns in the data without introducing as much novel variability.

Model	Unique Sequences	Helpdesk	BPI2012
	Real	45	64
LSTM	Overlap	29	33
	Synthetic	28	59
GAN	Overlap	26	31
	Synthetic	255	169

Table 6.3: Comparison of Unique Sequences generated by LSTM and GAN on Helpdesk and BPI2012 Datasets (Sample Size =500

GAN model demonstrates a substantial number of generated Unique Sequences—255 for Helpdesk and 169 for BPI2012 suggests that while it may be creating many novel sequences, these sequences are likely to contain subsequences or patterns that are derived from the original dataset. The GAN's ability to generate such a large volume of unique sequences while still maintaining a reasonable overlap with the original data (26 for Helpdesk and 31 for BPI2012) indicates that it is not just inventing random sequences, but rather it is recombining elements of the original data in new ways. Visual Inspects and Domain Specific Validity Checks can be applied to assess the validity of the generated sequences, to help differentiate them from noisy data.

6.7 Process Flow Analysis

This section presents a detailed analysis of process flow diagrams derived from a real data set and synthetic data generated using Long Short-Term Memory (LSTM) and Generative Adversarial Networks (GAN) models. The BPI 2012 dataset served as the foundation for this comparative study due to the availability of the activity names compared to the Helpdesk dataset which was anonymized, with an emphasis on structural integrity, data distribution, and model fidelity in reproducing complex process flows.

The synthetic data generated by LSTM and GAN models were first assessed for structural congruence with the original process flow. Although both models preserved the main pathways, such as 'A_SUBMITTED' to 'A_ACCEPTED', certain discrepancies were observed. The LSTM model introduced fewer deviations, suggesting a more conservative approach in sequence generation. In contrast, the GAN model demonstrated a tendency to explore a broader range of state transitions, occasionally introducing novel paths not present in the original data. Upon examination of the data distribution within the flows, the LSTM-generated diagram closely mirrored the original data's quantities at each step. However, the GAN model exhibited a marked variance, particularly in the bottleneck stages. These differences were hypothesized to arise from the GAN's mode of operation, which focuses on generating new data points rather than replicating the statistical properties of the input data. However, both models successfully replicated common patterns within the process flow, such as the frequent transition from

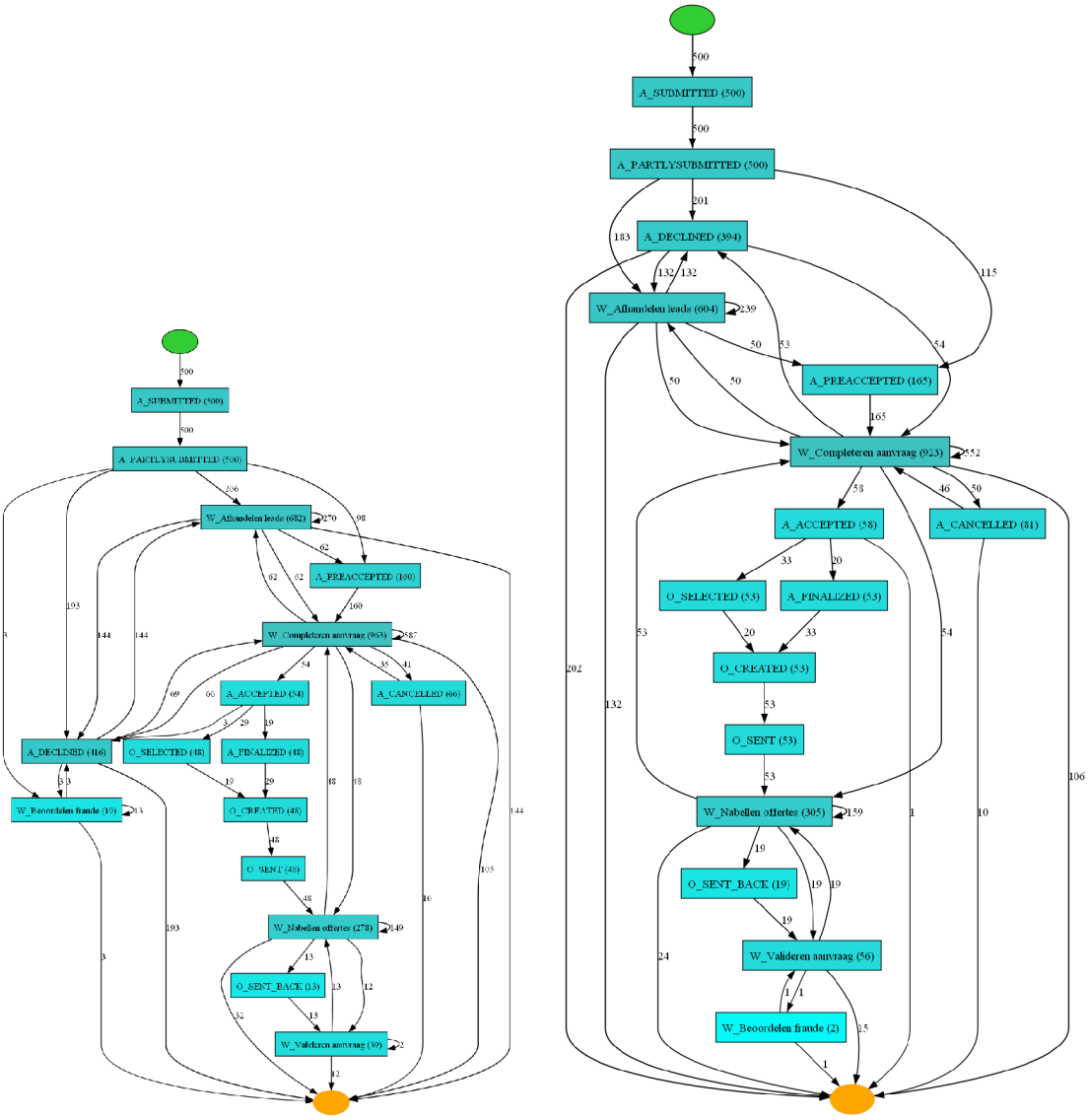


Figure 6.7: Process model by Heuristic Miner with BPI2012 real sequences (left) and LSTM synthetic sequences (right)

'A_PARTLYSUBMITTED' to 'A_PREACCEPTED'. In terms of model fidelity, the LSTM model displayed superior performance in replicating the complexity and variability of the original process flow. The GAN model, though less accurate, offered valuable information on alternative process pathways and outcomes.

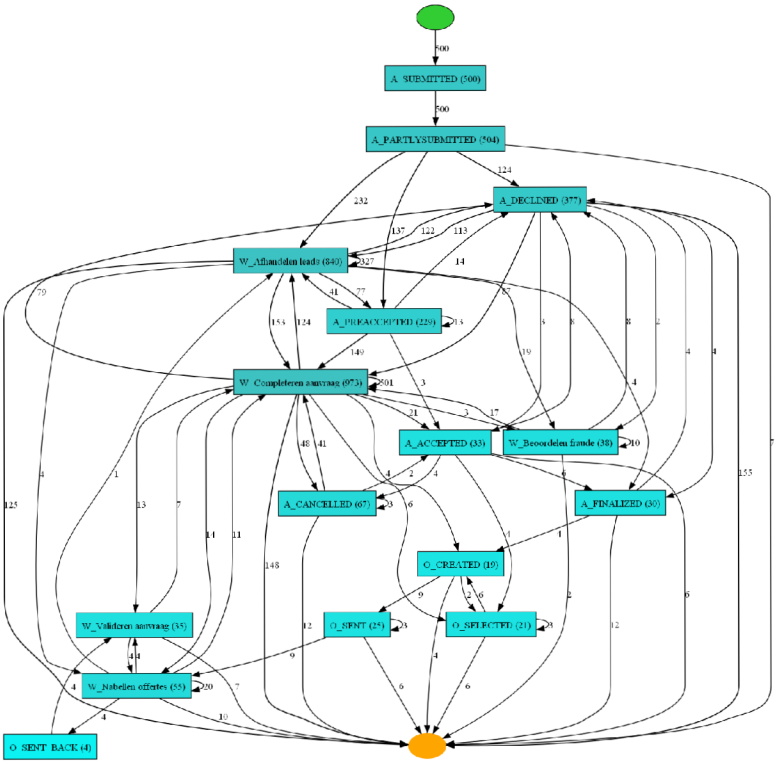


Figure 6.8: Process model by Heuristic Miner for BPI 2012 dataset with GAN generated synthetic sequences

7 Results and Discussions

Throughout the study, a detailed comparison was conducted between the efficiencies of Long Short-Term Memory (LSTM) networks and Generative Adversarial Networks (GANs), aimed specifically at synthesizing process log data for two most popular process mining datasets namely Helpdesk and BPI 2012. This analytical exercise spanned a broad spectrum of evaluative criteria, including the fidelity of data replication, the assessment of activity frequencies and sequence length, variance in sequence patterns, and the examination of unique sequence generation. Additionally, process flow diagrams were employed to provide a visual representation and deeper understanding of the pathways inherent in the generated process logs, facilitating a comprehensive grasp of the procedural dynamics.

The experiments show that, both LSTM and GAN models were capable of replicating common patterns within the process flow. However, the traditional LSTM model displayed high fidelity in replicating the original dataset's patterns, particularly adept at maintaining the sequence lengths and activity types closely matching the original data for both the Helpdesk and BPI2012 datasets. Due to its high data fidelity, LSTM models are more immediately applicable for training purposes where replicating exact sequences is necessary. They can work for situations requiring high accuracy and consistency with the original process behaviors. The GAN model with a transformer-encoder-based architecture demonstrates a unique strength to generate previously unseen sequences that extend beyond the patterns and structures present in the training datasets. GAN while slightly less accurate in replicating the exact sequence patterns of the original data, introduces a higher degree of novelty and variability. This is beneficial for generating diverse datasets that can simulate a wider range of scenarios, including rare events not covered in the training data.

For sequence length comparison, the LSTM-generated sequences aligned closely with real data for both datasets, particularly for shorter sequences. However, GAN displayed a slightly lower similarity to real data but exhibited greater diversity in sequence lengths, suggesting their capability to generate new, plausible sequences. Regarding sequence variance, LSTM-generated process sequences exhibited variance scores closer to the authentic data, especially in the simpler Helpdesk dataset. GANs, while generating higher variance scores, indicate their ability to explore a broader range of process variations, which could enrich process mining analyses by providing diverse data coverage. The LSTM model showed lower KL divergence values (2.62 for BPI2012 and 0.80 for Helpdesk), indicating that its synthetic data is closer to the actual data distribution, particularly for the Helpdesk dataset. This underscores the effectiveness of LSTM's in capturing temporal patterns in process mining data. GAN-generated data had higher KL divergence values (3.03 for BPI2012 and 2.76 for Helpdesk), showcasing its effectiveness at balancing the need for similarity with the introduction of differences. This characteristic of GANs

can be particularly beneficial for exploring a wider range of process variations and enhancing model robustness.

Through the process flow diagram, both models were assessed for structural congruence with the original process flow. The LSTM model introduced fewer deviations, suggesting a more conservative approach to sequence generation. The GAN model demonstrated a broader range of state transitions and occasionally novel paths, offering valuable insights into alternative process pathways.

In conclusion, while LSTM models offer high precision and fidelity to the original data, making them ideal for applications requiring exact data replication, GAN models stand out for their ability to generate diverse and novel data, pushing the envelope on what process mining algorithms can recognize and analyze. The choice between LSTM and GAN would, therefore, depend on the specific needs of the process mining task, whether it prioritizes accuracy and consistency or diversity and the exploration of new process variations.

8 Conclusion and Future Work

In conclusion, this thesis has explored an in-depth exploration of synthetic data generation for process mining, focusing on the comparison and application of LSTM and GAN models. The suggested approach, learns deep representations of process data to create a generative model, allowing the production of synthetic data useful for training Machine learning-based process mining tools. By integrating advanced computational techniques and rigorous methodological approaches, this thesis not only responds to the identified gaps from prior studies but also pushes forward the understanding and application of synthetic data generation in process mining. While the research has demonstrated the potential of these models to replicate and innovate process data, it is acknowledged that the study was conducted on relatively small datasets due to data availability and computational limitations. The constraints on dataset size were instrumental in ensuring a manageable computational load and focused analysis, yet they also posed restrictions on the generalizability of the findings. Despite these constraints, the research demonstrated the effectiveness of advanced generative models, including LSTM and GAN, in synthesizing process data that maintains the essential characteristics of real datasets. The nuanced comparison of LSTM and GAN models within this thesis serves as a significant contribution to the field, offering guidance on model selection based on specific requirements for fidelity, diversity, and innovation in data generation.

The detailed experimentation and analysis have illustrated that LSTM models are particularly effective in replicating the precise structure and sequence of the original data, which is crucial for applications demanding high fidelity and reliability. These models have shown their strength in maintaining consistency across sequence lengths and activity frequencies, closely mirroring the actual data from the Helpdesk and BPI 2012 datasets. This precision supports LSTM's suitability for training scenarios where the exact replication of data sequences is vital. Conversely, the GAN model utilizing transformer-encoder architectures with customized activity loss functions for process data, have demonstrated their effectiveness in generating novel and diverse data. While they may not match the exactness of sequence replication found in the LSTM model, they excel in producing a variety of plausible sequence lengths and introducing novel sequences. This capability is invaluable for expanding the scope of process mining by simulating scenarios that may not be present in the training datasets and for enhancing the existing datasets, thus providing a broader perspective for analysis and decision-making.

Expanding upon the achievements of this thesis, future research can unfold in multiple directions, all aimed at enriching the utility and application of synthetic data in process mining. To validate and generalize the findings, applying the generative models to a diverse and complex set of datasets spanning various industries and process types is crucial. This broader scope will reveal the models' adaptability and robustness, providing deeper insights into their scalability

and effectiveness. Involving domain experts to critically assess the generated data's realism and relevance can significantly guide iterative model improvements. Such qualitative assessments ensure the synthetic data adheres to statistical quality metrics and resonates with practical, domain-specific considerations. While this thesis focused on LSTM and GAN, exploring other deep generative models, like Variational Autoencoders (VAEs), holds the potential to unlock new avenues in synthetic data generation. Models like VAEs might offer unique strengths in capturing and replicating complex process dynamics.

Developing nuanced evaluation metrics that incorporate both quantitative and qualitative dimensions is essential. Future work could aim to refine these evaluation frameworks, providing a comprehensive view of the synthetic data's fidelity and applicability. Finally, integrating the generated data with existing process mining tools and workflows can offer valuable insights into its operational value. This practical understanding will reveal how synthetic data can enhance process mining analyses, tool development, and ultimately, business process optimization. By addressing these future directions, subsequent research can build on the foundation of this thesis, pushing the boundaries of synthetic data's potential and promoting a deeper integration of these innovative methodologies within the process mining domain.

Bibliography

[1] W. van der Aalst, "Process mining: Overview and opportunities," *ACM Trans. Manage. Inf. Syst.*, vol. 3, no. 2, jul 2012. [Online]. Available: https://doi.org/10.1145/2229156.2229157

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[3] D. P. Kingma and M. Welling, "An introduction to variational autoencoders," *Foundations and Trends® in Machine Learning*, vol. 12, no. 4, p. 307–392, 2019. [Online]. Available: http://dx.doi.org/10.1561/2200000056

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[5] W. M. Van Der Aalst, H. A. Reijers, A. J. Weijters, B. F. van Dongen, A. A. De Medeiros, M. Song, and H. Verbeek, "Business process mining: An industrial application," *Information systems*, vol. 32, no. 5, pp. 713–732, 2007.

[6] S. Yang, Y. Zhou, Y. Guo, R. A. Farneth, I. Marsic, and B. S. Randall, "Semi-synthetic trauma resuscitation process data generator," in *2017 IEEE International Conference on Healthcare Informatics (ICHI)*. IEEE, 2017, pp. 573–573.

[7] Y. Zisgen, D. Janssen, and A. Koschmider, "Generating synthetic sensor event logs for process mining," in *International Conference on Advanced Information Systems Engineering*. Springer, 2022, pp. 130–137.

[8] D. Sommers, V. Menkovski, and D. Fahland, "Process discovery using graph neural networks," in *2021 3rd International Conference on Process Mining (ICPM)*. IEEE, 2021, pp. 40–47.

[9] Z. A. Bukhsh, A. Saeed, and R. M. Dijkman, "Processtransformer: Predictive business process monitoring with transformer network," *arXiv preprint arXiv:2104.00721*, 2021.

[10] D. A. Neu, J. Lahann, and P. Fettke, "A systematic literature review on state-of-the-art deep learning methods for process prediction," *Artificial Intelligence Review*, pp. 1–27, 2022.

[11] F. Taymouri, M. L. Rosa, S. Erfani, Z. D. Bozorgi, and I. Verenich, "Predictive business process monitoring via generative adversarial nets: the case of next event prediction," in *Business Process Management: 18th International Conference, BPM 2020, Seville, Spain, September 13–18, 2020, Proceedings 18*. Springer, 2020, pp. 237–256.

[12] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," *arXiv preprint arXiv:1706.02633*, 2017.

[13] E. Choi, S. Biswal, B. Malin, J. Duke, W. F. Stewart, and J. Sun, "Generating multi-label discrete patient records using generative adversarial networks," in *Machine learning for healthcare conference*. PMLR, 2017, pp. 286–305.

[14] K. Li, S. Yang, T. M. Sullivan, R. S. Burd, and I. Marsic, "Generating privacy-preserving process data with deep generative models," 2022.

[15] L. Yu, W. Zhang, J. Wang, and Y. Yu, "Seqgan: Sequence generative adversarial nets with policy gradient," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 31, no. 1, 2017.

[16] I. Verenich, "Helpdesk," 2016.

[17] B. van Dongen, "Bpi challenge 2012," 2012. [Online]. Available: https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204/1

[18] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[19] "Pytorch." [Online]. Available: https://pytorch.org/docs/stable/_modules/torch/nn/modules/transformer.html#TransformerEncoder.forward

[20] "Pytorch." [Online]. Available: https://pytorch.org/tutorials/beginner/dcgan_faces_tutorial.html

[21] "Pytorch." [Online]. Available: https://pytorch.org/docs/stable/functional.html#torch.nn.functional.gumbel_softmax

[22] P. Lam, H. Ahn, and K. Kim, "Discovering redo-activities and performers' involvements from xes- formatted workflow process enactment event logs," *KSII Transactions on Internet and Information Systems*, vol. 13, pp. 4108–4122, 08 2019.