Fakultät Angewandte Informatik (Faculty of Computer Science)

Master Artificial Intelligence and Data Science

Topic

“A Comparision of GNNs and LSTMs for Process Mining Applications”

“Ein Vergleich von GNN und LSTM für Process-Mining-Anwendungen”

Master’s thesis in fulfillment of the requirements for the degree of:

Master of Science

at

Deggendorf Institute of Technology

Submitted by:
Surname, given name: Khadka, Aayam

First supervisor:
Mr. Markus Eider

Matriculation number: 12201267

Place, date: Deggendorf, 01.26.2024

Declaration of Authorship

Name of student: Aayam Khadka

Name of supervisor: Mr. Markus Eider

Thesis topic:

A Comparison of GNNs and LSTMs for Process Mining Applications.

1. I hereby declare that the thesis has been written in compliance with Section 35 (7) RaPO (State Examination Regulations in BavariaBayRS 2210-4-1-4-1-WFK) is my own work, has not been submitted for any other degree at any other university or institution, does not contain or use any sources or resources other than those referenced, and that all direct and paraphrased quotes have been duly cited as such.

[illegible]

Deggendorf, 01.26.2024

(Date) (Signature of student)

Table of Contents

1 Introduction

1.1 Preamble

A business organization executes multiple processes every day to accomplish its goals such as customer satisfaction, higher profit, and productivity. The proper execution of these processes relies on the effective, efficient, and transparent execution of processes and it directly correlates to the organization's operation. The better the execution, the better the flow of operations in an organization. While these processes are executed by different departments within an organization, there arise inconsistencies, discrepancies, and redundancies that affect the profitability and efficiency of the organization. Profitability may be affected by factors such as evolving business goals, the need for faster transfer of information, and an increase in competitors. Due to the constantly evolving process models and operations, the process models are increasingly leaning towards more complexity and variability. These affect the operation of an organization. To mitigate these, companies use information management systems such as Enterprise Resource Planning (ERP), Supply Chain Management (SCM), and Relationship Management System (CRM) which collect large numbers of event log data. The event log data is used in process mining to derive process graphs that provide insights and analytical information about the process execution. This information is valuable in identifying variations and assisting in monitoring which improves the process flow by resulting in the efficient and profitable operation of business processes in an organization.

The research is motivated by the aims and objectives of the KIGA project emphasizing the importance of artificial intelligence-supported predictive business process mining applications. High-level processes such as Order to Cash, and Purchase to Pay event log data may contain redundancy, and irregularities that are crucial therefore the research aims to provide valuable insights into the effectiveness of deep learning methods in process mining and additionally support the KIGA project in applying predictive techniques to event log data.

1.2 Research Question

The focus of the research question is on the comparative analysis of two models, GCN and multi-task LSTM regarding event log data. The study aims to evaluate the implementation of these models in future event prediction and time prediction. The key aspects include:

Model Performance and Evaluation

The research question evaluates the performance of GCN models and multi-task LSTM networks in accurately predicting both events and related timestamps of both the test data set and the KIGA dataset. Furthermore, it highlights the comparative nature of the study on the performance of these two models. It aims to contrast and compare the performance of these models to realize their potential in predictive business process mining applications.

Performance Metrics

The performance metrics include accuracy for the event prediction whereas MAE Mean Average Error for the time prediction. At the same time, it further emphasizes the computational efficiency of both models as well.

1.3 Methodology

1.3.1 Data preprocessing

The data provided for the KIGA project is real life process event log data from a business organization. This gives a possibility that the data requires thorough inspection and preprocessing of any inconsistent, and redundant data. The most likely case is that the events are recorded multiple times for different timestamp records, in this case, only the last recorded event will be kept as event is not considered as completed until last event is recorded. Additionally, any null values or missing values are checked and maintained.

1.3.2 Feature Encoding

To keep the comparison of both models justifiable, it is important that both model use similar features from the data. Although they differ in their model architecture and learning procedure, the input feature vectors can be kept similar. Therefore, to capture timestamp values comprehensively, 4 temporal dependencies are considered which are: Time since the previous event in the case, Time since the start of the case, Time since midnight (of the day), Day of the week. For event, the unique events are one-hot encoded to get numerical representation.

1.3.3 Training procedure

1.3.3.1 GCN

The GCN captures the graphical information of the dataset through each iteration. Initially, it is required to represent data into graphical representation, therefore, Directly Follows Graph is used. This gives information about the nodes and edges and their relationship by representation of weights which is then captured using adjacency matrix. The weight vectors for events in cases are then iteratively updated in the GCN layer using adjacency matrix and input features.

1.3.3.2 LSTM

For multi-task LSTM, preprocessing, cross-validation set, and input features are kept the same. However, the multi-task LSTM model will be able to learn event parameters and timestamp parameter through single model.

1.3.4 Evaluation

The evaluation and comparison are carried out using overall accuracy and mean absolute error of the predicted results as well as metrics for different prefix length of events and time values. Results for different prefix length gives detailed look into how model performance are affected by the number of prefix lengths in event and timestamp. It gives information about the effect of length of event or timestamp in predictive performance. All in all, the overall metric as well as results for different values or length of prefix sequence are considered.

1.4 Thesis Structure

Chapter 2: Background. This chapter presents the background information on the major topics within the thesis. These topics contain Process Mining, GNNs, and LSTMs.

Chapter 3: Related Work. This chapter presents the related work on implementing deep learning methods, GNNs, LSTMs, and KIGA data in event and time prediction.

Chapter 4: Methodology. This chapter describes the data, and preprocessing steps, and explains the methods used in training the GCN and LSTM models for event prediction and time prediction.

Chapter 5: Implementation. This chapter explains in detail about the training procedure for GCN and LSTM models in event and time prediction.

Chapter 6: Evaluation and Discussion. This chapter explains the results of the approach and presents its evaluation based on the research question. A discussion on the limitations, and improvements is conducted and a pathway for future work is provided.

Chapter 7: Conclusion and Outlook. This chapter concludes the research and summarizes the complete approach.

2 Background

This chapter includes information and description of background information regarding dataset, topics of process mining and overview of the GNN and LSTM model architecture.

2.1 Data

As mentioned, the datasets used in the experiment consist of event log data of Order-to-Cash and purchase-to-pay processes. The dataset is extracted from a real-life event log of the mentioned processes of a business organization.

- Order to Cash
  This process is one of the fundamental business processes that include information about the receipt of Customer Orders to Receive the payment for the delivered goods or services. In general, it represents how a company receives, manages, and completes the order placed by the customer. It includes the cycle of events after the order placed by the customer to the payment of the order and hence the conversion of the order to cash [1]. A common step in Order-to-Cash contains Order Placement, Order Processing, Inventory Check, Shipping, Invoice, Payment, and Order Completion.

  The benefits of using the O2C process include improved cash flow, customer satisfaction, reduced order processing costs, visibility into sales performance, compliance and risk management, faster order fulfillment, and inventory management.

- Purchase to Pay
  Purchase to Pay (P2P) also known as Purchase to Pay is another fundamental business process that organizations follow to buy goods and services from external vendors. These goods and services in turn are converted to finished products and provided to customers. It represents the cycle of events from identifying the need for goods or raw materials to payment and record keeping. The events in Purchase-to-Pay include Purchase order, Supplier selection, Order Receipt, Invoice Verification, Approval, Payment, and Reporting or Record Keeping.

The usage of the P2P process provides organizations with the benefit of cost savings, vendor relationship management, efficient procurement cycle time, transparency, decision making, invoice processing, and risk management [2].

2.2 Process Mining

Process Mining, a subfield of BPM emerged to assist in process discovery, and modeling and improve workflows using process information in the event log. This helps organizations improve efficiency, reduce operational costs, and improve the overall quality of services. It assists in BPM life cycle activities by reducing the need for manual processing of the business processes. This allows organizations to improve process flow, re-modeling, and re-engineering of process models which provides organizations with factual representation of their processes to achieve the needed outcomes.

Process Mining techniques can be both forward-looking and backward-looking. [2] Backward-looking is related to techniques related to finding the root causes of bottlenecks and discrepancies in a process whereas forward-looking is related to the prediction of future events or prediction of time differences in future events in a case.

In essence, Process Mining answers the following questions.

- What are the processes that are followed?
- How does the actual process execution differ from the executed process?
- What are the deterrents in the process?

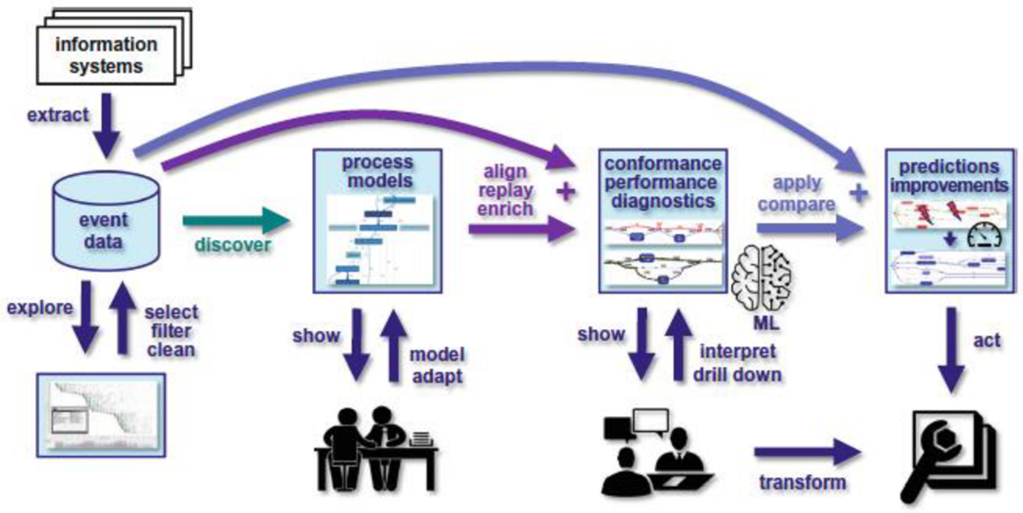


Figure 1: Overview of Process Mining showing cycle of process mining showing the complete cycle from extracting event logs, building models, analyzing, interpreting, and making decisions [2].

2.2.1 Types of Process Mining

Process Mining may consist of different objectives. Depending on the goal and objectives, different types of process mining are used. Some of the most common types are:

Process discovery. This represents different techniques and measures to discover a process model based on the event log data. For example, the usage of Directly Follows Graph (2.4.5) where the nodes and vertices are represented as events and relations respectively. However, the model is prone to overfitting and underfitting during the representation. Therefore, different algorithms are proposed based on diverse tasks. In particular, process discovery algorithms take an event log as input and produce a process model that needs to possess several integral attributes of an optimal process model [3]. More commonly, these attributes are represented as:

- Recall. The discovered model should be able to show the behavior as shown by the event log.
- Precision. The model should not deviate from the actual behavior observed in the event log.
- Generalization. The discovered model should generalize the behavior and avoid overfitting.

- Simplicity. The model should be able to represent the behavior with an overly complex model. This is related to Occam's Razor which states "One should not increase, beyond what is necessary, the number of entities required to explain anything".

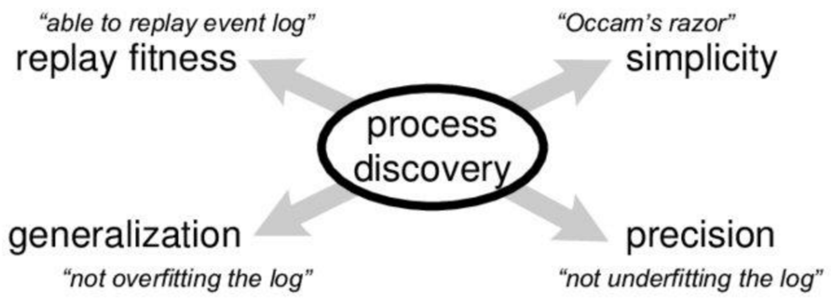


Figure 2: Attributes of process discovery that are important for creating process models. [3].

Conformance Checking. This represents the comparison of the extracted model with the actual data in the event log. The goal of conformance checking is to represent the degree of agreement or disagreement between the process model and the information contained in the event log [2]. The techniques used in the conformance check provide information about the discovered behavior of the process and the modeled process.

Model enhancement. It involves the techniques used to improve the process model using the information about the actual process contained in an event log. Extraction of information from event logs may reveal bottlenecks, discrepancies, and deviations of the process model from the actual process. This extracted information is used to analyze and optimize the process model in order to correctly align it with the actual process execution. It reveals transparency and brings out factual insights from event logs helping to eliminate bottlenecks and cleaning of redundant events and activities [2]. As a result, the model is more refined because of the improvement in accuracy and efficiency.

Predictive Process Mining. This type of Process Mining relates to the predictive nature of the model. While previous types are referred to as backward-looking, predictive process mining represents the forward-looking aspect of process mining. Sudden changes in process, performance issues, and bottlenecks may occur which can be mitigated by using predictive analysis from predictive process mining. The resulting predictive analysis in combination with machine learning techniques allows forecasts about the likely behavior of the process. This allows the organizations to take precautionary measures, provide informed decisions, and ensure an efficient flow of process execution.

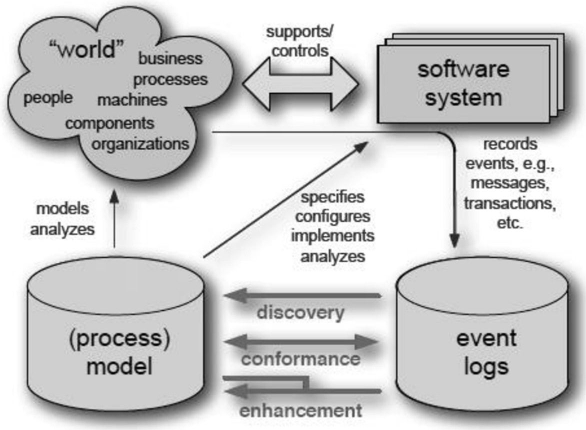


Figure 3: Common types of process mining. i) discovery; ii) conformance; iii) enhancement [4]

2.2.2 Event Log

An event in an event log refers to an instance of the case/process, a task, or a point in time. It contains multiple attributes such as case, activity, timestamp, resource, and customer. The most common attributes include case, activity, and timestamp where a case refers to a process instance, activity to the operation or task, and timestamp to the instance when the case is instantiated. It gathers information about the type of activity performed, the responsible entity, and the moment of event execution [2]. Typically, cases are identified using an identifier ID that represents a unique number for each case where events belonging to the same case are grouped by the case identifier. There are three possible types of event logs [4]:

1. Real-life logs. It contains processes recorded from real-life day-to-day organizational activities.
2. Synthetic logs. It contains processes produced from real-life process model.
3. Artificial logs. It contains process information automatically extracted from a non-real-life process model and is manually created to resemble real events.

Case ID	Timestamp	Activity	Resource
001	02-11-2023	Register request	Pete
001	02-11-2023	Check ticket	John
001	02-11-2023	Make decision	Michael
001	02-11-2023	Accept request	Hari
002	02-11-2023	Register request	Hari
002	02-11-2023	Make decision	Marcel
002	02-11-2023	Pay compensation	Elyria

Table 1: Example of an event log.

Table 1 shows a typical form of an event log. Here, each row denotes an event that has occurred whereas each column denotes the attributes of that certain event. As mentioned before the case identifier groups the activities that are related to the same case. Formally, given that all possible activities A are shown including the set of all possible case identifiers C and the set of timestamps T, it can be defined as:

Definition 1: Event. An event e is a tuple where $e = (c, a, t) \in C\ X\ A\ X\ T$, that represents the number of activity a for the case c at timestamp t. Moreover, the complete set of possible events is known as the event universe denoted by $\mathcal{E} = C \times A \times T$.

Definition 2: Trace. If A' denotes all the non-empty finite sequences of activities from A. $\sigma \in A'$ is called a trace when σ represents a sequence of activity of the process model.

Definition 3: Event stream. Given an event set $\mathcal{E}$, S denotes an event stream such that those events are initiated one after the other. It is an indefinite stream of events such that two consecutive events may be associated with different cases in the timeline.

2.3 LSTM

2.3.1 RNN

Neural networks consist of several networks composed of interconnected layers of neurons that can learn patterns and behavior of the input data. RNNs particularly contain recurrent connections with hidden states distributed over time that allow the network to preserve past information. At each time step, the input and previous output are considered to obtain a new output which again is fed into the next time step as shown in Figure 4. As a result, this allows RNNs to learn long-term dependencies in a sequence of data and preserve information about prior information [5].

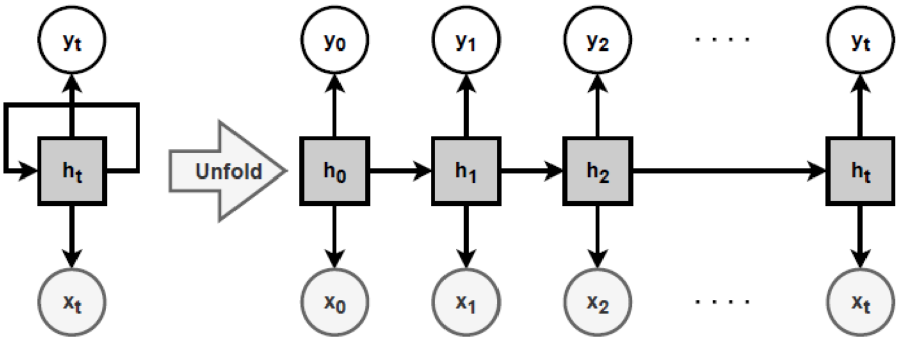


Figure 4: Simple Recurrent Neural Network Architecture showing detailed architecture and process of information transfer [6].

The mathematical equation for a simple recurrent cell is given as follows:

$$h_t = (W_h h_{t-1} + W_X x_t + b),$$

$$y_t = h_t,$$

where x_t, h_t, and y_t states the input, past recurrent information, and output of the cell at time t respectively. W_h and W_X denote the weights and b is the bias. This allows the model architecture to keep information about the context while making a prediction, e.g., next-word prediction in a sentence or classification tasks. However, the amount of information that a simple RNN model can capture is limited due to the problem of vanishing gradients. As past information gets longer, vanishing gradients arise when error signals during backpropagation either blow up or vanish [6].

2.3.2 Long Short-Term Memory Networks

Long short-term memory (LSTM) is an advanced variation of RNN that has been developed to tackle the problem of vanishing or exploding gradients of traditional recurrent neural networks. It contains special LSTM cells as hidden units that are able to retain information and was developed by Hochreiter and Schmidhuber in 1997 [7]. Thereafter, the proposed architecture has been modified and improved multiple times and is considered among the successful recurrent neural network models [6].

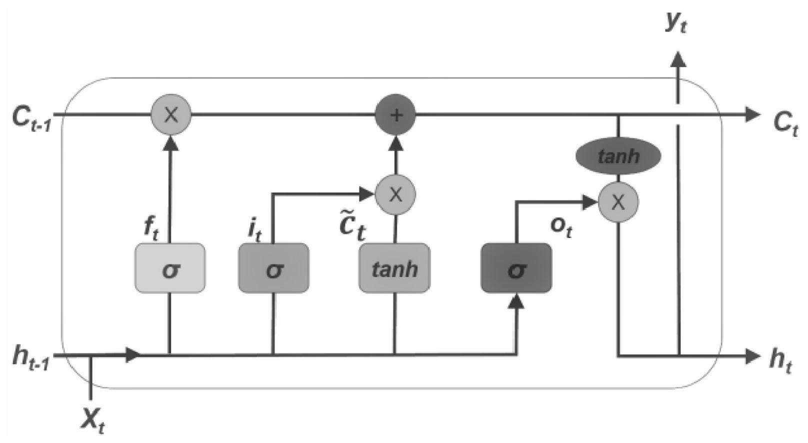


Figure 5: LSTM cell showing its gates [4].

LSTM cells are also called memory blocks and contain three gates that control the cell state. Information from input data is updated or removed from cells by these three gates. These three gates include:

- The Input gate (i) which takes input at the current time step and hidden state of the previous time step to update the cell states and obtain output.
- The Forget gate (f) which also takes input at the current time step and hidden state of the previous time step but controls the discarding of previous information concerning the upcoming input information.
- The Output Gate (o) which considers the input and last hidden states to determine the output.

Combined, the formula for the cell operations can be described as:

$$f_t = \sigma(W_f \,.\, [h_{t-1}, x_t] + b_f)$$

$$i_t = \sigma(W_i \,.\, [h_{t-1}, x_t] + b_i)$$

$$\tilde{C}_t = tanh(W_c \,.\, [h_{t-1}, x_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh(C_t)$$

Taking previous hidden states h_{t-1} and current input x_t as current input, it is passed through forget f_t, which calculates the output for the forget gate. $\tilde{C}_t$ computes a new eligible candidate which is passed for C_t to output a new cell state. i_t uses h_{t-1} and x_t to calculate the output of the input gate. o_t computes the output for the output gate and h_t computes the new hidden state.

Here, W and b represent weight and bias respectively. The activation functions applied in the cell are sigmoid (σ) expressed as $\frac{1}{1+e^{-x}}$ and tanh as $\frac{(e^x - e^{-x})}{(e^x + e^{-x})}$. The weights and biases in the network are iteratively updated to minimize the loss of the of the input and output pairs during training. Epoch is the iteration through the dataset that denotes the amount of forward and backward pass through the network. The parameter of the optimization algorithm is crucial in updating the weights with optimal values [7].

2.4 GNN

2.4.1 Overview of Graph as a Data

Graph data is a type of structured data that represents information using nodes and edges. Graphs are implemented in various tasks relating to node classification, link prediction, and clustering. The key features in a graph are nodes (vertices), and edges that connect the nodes. Edges between the nodes represent the relationship between the nodes. Nodes are also known as entities or objects. The edges are either directed, undirected, weighted, or unweighted depending on the type of relationship between the nodes/entities. The feature vectors are associated with nodes that capture the value.

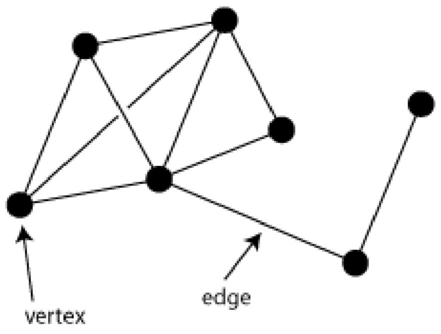


Figure 6: Graph showing nodes/vertices and edges.

Typically, a graph can be represented as $G = (V, E)$, where V denotes the set of vertices and E denotes edges between two vertices. The edges between node i and j are represented by $e_{ij} \in E$. Primarily, graphs can be categorized by several types [8].

- **Directed and Undirected Graph.** A directed graph contains edges that are connected and directed from one node to another. An undirected graph represents the symmetric relationship between the edges as there is no direction. Directed graphs typically provide more information than undirected graphs.
- **Homogenous and Heterogeneous Graph** that has the same type of nodes and multiple types of nodes, respectively.
- **Static/Dynamic Graphs** have input features of the graph that vary with time. This graph contains time information which needs to be carefully considered in these types of graphs.

2.4.2 Graphical Neural Networks

Graphical Neural Networks (GNNs) are deep learning techniques that are able to operate on graph data. Generally, GNNs differentiate themselves in their ability to generate representations of nodes and their relationship represented by edges. The fundamental idea of GNN is to generate representations of nodes, their structure, and feature information by iterative passing of messages between the nodes and updating the feature vector representation [9]. Unlike, CNN that operates on Euclidean data such as images (2D), or LSTM operates on sequential and time-series data, GNN focuses on the non-Euclidean domain of data where the nodes are not arranged in space such as an image [10]. It is particularly used in tasks such as node classification, node and edge prediction, graph regression, classification, and clustering. Graphs can be irregularly shaped and unordered. Nodes in a graph can have variable size and can have variable numbers of neighboring nodes making networks such as CNN difficult to complete the task of convolutions. Therefore, GNNs are better suited for graph data.

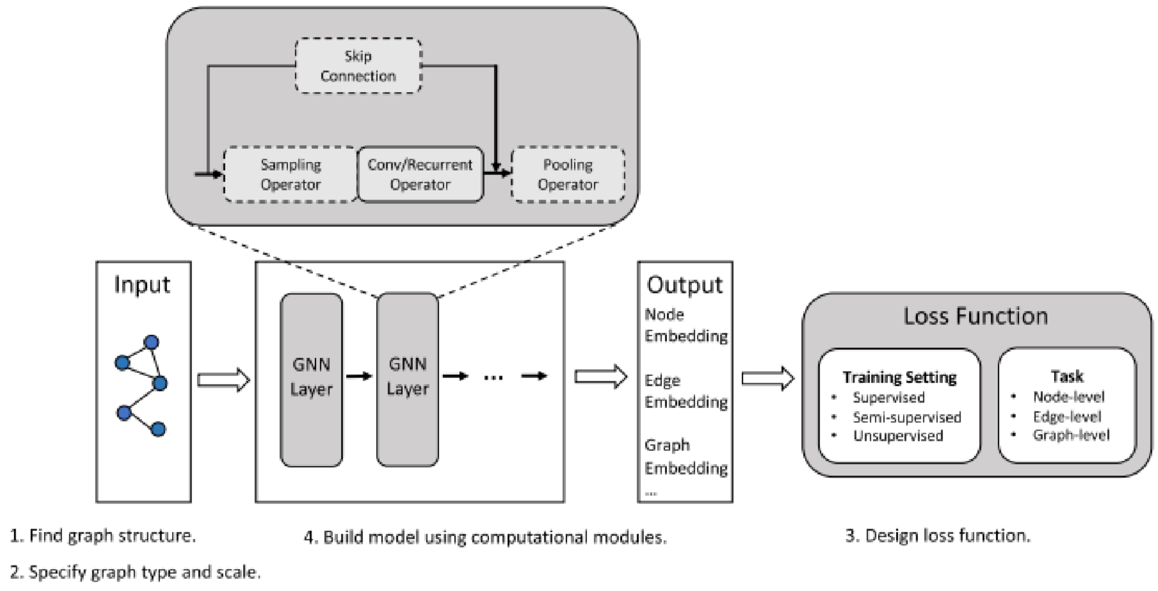
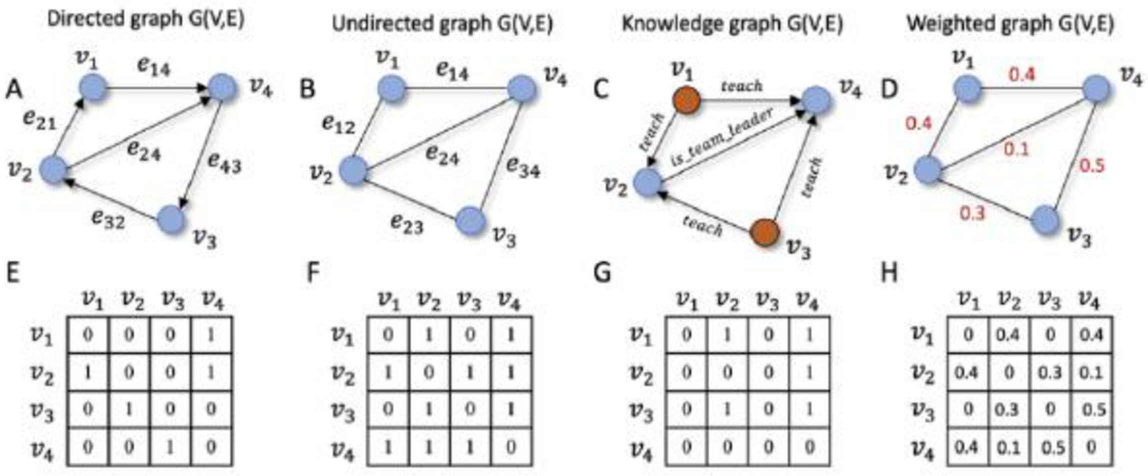


Figure 7: Pipeline of GNN architecture [11].

The pipeline for building GNNs typically consists of the following steps:

- Finding the structure of the graph data
  In this initial stage, the dataset is analyzed to find the structure of the graph data and its scale. Non-structured data such as process mining datasets are transformed into a graph using tools such as pm4py which is able to create graphs such as Business Process Model and Notation (BPMN), Directly Follow Graphs (DFG), Process Tree Graphs, and Petri Net.

- Identifying the graph type
  As mentioned earlier, there are different types of graph data. It is an important part of the GNN pipeline to identify the type of graph. Different types of a graph provide different information and it is a crucial part of the GNN pipeline to consider the additional information provided by each type in the architecture after creating the graph data from the dataset.



E

	v_1	v_2	v_3	v_4
v_1	0	0	0	1
v_2	1	0	0	1
v_3	0	1	0	0
v_4	0	0	1	0

F

	v_1	v_2	v_3	v_4
v_1	0	1	0	1
v_2	1	0	1	1
v_3	0	1	0	1
v_4	1	1	1	0

G

	v_1	v_2	v_3	v_4
v_1	0	1	0	1
v_2	0	0	0	1
v_3	0	1	0	1
v_4	0	0	0	0

H

	v_1	v_2	v_3	v_4
v_1	0	0.4	0	0.4
v_2	0.4	0	0.3	0.1
v_3	0	0.3	0	0.5
v_4	0.4	0.1	0.5	0

Figure 8: Types of graphs with respective adjacency matrix.

In Figure 8, The knowledge graph (C) represents a directed heterogeneous graph that has two different types of nodes that carry directed information. The weighted graph (D) has a value associated with its edges. The adjacency matrix (E-H) gives information about the relationship of a node to other nodes in the graph. The value of the edge between nodes is 1 if and only if there is an edge connecting the nodes or else 0.

- Specification of output task and build model.
  The next step is to specify the type of output task to be conducted. Depending on the learning task to be conducted, there are three types of tasks: Node level, Edge level, and Graph level which will be explained in section 2.3.3.

2.4.3 Types of GNN Output Tasks

- Node level prediction
  Node-level predictions are tasks related to nodes such as node classification, node regression, node clustering, node embedding, etc. Node classification tasks assign a label or category to each node in the graph, Node regression predicts a continuous value or property, Node clustering groups nodes into clusters based on the attribute, and node embedding relates to learning a low-dimensional vector representation for each node.

- Edge level prediction
  Edges carry information about the relationship between nodes. Edge-level predictions are tasks such as edge classification and link prediction in which the model classifies an edge/relation type or predicts the occurrence of an edge between the nodes of the given graph. A neural network or similarity function is generally used to evaluate the presence or classification of an edge based on the encoded information of any two nodes.

- Graph level prediction
  In this task, the GNN is used to extract features of the entire graph data. The representation from the extraction is further used in tasks related to making predictions and classification of entire graphs using Aggregation, Graph Pooling, and Graph classification or Regression techniques.

  Regarding the type of graph data to be used in the task, the tasks are further represented in different types relating to the training environments:

- **Supervised setting** where data is labeled and available for training.

- **Semi-supervised setting** where a small number of nodes are labeled with many unlabeled nodes. The training phase requires the model to predict the labels of unlabeled nodes or provide new unlabeled nodes for inference, depending on the training requirement. It is mostly used in edge and node classification tasks.

- **Unsupervised setting** where only unlabeled data are provided to the model to extract features and patterns from the provided graph data. Tasks related to clustering of graph data are unsupervised tasks of graph data.

2.4.4 Graph Convolutional Networks (GCNs)

Graph Convolutional Networks (GCNs) are one of the most applied GNN architectures in applications based on graph data. It takes graph data as an input to make predictions based on full graphs or nodes. It is similar to a Convolutional Neural Network such that CNN uses a spatial cluster of pixels to represent a cluster of vectors whereas GCN uses a cluster of nodes and its surrounding nodes [10].

The key idea of GCN is to learn the information from input data using graph convolutions by using linear transformation on all neighboring nodes followed by a nonlinear activation function and then updating the feature representation of each node with information from all the nodes and edges. It collects feature information from all the nodes and their neighbors iteratively through broadcasting followed by aggregation. The aggregated information is initially transformed linearly which is then passed to a non-linear activation function such as ReLU to get non-linear representations [12]. The non-linear transformation is used to capture meaningful and comprehensive representations from the graph data. They are further used in the update function which consists of some information about the global representation of the graph [13].

The GCNs are differentiated by two distinct approaches: The Spatial approach and the Spectral approach. The spatial approach operates directly on nodes and edges of the given graph data and the Spectral approach utilizes graph theory and graph transformation in order to represent a graph. The spectral approach depends on the non-recursive architecture and borrows the idea of RGNNs for message passing [9].

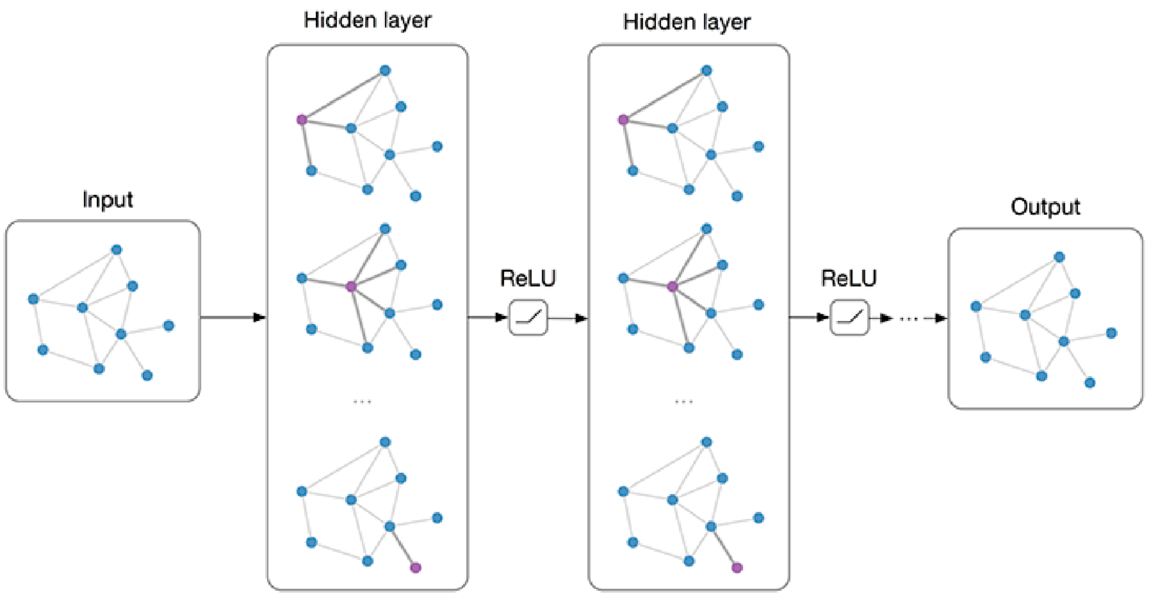


Figure 9: The Graph Convolutional Network architecture [12].

Mathematically, In the hidden layer, the convolution operation is denoted as:

$$GCN(X, A, W) = \sigma(D^{-1}AXW)$$

where X represents the node feature matrix, A is the adjacency matrix of the graph representing connectivity of nodes, D is the degree, W is the weight matrix, and σ is the activation function such as ReLU in the above figure. A degree matrix is a diagonal matrix which stores the degree of each vertex which numerically corresponds to the number of edges that the node is attached to.

In the given equation, the product $D^{-1}A$ represents an attempt to normalize the adjacency matrix. However, as matrix multiplication is non-commutative, an alternative symmetric normalization is preferred, changing the GCN layer's operation to:

$$GCN(X, A, W) = \sigma(D^{-1/2}AD^{-1/2}XW)$$

2.4.5 Directly Follows Graph

Process discovery is one of the crucial parts of process mining. It allows the discovery of process models using various tools. Some of the tools include Petri nets, BPMN models, and process trees that are able to construct complex maps but lack simplicity. For simpler business processes overgeneralization and simplicity are needed. One such tool used in generalized visualization of the process models in process mining is the Directly Follows Graph (DFG) [14].

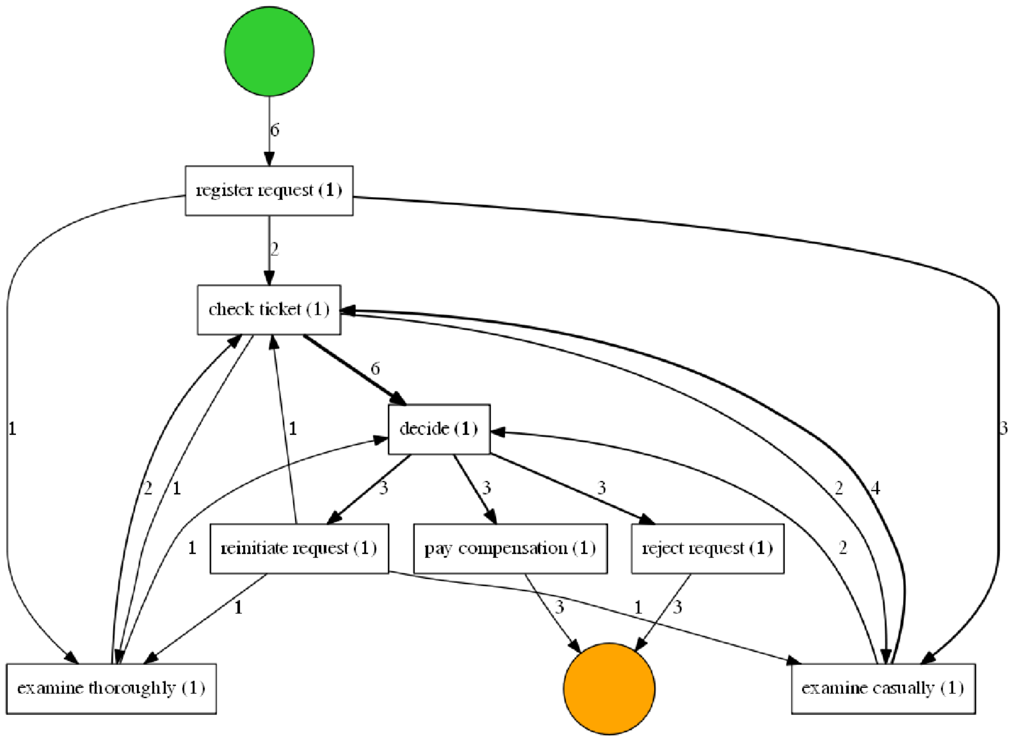


Figure 10: DFG of process related to refund requests. [self created

In Figure 10, The DFG captures the information about the relation between events in the event log. The "directly follows" operation between two events is shown if and only if there is a case instance in which the source event is followed by the target event. In the graph, the vertices represent the unique activities that are in the event log and the directed edges exist if there is a directly follows relation between the events/vertices. The number above the edges represents the "directly follows" relation denoted as weight.

2.4.5.1 PM4Py

PM4Py is an open-source Python library used in process discovery to analyze and generate graphs. It is compatible with most other Python libraries such as pandas, NumPy, SciPy, and sci-kit-learn and works with Comma-Separated Values (.csv), pandas data frame as well as XES format. XES is an XML-based standard format used for storing event logs and is primarily used in process mining. Some of the key features include:

3 Related Work

This chapter describes various types of implementations of deep learning methods in Business Process Management, Predictive Process Monitoring, Process Mining and further sheds light into applied approaches in GNN and LSTM for event and time prediction using literature reviews.

3.1 Business Process Management

Business Process Management (BPM) consists of a group of activities and techniques that manage and improve processes within an organization. It is considered a cycle of activities running in phases. In BPM, the cycle of activities contains process identification, discovery, improvement, monitoring, and controlling based on the data collected. Traditional data-driven approaches are considered as process discovery, analysis, and improvement whereas modern data-driven approaches are in lifecycle phases such as process monitoring [2]. In the latter, data are applied in real-time to forecast process behavior, performance, and outcomes. The significance of predictive process monitoring is ever-growing. It is driven by the need for organizations to mitigate performance issues and process efficiency [3].

Predictive process monitoring itself is a branch of BPM and Process Mining that leverages data-driven methodologies and predictive information to forecast future behavior and outcomes. Predicting remaining cycle time, sequence of process activities, process outcomes and prioritization of processes are part of predictive process monitoring that helps organizations adapt to evolving business goals, efficient flow of operational activities, and resource allocation [15].

3.2 Deep Learning in Predictive Process Monitoring

Deep learning is a neural network-based system that builds upon multiple layers of artificial neurons. The hierarchal structure of artificial neurons enables the layers to learn complex features and removes the need for the custom setting of feature parameters. In recent years, business process discovery methods have passed through a transformation because of an increasing number of deep learning approaches. This is led by the realization of organizations on the importance of process monitoring in run time [16]. Neural network architectures such as Graphical Convolutional Neural Networks (GCN), Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), and Long Short-Term Memory (LSTMs) have transpired as valuable tools in predictive process mining [12]. RNNs especially LSTMs have been applied to predict events, time of completion, and resources because of their ability to retain information on long sequences of events.

3.3 Deep Learning Methods in Process Mining

Deep learning methods have been increasingly used in process prediction tasks because of their ability to complete a wide range of tasks. The type of deep learning method depends on the use case such as the prediction of time-related attributes, activity-related attributes, or resource-based attributes which are explained in more detail in upcoming sections.

In terms of neural networks applied in the process, the common architectures can be represented by three different approaches which are Feed forward networks (FFNN), Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), LSTMs and GNNs [17]. The machine learning tasks in Process Mining could be categorized into multiple areas: Process Analysis, Anomaly Detection, Data Handling, User analysis, Resource Management, Clustering, and Classification [18].

Due to its simplicity and ability to extract related information from data with frequent occurrences of similar attributes, FFNN is mostly implemented in simple use cases and as a baseline metric. Nolle et al. implemented an FFNN for anomaly detection task that can distinguish between normal and anomalous traces of activities occurring in an event log. The focus is made on the ability of FFNN to detect anomalies early in the event log since the occurrence of anomaly is more distinguishable from more frequent traces [19].

Theis et al. [20] introduced a novel approach for predicting the next event. The implementation applies Petri Net process models in combination with FFNN for the prediction task. The output from the Petri Nets decay functions is used to train the model which can store time-related information from the event log to predict the next event. It shows its potential in applications related to various real-world events.

Di Mauro et al. [21] presented a deep neural network model based on convolutional neural networks for next activity prediction. They propose an inception architecture similar to computer vision that has been modified for sequential data use cases. They showcase the advantage of CNNs in sequential data in comparison to LSTMs in terms of both accuracy and computing efficiency.

Pasquadibisceglie et al. proposed a CNN approach to predict the completion of a trace. The end of a trace in the event data log is denoted by “END” which is the target variable to be predicted. They propose 2D image-like data structures to model trace data of event logs showing their ability to predict more accurately on frequent activities. The model shows that it can achieve better accuracy on the Helpdesk dataset compared to BPIC12 [22].

3.4 Usage of LSTMs in Process Discovery

The use of LSTMs in machine learning tasks is particularly in sequential data as it addresses the vanishing gradient problem. It is one of the most popular approaches taken in process prediction tasks [4] [17].

Tax et al. [23] approach implemented the LSTM approach with multi-task learning that can predict the next event and its time stamp. They proposed three types of architecture single-task layers, shared multitask layer and hybrid multitask layer. The architecture that was able to get better results was the multi-task layers, particularly hybrid multitask architecture. The multi-task and hybrid architecture differ in the initial LSTM layers that take both activity and timestamp information which are then passed to further individual layers. By using one hot encoding, the separate layers compute the prediction output for the next activity and timestamp separately. The models are able to predict the continuation of the case till the end and the remaining cycle time of the running cases. However, the LSTM model does not perform well when predicting longer sequences of event log.

Similarly, Khan et al. implemented a multi-task model for next activity prediction, time to event, and suffix prediction. The implementation focuses on a specific type of network known as Memory Augmented neural networks (MANN). A Differential Neural Computer (DNC) is implemented which is a type of MANN that uses a memory module. It comprises two controllers (separate LSTM layers) for encoding and decoding that are linked by a Memory Module. During encoding, the input sequence is fed into the encoder which is used by the memory module to update its information about the process states. The state of the encoder and memory combined are passed into the decoder controller which is used to predict by the decoding controller. The approach shows a competitive performance with fewer parameters compared to GRU, FFNN, and LSTM including the approach by Tax et al.

Evermann et al. proposed an LSTM network to predict the next event of a case from an event log. Unlike Tax et al., they take motivation from Natural Language Processing techniques for input rather than one hot encoding. In this approach, they take embeddings as input to complete the prediction task where the event logs are considered as text, traces as sentences, and events in a trace as words. The model consists of RNN architecture with a single hidden LSTM layer. In prediction, it faces the same issues as Tax et al. where the model predicts overly long sequences of the same activity when a longer sequence of event logs is given.

Lin et al. [24] implemented an RNN-based modulated model for multi-task prediction of event sequences. They implement LSTM networks for encoding decoding and a modulator that can combine information from attributes and encodings to output weight vectors. This weight vector is fed into the decoding LSTM layer to get the prediction of the next event where individual layers are used for each task. The model called MM-Pred achieves accurate predictions for the next event and its attributes showing results that it can effectively capture event information and its attributes.

Camargo et al. [25] introduced LSTM architecture with embedding vectors as input to predict event traces, timestamps, and role-associated event roles allowing iterative prediction of the entire trace. The approach includes pre-processing, LSTM model, and post-processing. The embedding technique uses n-grams with fixed size to scale and extract fixed-sized n-grams from each event log trace which is taken as an input. The resulting prediction through the LSTM layer is post-processed using a random selection of the next event based on LSTM probabilities. This post-processing technique makes use of arg-max and random selection.

Study	**Year**	**Model**	**Prediction**
Nolle et al.	2016	FFNN	Trace Class
Theis et al	2019	FFNN	Next Event
Di Mauro et al.	2019	CNN	Next Event
Pasquadibisceglie	2019	CNN	Next Event
Tax et al.	2017	LSTM	Next Event Time
Khan et al.	2018	LSTM	Next Event Time Trace class
Evermann et al.	2016	LSTM	Next Event
Lin et al.	2019	LSTM	Next Event
Camargo et al.	2019	LSTM	Next Event Time

Table 2: Comparison of Process Prediction methods.

3.5 GNNs in Process Mining Applications

“The Graph Neural Network Model” (GNN) capable of interpreting data in the graph domain was first introduced by Scarselli et al. in 2008. Since then, it has been widely used in several fields particularly demonstrating outstanding performance in vertex classification, link prediction, and recommender systems. The state-of-the-art GNNs can be categorized into different types: Convolutional GNNs (GCN), Recurrent GNNs (RCNN), Graph Autoencoders, and Spatial-temporal GNNs.

Sommers et al. [26] adopted the idea of using GCNs in supervised learning for process discovery. The implementation focused on the improvement of the supervised learning process when using event log data with neural networks. An ML-based approach that is able to translate event logs into Petri nets is improved using GCNs. The ML model trained on synthetically generated pairs of input event logs and output process models is improved using GCNs. After the implementation, the model is able to translate previously unseen synthetic and real-life event logs into structured models comparable to state-of-the-art techniques. The outcome showed improvement in generating process model using GCNs and its potential in enhancing process discovery techniques.

Similarly, Philipp et al. [27] presented the usage of GCNs to analyze the control flow graphs in process mining. The adjacency matrix is created using activity relationships of the entire event log. In terms of input, the adjacency matrix and additionally the feature matrix are used by combining a layer-wise propagation technique. The GCN model is compared to FFNN. The GCN outperforms the FFNN in the regression task.

Further, Weinzierl investigated the usage of Graph Sequence Neural Networks (GGNNs) in the next activity prediction. GGNNs are designed for sequential graphs and implement a gated recurrent unit (GRU). The author implements three different representation of event log as graphs and investigates the best-performing representation which is further compared to other DL techniques. The result shows that representing events as nodes and using a prefix-based adjacency matrix is able to achieve better results.

Maneiro et al. [13] developed a new approach named Recurrent Graph Convolutional Process Predictor that utilizes GCNs and RNNs to leverage both the structural information available in process models and the temporal information available in event logs. The petri net model is initially mined from an event log which is then converted and represented as an adjacency matrix similar to the approach in the GCN approach used in this thesis originally by Sorathiya [28] which is described in more detail in section 3.4. Additionally, the information from attributes of the event log is transformed into feature matrices which are fed into the LSTM network. Leveraging both structural and temporal information, the stacked GCN and LSTM combined model performs consistently well and achieves better predictive accuracy compared to Tax et al., Khan et al., Hinka et al., Camarago et al., and Evermann et al.

Venugopal et al. implemented four different variants of GNN, particularly GCN, and a Multi-Layer Perceptron (MLP) for predicting the next event and timestamp. They evaluated the performance at different process stages denoted by quartiles of the number of events similar to the approach by Sorathiya [28]. The MLP achieved better results in most cases whereas GCN with Laplacian Weighted adjacency matrix achieved the least MAE in time prediction. The performance is compared with state-of-the-art methods. However, the comparisons are found to be challenging because of differences in train-test split ratios and training procedures.

Many of the methods reviewed show different techniques to translate event logs into an adjacency matrix for the GNN with the help of a process model. The reviews give a better understanding of the use of process modeling to get comprehensive information from the event log and show how predictive tasks are performed with different combinations of process modeling and GNNs.

3.6 Related work on the KIGA dataset

The KIGA dataset contains an event log for the O2C process and the P2P process. An implementation of GCN by Sorathiya [28] in event and time prediction has previously been applied to the KIGA dataset which is thoroughly studied and used for comparison with multitask LSTM methods in this thesis.

Sorathiya [28] applied methods similar to Tax et al. [23] in data preprocessing and an approach similar to Venugopal et al. [12] in feature generation and GCN model implementation. The preprocessing used CaseID, ActivityName, and Timestamp data. Particularly, for vector representation of timestamp data, for each row of data, four variations of data were used: time since the case's most recent event, time since the case began, time since midnight, and the day of the week of the event. These four attributes are used for feature generation for each row of timestamp data. This method is further updated to include caseID information in the feature representation by a similar approach to Venugopal et al. [12].

For graph representation of event logs, an approach similar to Somaro et al and Manrairo et al is taken but instead of Petri nets, a more simpler and efficient process modeling technique, DFG is taken. The adjacency matrix is created using weighted values from the DFG and normalized then it is fed into the GCN. The model contained a GCN layer, 2 dropout layers, and 3 fully connected layers. An FFNN model is also implemented as a baseline metric to retrieve more information on the performance of GCN implementation. For measurement, accuracy was considered for event prediction and MAE (in days) for timestamp prediction. Furthermore, Classification metrics are used for further performance measures.

The result showed that MLP can achieve better overall accuracy in event prediction and lesser MAE in time prediction. The author makes notice of class imbalance in the event logs and suggests different procedures to handle it.

4 Concept

This chapter explains in detail about the dataset provided, data preprocessing techniques applied, feature encoding, model architecture and training procedure.

4.1.1 KIGA Dataset

The KIGA dataset is the dataset that is required to be experimented with and implemented in this thesis. It is extracted from real-life business organizations' event logs and contains event logs relating to both O2C and P2P. The following Table 3 shows the attributes and their values of the KIGA dataset.

Attribute	Order to Cash (O2C)	Purchase to Pay (P2P)
Unique cases	87,850	126,378
Unique events	20	22
Longest sequence of events	9	14
Events per case (mean)	3.692	4.332
Mean case duration (days)	180	98
Number of rows before preprocessing	509,257	1,633,071
Number of rows after preprocessing	324,364	547,542

Table 3 : Overview of KIGA dataset with their key attributes.

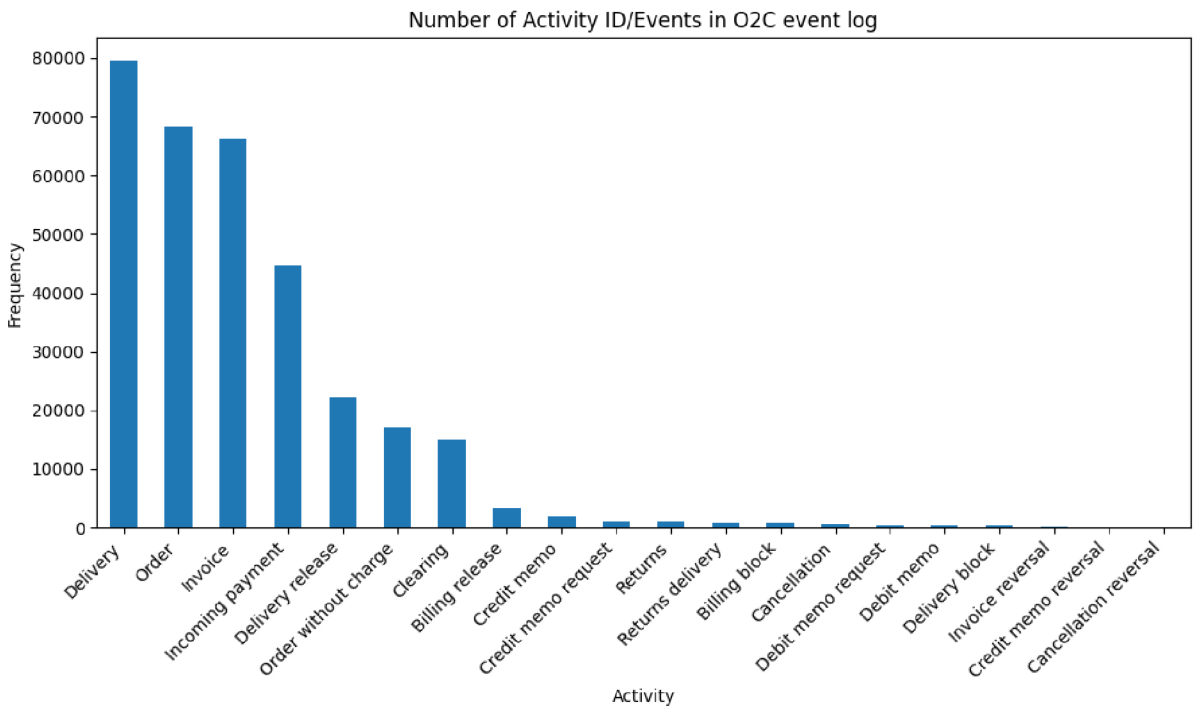
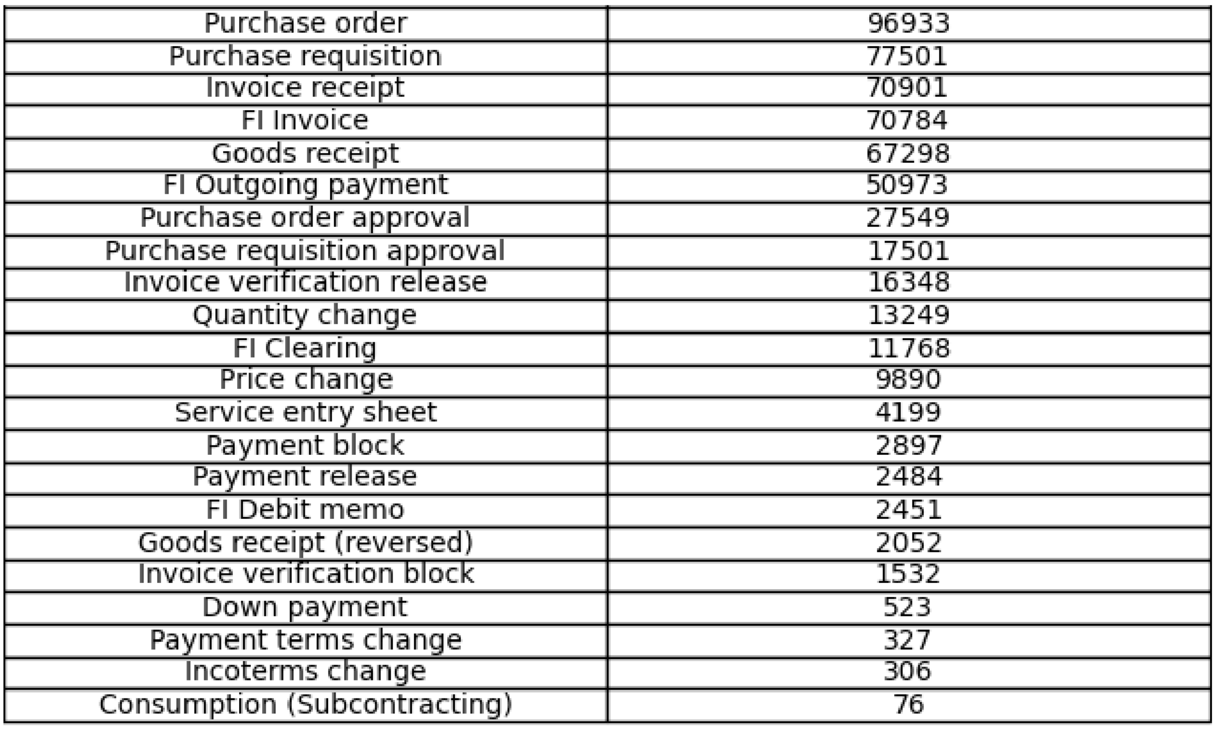


Figure 11: Bar plot showing unique activities and their quantity in the O2C event log.



Purchase order	96933
Purchase requisition	77501
Invoice receipt	70901
FI Invoice	70784
Goods receipt	67298
FI Outgoing payment	50973
Purchase order approval	27549
Purchase requisition approval	17501
Invoice verification release	16348
Quantity change	13249
FI Clearing	11768
Price change	9890
Service entry sheet	4199
Payment block	2897
Payment release	2484
FI Debit memo	2451
Goods receipt (reversed)	2052
Invoice verification block	1532
Down payment	523
Payment terms change	327
Incoterms change	306
Consumption (Subcontracting)	76

Table 4: Table showing activity and its frequency in the O2C dataset.

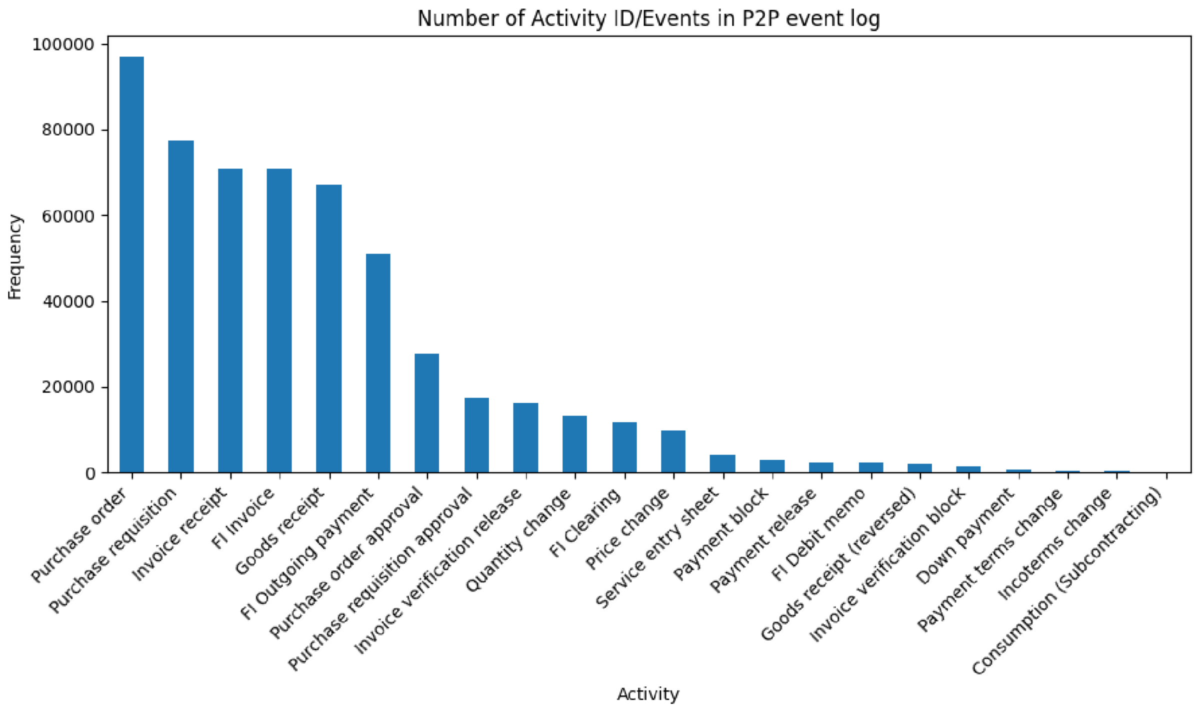
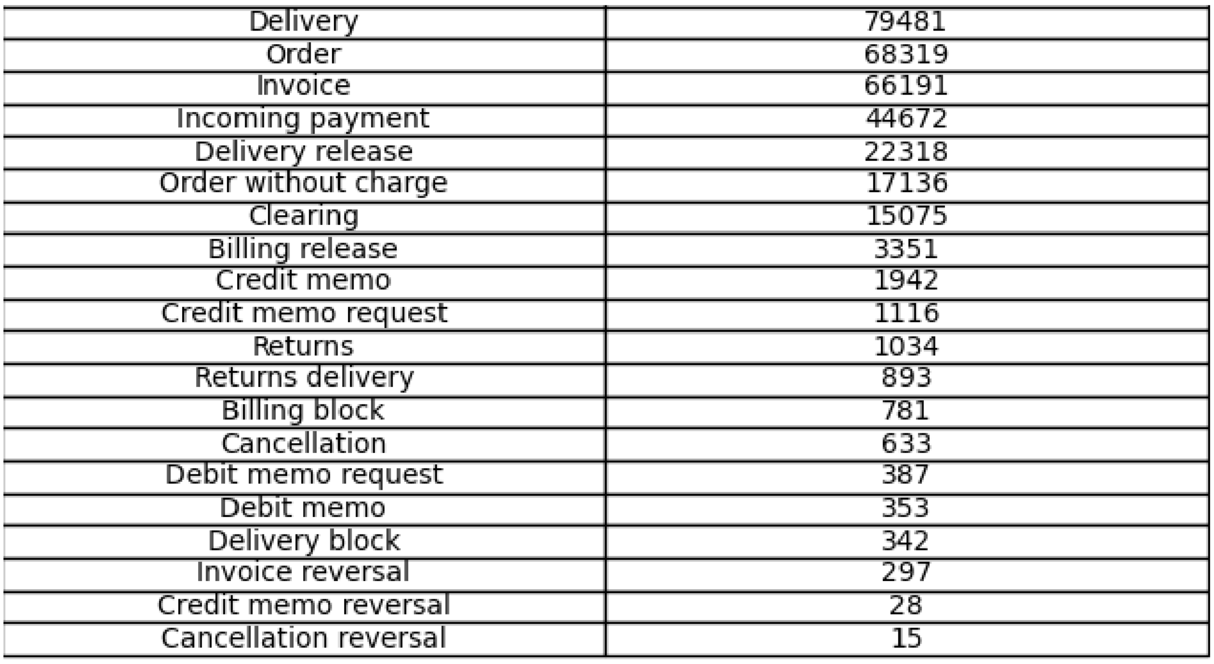


Figure 12: Bar plot showing unique activities and their quantity in the P2P event log.



Delivery	79481
Order	68319
Invoice	66191
Incoming payment	44672
Delivery release	22318
Order without charge	17136
Clearing	15075
Billing release	3351
Credit memo	1942
Credit memo request	1116
Returns	1034
Returns delivery	893
Billing block	781
Cancellation	633
Debit memo request	387
Debit memo	353
Delivery block	342
Invoice reversal	297
Credit memo reversal	28
Cancellation reversal	15

Table 5: Table showing activity and its frequency in the P2P dataset.

Figure 11 and 12 shows the distribution of unique events in the KIGA dataset and Table 4 and 5 gives quantity of those events. This provides some context in the distribution of the data and how it may affect model training during implementation.

4.1.2 Data Preprocessing

4.1.2.1 Data Cleaning

Data preprocessing and preparation is one of the fundamental parts of training a machine learning model. Business processes often contain raw event log data that are unprocessed and redundant. Generally, any case in a business process is expected to have a single instance of an event and may not contain the same event more than once. For example, in O2C when a customer makes an order, that case is expected to have a single event "Order Created".

In the KIGA dataset, in case there is a redundant activity logged a case, only the last activity and its corresponding timestamp are considered similar to Venugopal et al. [12]. Table 3 shows the number of rows before and after processing redundant activity for both datasets. A redundant number of the same event represents human error and inefficiency in business processes, which is not the case most of the time in an organization as they are closely monitored.

4.1.2.2 Capturing Temporal Dependencies

The time-related patterns in the process are extracted by capturing the temporal dependencies of each timestamp related to an event. The temporal dependencies are a set of time-based features computed from the timestamp data. Each activity in a row has time related feature of dimension 4.

Feature	Description
T1	Time since the previous event in the case.
T2	Time since the start of the case.
T3	Time since midnight (of the day)
T4	Day of the week for the event.

Table 6: Features for capturing timestamp information [12].

These features are used to capture time-related correlations within each particular case. It is used to effectively capture time-related information from the data in detail. The feature T1 describes the time difference between the current event and the previous event in a case. T2 describes the time difference between the current event and the first event in the case which is also the start of the case whereas T3 is the time difference since the start of that day, i.e., midnight and T4 denotes the day of the week.

4.2 GCN

4.2.1 Feature Encoding

The variant of the GCN model implementation consists of a Laplacian matrix of the Binary Adjacency matrix. The Binary Adjacency Matrix is implemented because of its simplicity and computing efficiency in processing information about nodes and edges. It helps to provide clear and concise representation of the graph. The Laplacian matrix provides information about the frequency of relation between nodes and edges which is explained in detail on Section 4.2.1.2.

Figure 13 below shows the model architecture for the GCN model. The adjacency matrix in the first layer corresponds to the Laplacian transformation of the Binary adjacency matrix of the dataset. The step-by-step method of generating an adjacency matrix and input vectors is explained below.

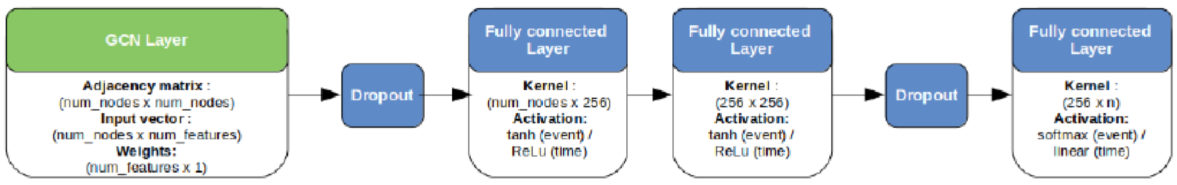

GCN Layer
Adjacency matrix : (num_nodes x num_nodes)
Input vector : (num_nodes x num_features)
Weights: (num_features x 1)
Dropout
Fully connected Layer
Kernel : (num_nodes x 256)
Activation: tanh (event) / ReLu (time)
Fully connected Layer
Kernel : (256 x 256)
Activation: tanh (event) / ReLu (time)
Dropout
Fully connected Layer
Kernel : (256 x n)
Activation: softmax (event) / linear (time)

Figure 13: Model architecture for the implemented variant of GCN. 'n' in the final layer represents the number of classes for event prediction and n = 1 for time prediction [28].

The Laplacian matrix has a dimension of ‘num_nodes x num_nodes’. The number of nodes denotes the number of unique activities in the ‘directly follows graph’ (DFG) of the given dataset. From Table 3, it can be deduced that the node value is 20 for the O2C dataset and 22 for the P2P dataset. The number of features corresponds to the temporal dependencies of the timestamp value for each row of data. Since temporal dependency has dimension 4, the num_features value is 4. This results in an input vector of size 20 x 4 for O2C and 22 x 4 for P2P. The weights are 4 X 1 for both datasets.

4.2.1.1 Generating DFG

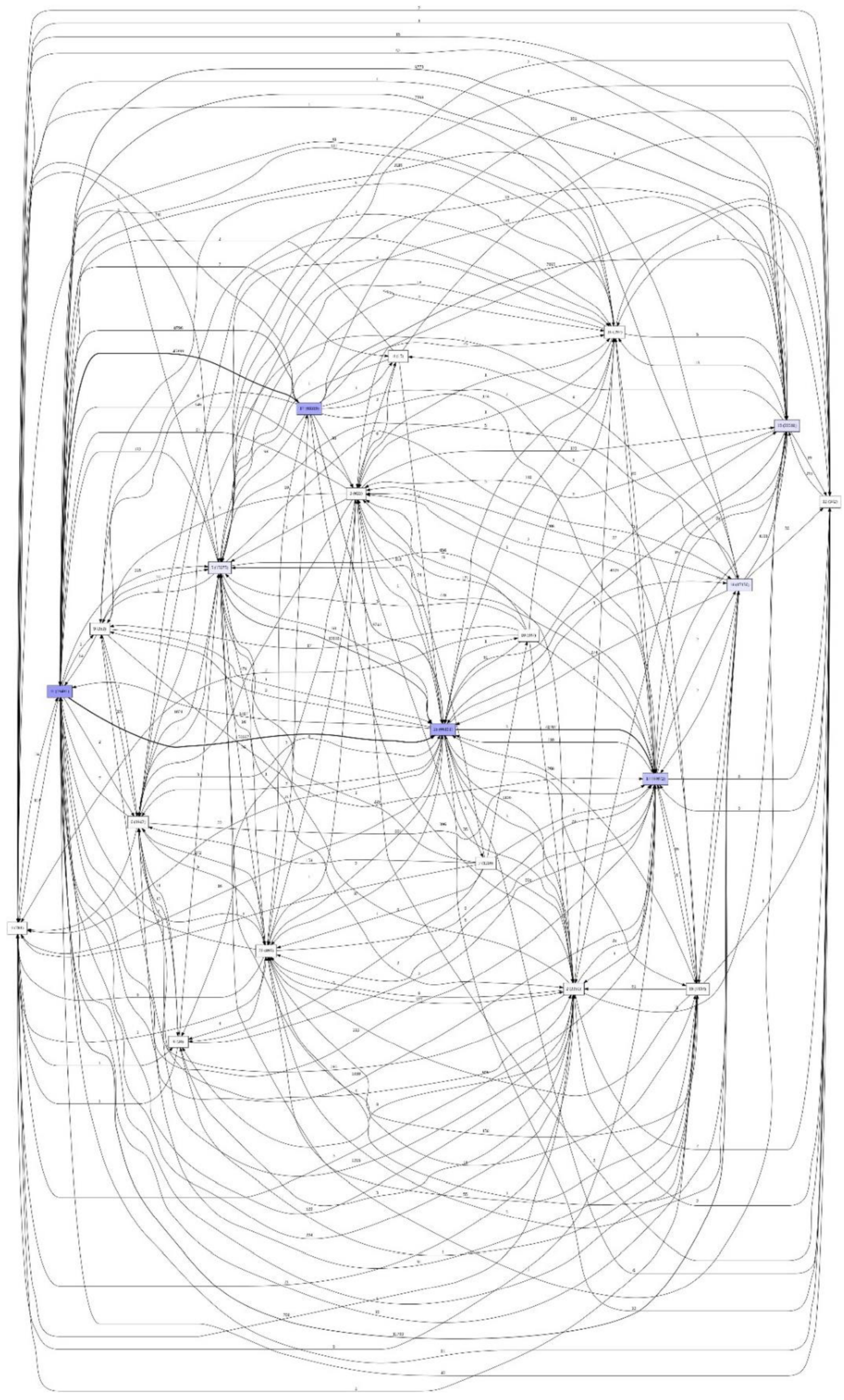


Figure 14: Directly Follows Graph visualization using PM4Py for O2C dataset.

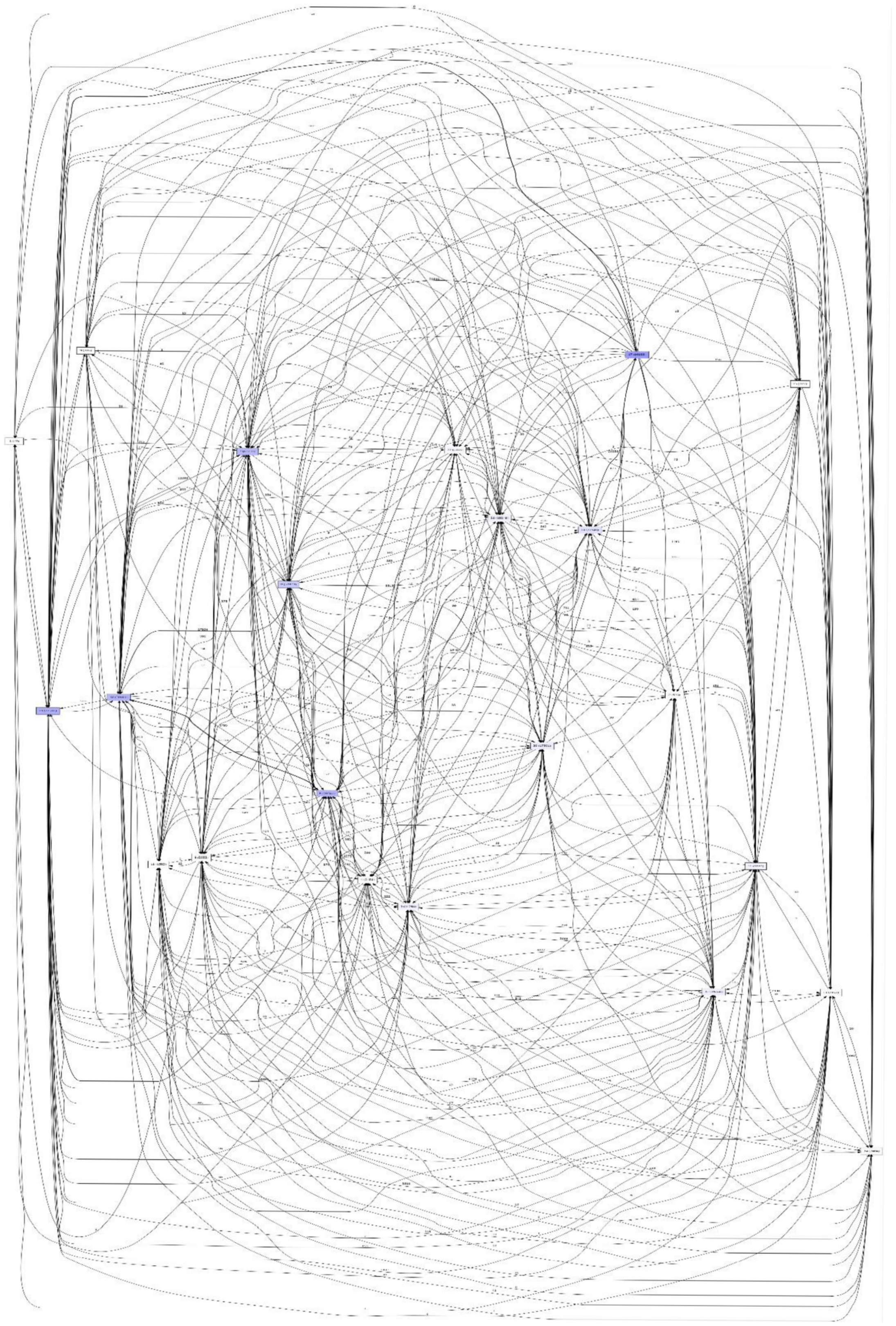


Figure 15: Directly Follows Graph visualization using PM4Py for the P2P dataset.

As mentioned in 2.3.5, the graph representation of the dataset and its corresponding weights and values are extracted using DFG because of its simplicity and performance. The PM4Py tool mentioned in 2.3.5.1 is used to generate the DFG and extract weights corresponding to the nodes. A Directly-Follows Graph for an Event Log L is denoted:

$$G(L) = \left(A_L, \mapsto L, A_{L^{start}}, A_{L^{end}}\right)$$

where A_L is the set of activities in L with $A_{L^{start}}$ and $A_{L^{end}}$ denoting the set of start and end activities, respectively. $\mapsto L$ denotes the directly follows operation.

Figures 14 and 15 above show the DFG visualization of the O2C and P2P datasets with their weights in relation to nodes. Here, the number of nodes / unique events is 20 for O2C and 22 for P2P. The visualization also shows the complexity of the case and activity relationship in both datasets.

4.2.1.2 Extracting Matrices from Process graph

The adjacency matrix represents the connecting nodes and the weight/frequency of the directly followed relationship between them. The frequency shows the number of times an event is followed by another event in the dataset. Initially, a raw weighted adjacency matrix of number of nodes x number of nodes with values corresponding to the edge value is extracted. This matrix is then represented as a binary adjacency matrix which converts all the values to either 0,1 where 0 represents no relationship between the nodes in the graph and 1 represents a relationship between nodes.

As described in Equation in Section 2.3.4, D is the diagonal matrix that stores the degree of each node which numerically corresponds to the number of edges that the node is attached to. A is the adjacency matrix extracted from DFG. However, the GCN variant implemented uses Laplacian transformation of adjacency matrix which requires the equation to be updated. The process is explained in detail in the following section.

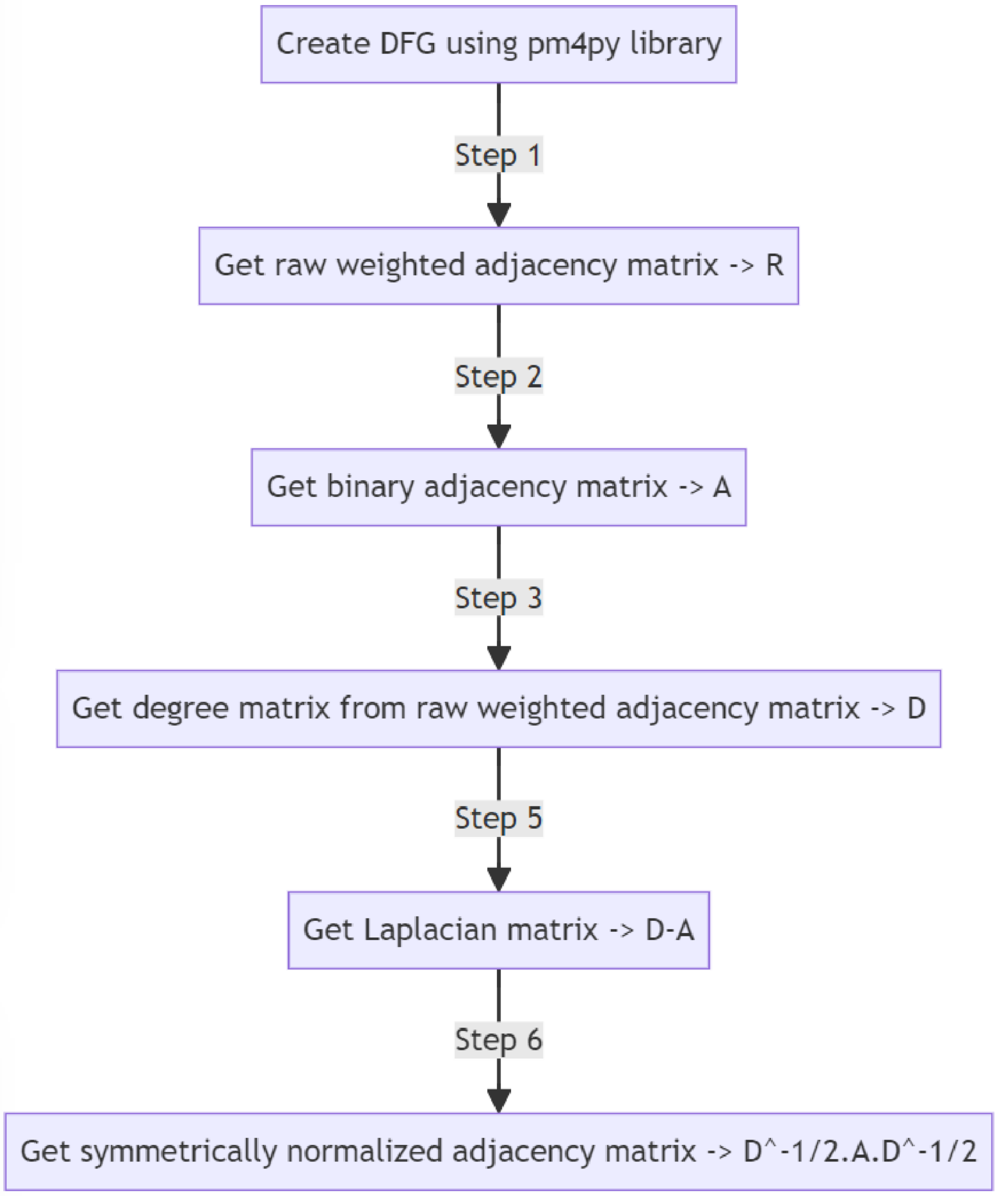

Create DFG using pm4py library
Step 1
Get raw weighted adjacency matrix -> R
Step 2
Get binary adjacency matrix -> A
Step 3
Get degree matrix from raw weighted adjacency matrix -> D
Step 5
Get Laplacian matrix -> D-A
Step 6
Get symmetrically normalized adjacency matrix -> D^-1/2.A.D^-1/2

Figure 16: Flowchart showing detail of calculating the Laplacian matrix of binary adjacency matrix that is implemented [12].

The above figure shows the flowchart of getting the final adjacency matrix for the GCN variant used in the experiment. The final adjacency matrix is known as the Laplacian transformation of the binary adjacency matrix of the given dataset. This matrix is used for all computations involved within the Graph Convolutional layer.

Therefore, the final GCN equation is then denoted by:

$$GCN(X, A, W) = \sigma(D^{-\frac{1}{2}}(D - A)D^{-1/2}XW)$$

Using the methods described in Figure 16, both O2C and P2P datasets are used to extract the necessary matrix for the GCN prediction task. The matrix below shows the Laplacian transformation of the Binary adjacency matrix of the O2C dataset.

$$\begin{pmatrix} 1.00 & -0.0925 & 0 & \dots & \dots & \dots & 0 & 0 & -0.0962 \\ -0.0925 & 1.00 & -0.0801 & \dots & \dots & \dots & 0 & 0 & -0.0801 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \ddots & \dots & \dots & \dots & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ -0.0962 & -0.0801 & -0.0833 & \dots & \dots & \dots & -0.0870 & 1.0000 & -0.0833 \\ -0.0962, & -0.0801 & -0.0833 & \dots & \dots & \dots & -0.0870 & -0.0833 & 1.0000 \end{pmatrix}$$

As detailed in Figure 13, The dimension of the matrix above is number of nodes x number of nodes which is 20 x 20. The matrix is then multiplied with input vector X which is a 20 x 4 represented as a number of nodes x 4 and weights W which is a 4 x 1 matrix. In case of the P2P dataset, the transformed adjacency matrix has a dimension of 22 x 22, input matrix X has 22 x 4, and W has a 22 x 1 dimension.

4.2.1.3 Input vector

For input vector X, the temporal dependencies in section 4.1.3.2 are taken into consideration to get complete information about an event. The number of nodes depends on the number of unique activities in a dataset. Each activity is assigned a unique number that corresponds to the row in the X i.e., 20x4. Each row has a 20x4 matrix which corresponds to the activity ID in the particular row. For example, if a row has activity ID 5 with timestamp data, the 20 x 4 input matrix stores information on the 5th row index whereas other rows have null values. This is necessary to facilitate the matrix multiplication in the GCN layer.

The activities are separated according to the Case ID assigned which enables the calculation of temporal dependencies that depend on the case. This method of representation gives each row in the O2C dataset a 20 x 4 matrix and each row in the P2P matrix a 22 x 4 matrix. The preprocessing procedure in section 4.1.3 ensures that each case does not contain redundant activities and considers only the latest executed activity.

4.2.1.4 Weight Matrix

The weight matrix W is the learnable parameters that are trained and updated during backpropagation and gradient descent while running the training procedure. It has the dimension of number of features x 1 i.e. 4 x 1.

4.2.2 Training Procedure

Using the encoding procedure described above, the traces of events can be transformed into an encoded sequence of events for each case ID. The encodings are then used to fit in the GCN layer in the model. The adjacency matrix captures the graphical information of the dataset and combines it with encoding to train the model. Furthermore, the method of extracting the adjacency matrix can be varied in order to apply the desired training procedure in terms of speed and memory. Figure 13 shows the complete architecture of the layers implemented for training. Particularly, the model sequentially includes an initial GCN layer and three fully connected layers. The Dropout is present between GCN and the first fully connected layer and before the last fully connected layer.

4.2.2.1 Split of dataset

The dataset is divided into three splits Train, Validation, and Test data. The training and validation data contain 2/3 of the dataset whereas the test data contain 1/3 of the dataset. Furthermore, the validation data contains 20% of 2/3 of the dataset. The sequential order of rows/events has been preserved during the splitting. This ensures the events are ordered as the traces of events for each case.

4.2.2.2 Event Prediction Model

The event prediction model follows the same architecture. The first two fully connected layers after the GCN layer use tanh activation for event prediction whereas the last fully connected layer uses the SoftMax activation function to estimate the probability for each target value as implemented by Sorathiya [28] in previous work. Cross Entropy loss is used during training to calculate the loss after the SoftMax activation. The update of weights is carried out using the rule of Adam optimizer through backpropagation and gradient descent.

4.2.2.3 Time Prediction Model

In the time prediction model, the first two fully connected layers use ReLU activation whereas the last fully connected layer uses linear activation function. The L1 loss function is used during training to obtain absolute loss value during training. The weights are then updated using the Adam optimizer through backpropagation and gradient descent.

4.3 LSTM

The type of LSTM applied is a multi-task vanilla LSTM that is able to take input vectors regarding both event and time and perform event prediction and time prediction through outputs from a single model.

The multi-task LSTM model implements the same preprocessing as GCN as explained in section 4.1.3 for comparison. The training, testing, and validation data are split in the same manner 2/3 train data and 1/3 test data. The features in LSTM also apply 4-dimensional features from temporal dependencies from timestamp values.

Encodings

The input vector X is a 3-dimensional tensor with the shape of several rows/samples x maximum length of event trace x num_features. The LSTM model requires sequential data to train. To create this, for each row in the dataset, the events corresponding to each case are sequentially added to the row for each time step. In case when the end of the case is reached, the next row in X starts with only the first event in the case. This results in the preservation of the sequence of events according to the case in the dataset and removes any loss of information regarding the case and its events when obtaining input data for the model. For example. Figure 15 below shows an illustration of how the sequence of events is transformed into input and target labels.

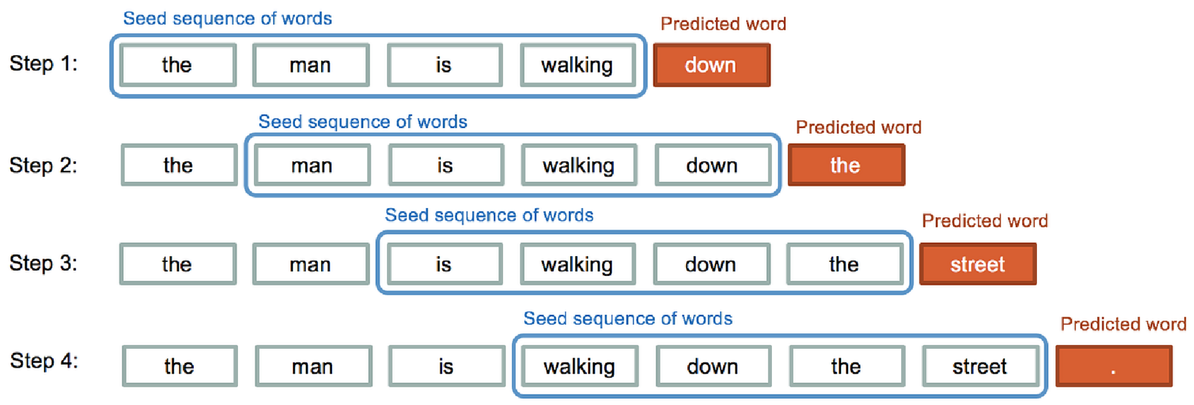


Figure 17: Figure showing extraction of a sequence of events and target/predicted values.

For each time step, the events and sequence of events are recorded with the target value as the next event in the sequence. For the next new case, the time step resets, and a new sequence of events is recorded.

For timestamp prediction, the target timestamp values are scaled by the average time between events which is a common practice in time-series analysis and predictive modeling. The scaling contributes to the normalization which helps the model learn about the pattern and relationship in time sequences and reduce bias. Therefore, for each prefix step, the target timestamp is calculated as the difference between the current and next timestamp/average time between events.

4.3.1 Model architecture and training

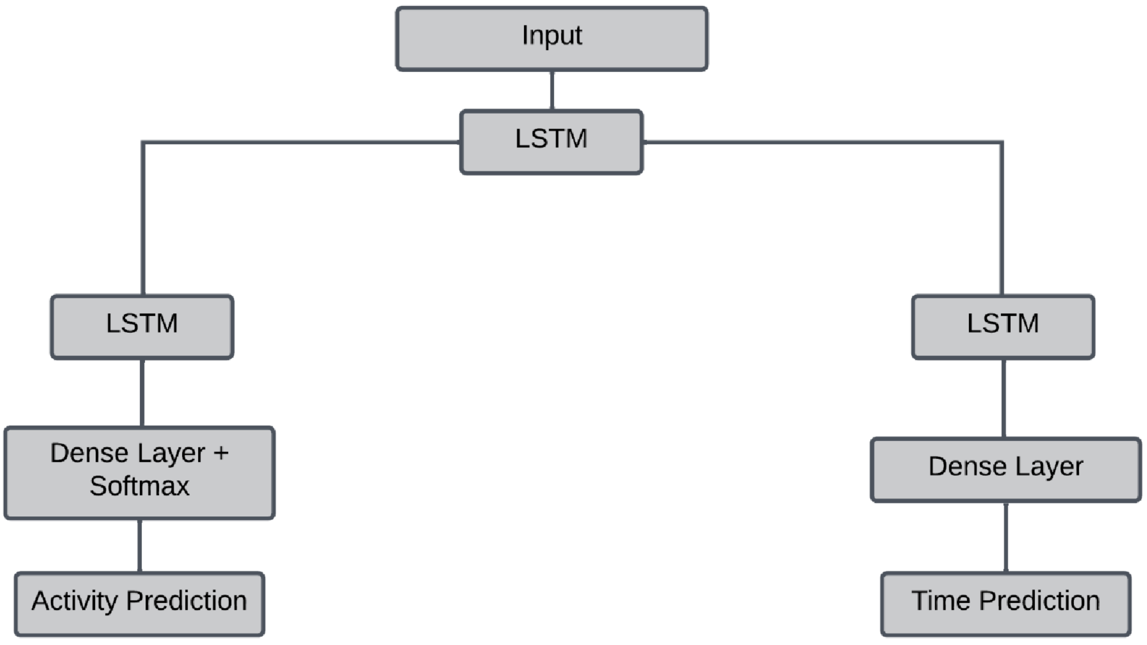


Figure 18: Architecture of multi-task LSTM.

As shown in Figure 18, The model architecture consists of a single input layer and three LSTM layers. The input layer of shape maxlen x num_features contains information about each row of the dataset which passes through the first LSTM layer. The first LSTM layer returns sequences with dropout and batch normalization. The second and third LSTM layers return single output features and are associated with event prediction and time prediction respectively.

For event prediction, SoftMax activation is implemented where the output dimension is equal to the number of unique activity labels. The output gives a probability of each of the unique activity labels and the highest probability number is selected as the predicted value. The softmax function normalizes a vector of real numbers into another vector of the same dimension, so that all components are in the interval [0,1], and the sum of all components is equal to 1. Hence, the transformed vector can be interpreted as a probability distribution while keeping the vector's original proportions.

The loss function used is the cross-entropy loss function that gives the difference between actual one-hot encoded values and prediction values. This difference is then used to update the weights using backpropagation and gradient descent.

For time prediction, the output is a single scalar output. The loss function is MAE where the actual values are compared with predicted values which is used to update the weights using backpropagation and gradient descent. The total loss during training is the sum of activity prediction loss and time prediction loss.

4.3.2 Multi-task training and advantages

Multi-task model and training are implemented because of the nature of the task to be performed and the property of input feature that contains information about both event and time stamp temporal dependencies. Although traditionally, a separate LSTM model for event and time prediction might be regarded as best practice, The input features and the dependency of event and time-related features in the dataset allow the implementation of a multi-task LSTM model. Some of the advantages are:

1. Joint learning of representation of both event and time as each event in the dataset is associated with the 4-dimensional feature of temporal dependencies. This allows the model to capture features relevant to both prediction tasks [12] .
2. Multi-task learning may help prevent overfitting because of the requirement to find and learn from common representation [29].
3. The parameters shared between the tasks might help the model to generalize better [30].
4. Less needed resources and computationally efficient as multiple tasks can be achieved using a single model and training procedure [30].
5. Deployment is made much easier because of a single model that can be used to perform both event and time prediction.

5 Implementation

This chapter gives information about the model hyperparameters used to get the best model for event and time prediction for both models and explains the hyperparameter tuning process in multi-task LSTM.

5.1 GCN

5.1.1 Event Prediction

The event prediction model for both O2C and P2P datasets was run with different learning rates to get the best value for each dataset. It is necessary to find the learning rate depending on the dataset used because, for the same model training, the dataset might have different learning rates that can converge during the learning phase. The learning rate ranging from 0.1 to 0.00001 was tested.

Hyperparameter	Values
Dropout Rate	0.0
Optimization algorithm	Adam
Learning Rate	0.00001, 0.0001, 0.001, 0.1
Number of Runs	10
Epochs	50
Batch size	32

Table 7: Table showing parameters in training event prediction model.

For the event prediction, an accuracy measure is considered. Accuracy is the ratio of correct prediction to the total number of predictions. The accuracy is given by:

$$Accuracy = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions}$$

During model training, the model with the least validation loss was saved while running for several epochs. The model training was run for 10 different runs of 10 epochs for each learning rate. Overfitting was limited by using the dropout value of 0.5.

5.1.2 Time Prediction

Similar to event prediction, the model training was run with different learning rates to get the optimum learning rate value resulting in the least validation loss. Similar ranges of learning rates were applied.

Hyperparameter	Values
Dropout Rate	0.0
Optimization algorithm	Adam
Learning Rate	0.00001, 0.0001, 0.001, 0.1
Number of Runs	10
Epochs	50
Batch size	32

Table 8: Table showing best hyperparameters in training time prediction model.

For time prediction, the mean absolute error (MAE) is applied to measure the difference between predicted time and actual time. It is measured in terms of days. The MAE is given by:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |x_i - x|$$

The model with the least validation loss was saved and used on the test set for evaluating the results.

5.2 LSTM

5.2.1 Hyperparameter Tuning

During training, hyperparameter tuning is implemented to get the best possible combination of hyperparameters for the model. The model with the lowest validation loss was chosen for the test set evaluation.

Hyperparameter	Values
Number of LSTM units	64, 100
Dropout Rate	0.0, 0.2
Optimization algorithm	Adam, Nadam
Learning Rate	0.0001, 0.0002, 0.001, 0.002, 0.01
Epochs	50
Batch size	maxlen value

Table 9: Hyperparameters implemented during multi-task LSTM model training.

Grid search using these parameter values was implemented in order to get the best model for O2C and P2P separately. The metric used to get the best model was validation loss. Being a multi-task model, the validation loss is the sum of both event prediction and time prediction losses in each epoch. The hyperparameter tuning allows the best model to be chosen with the least validation loss and be automatically saved. The saved model later is used to derive results on the test set.

Additionally, early stopping is implemented with 20 percent of 2/3 of the dataset as a validation set. The patience value is set as 25 which stops the learning process if there is no improvement in 25 consecutive epochs. The use of early stopping allows the regularization of the model to prevent overfitting and helps to prevent the model from learning noise in the training data

6 Evaluation and Discussion

This chapter provides detail about the parameters and procedure to get the best model, model evaluation and later provide insights about the results of event and time prediction for both models.

6.1 Model evaluation

6.1.1 GCN

Event prediction for both O2C and P2P was best when training with a learning rate of 0.0001 and Adam optimizer and the test set are evaluated on the model. For time prediction also both O2C and P2P were best when trained on a learning rate of 0.0001.

Hyperparameter	Values
Number of LSTM units	64
Dropout Rate	0.0
Optimization algorithm	Adam
Learning Rate	0.0001

Table 10: Table showing the hyperparameters resulting in best value for both metrics.

6.1.2 LSTM

For O2C, the multi-task layer performed best in terms of validation loss when trained using hyperparameter values as below:

Hyperparameter	Values
Number of LSTM units	64
Dropout Rate	0.0
Optimization algorithm	Nadam
Learning Rate	0.001

Table 11: Table showing the hyperparameters resulting in best value for both metrics.

Similarly for the P2P dataset, the best performance was achieved using the same hyperparameters as the O2C model. However, the learning rate of 0.002 resulted similar score in terms of validation loss whereas different units and dropout rates resulted in lower performance.

6.2 Next Activity and Time Prediction

First, the results are summarized in the table for both methods and later the results are evaluated and discussed using graphs. Furthermore, it explains the result of the methodology applies in the thesis.

6.2.1 GCN

The evaluation is carried out for different prefix quartiles of the length of events for each case in the test set. The quartiles are separated concerning the longest trace in the dataset.

Dataset	Accuracy for Event Prediction				
	Quartiles of Events				
	Q1	Q2	Q3	Q4	Overall
O2C	70.9	64.08	74.94	76.71	68.27
P2P	49.78	57.57	82.2	76.92	52.87

Table 12: Experimental results for event prediction on GCN model.

Dataset	MAE (days) for Time Prediction				
	Quartiles				
	Q1 3.5	Q2	Q3	Q4	Overall
O2C	81.43	16.67	4.643	4.66	51.97
P2P	26.63	9.05	2.70	2.65	20.88

Table 13: Experimental results for time prediction on GCN model.

This allows for the cases with a smaller number of events to be in lower quartiles whereas cases closer to the longest trace are in higher quartiles. Separating shorter traces, medium-length traces, and longer traces gives a better understanding of the predictive performance of models for short, medium, and long traces of events.

6.2.2 LSTM

Dataset	Accuracy for Event Prediction							
	Prefix No.							
	2	3	4	5	6	7	8	Overall
O2C	41.17	67.23	66.72	47.26	56.78	28.00	0	43.88
P2P	18.61	28.86	43.47	59.24	77.15	-	-	45.48

Table 14:Experimental results for event prediction on multi-task LSTM model.

Dataset		MAE (days) for Time Prediction						
		Prefix No.						
	2	3	4	5	6	7	8	Overall
O2C	44.71	40.22	40.57	35.49	35.56	18.21	14.67	32.78
P2P	46.01	58.16	70.84	54.87	30.24	-	-	52.02

Table 15:Experimental results for time prediction on LSTM model.

Similar to GCN model evaluation, for LSTM as well, the evaluation on the test set is performed on a different number of events denoted by prefix number, giving more information about the performance on varying lengths of traces. The prefix length can be associated with shorter, medium, and longer traces, and similarly, the performance is differentiated according to length. This also allows for comparison with GCN results as the quartiles are separated likewise.

6.2.3 Comparision and Key Findings

6.2.3.1 Event Prediction.

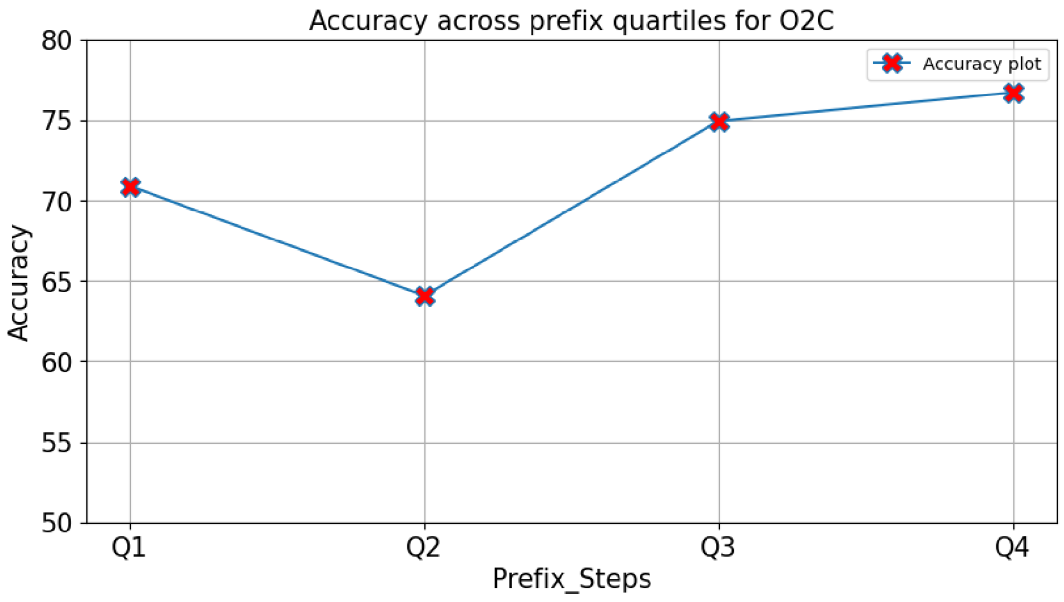


Figure 19: Q1 contains all prefixes <= 2.25, Q1< Q2 <=4.5, Q2< Q3 <= 6.75, Q3< Q4 <= 9 for event prediction of O2C on different lengths of prefix events.

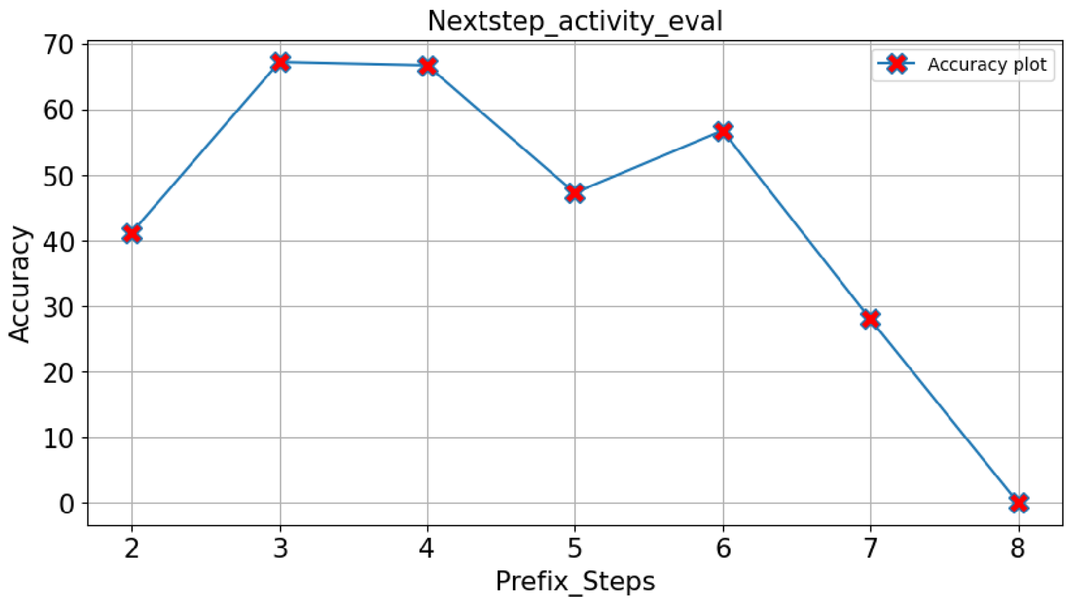


Figure 20: Plot showing performance of LSTM model for event prediction of O2C on different lengths of prefix events.

From Table 3, it can be known that the average number of events in both datasets is approximately 4. This information needs to be taken into context while analyzing the model performance. Therefore, the majority of cases lie between Q1 and Q2 for the GCN plot whereas the prefix number between 3 to 5 contains the majority of cases. All in all, it can be deduced the medium length contains and reflects results for the majority of data.

On the O2C dataset, the GCN model performs better than LSTM in terms of overall accuracy as well as for short, medium, and long traces of events. For short and medium traces, the accuracy is near 70 percent for LSTM whereas it is between 70.9-74,94 percent for GCN as seen in Table 11. It can be observed that both models can predict better on short and medium length of traces and accuracy declines as traces get longer. The overall accuracy for GCN is 68.27 whereas for LSTM is 43.88.

The performance of GCN for short prefixes is 70.9 percent which is higher than LSTM model whereas for the medium number of prefixes, both models achieve similar accuracy. The longer prefixes from 5 to 8 are in less quantity where GCN achieves better increasing accuracy ranging from 74.94 to 76.71 percent. The LSTM model however does not perform better on longer traces shown by declining accuracy in the plot.

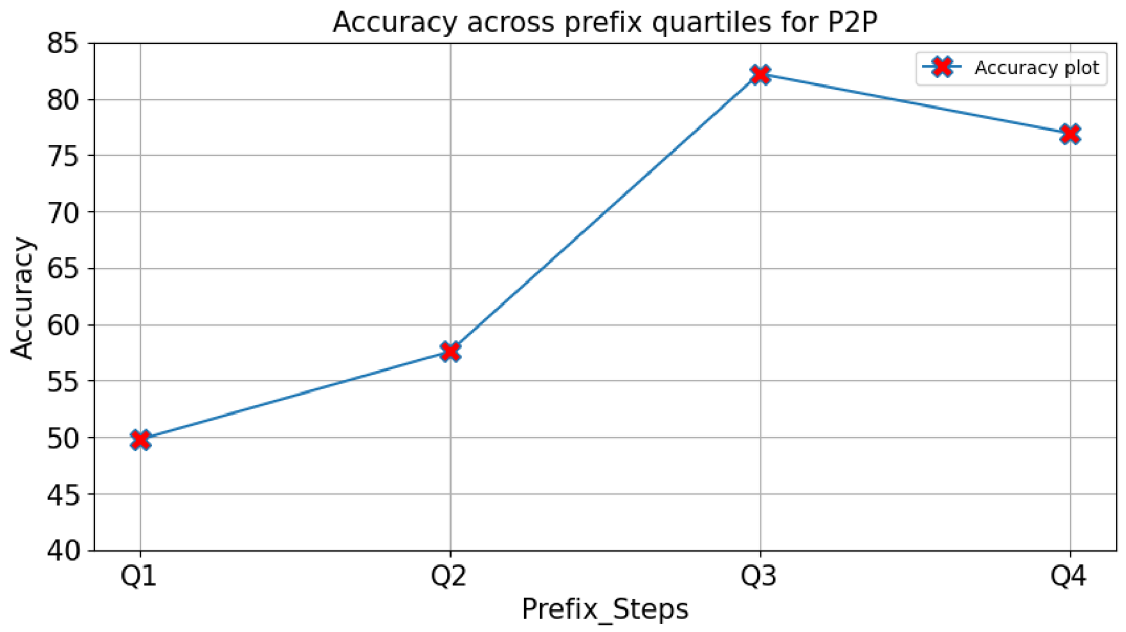


Figure 21:. Q1 contains all prefixes <= 3.5, Q1< Q2 <=7, Q2< Q3 <= 10.5, Q3< Q4 <= 14 for accuracy of event prediction of P2P.

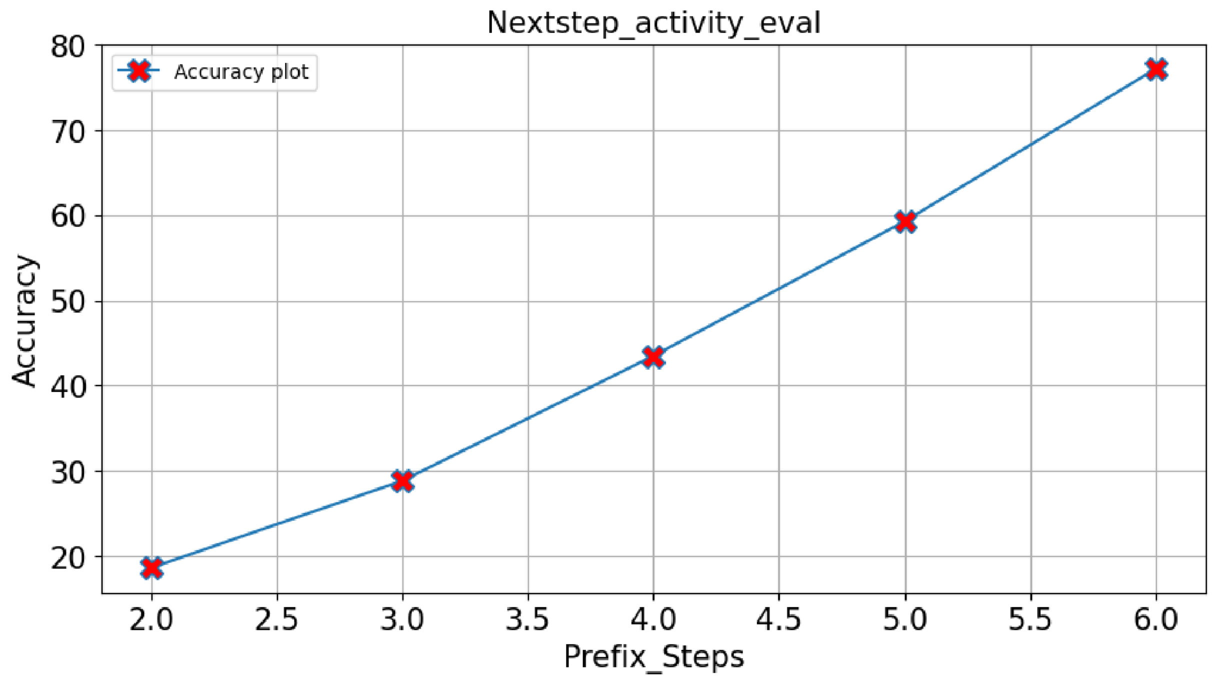


Figure 22: Plot showing performance of LSTM for event prediction of P2P dataset on different traces.

Given that average traces are medium-sized prefixes, similar to O2C, the majority of cases lie between Q1 and Q2 for GCN and 3 to 5 for LSTM plots in Fig. 21 and 22. However, the P2P dataset contains a larger quantity of data. For shorter prefixes, GCN achieves 48.78 percent accuracy whereas LSTM achieves 18.61 to 28.86 percent of accuracy. For medium size prefixes, the GCN achieves 49.78 to 57.57 percent accuracy and LSTM achieves 28.86 to 59.24 percent accuracy. This shows the performance is almost similar for both models for the majority of cases in the dataset. For longer prefixes, GCN performs better with 82 percent in Q3 and 76.82 percent in Q4. The LSTM for longer prefixes achieves 77 percent.

6.2.3.2 Time prediction

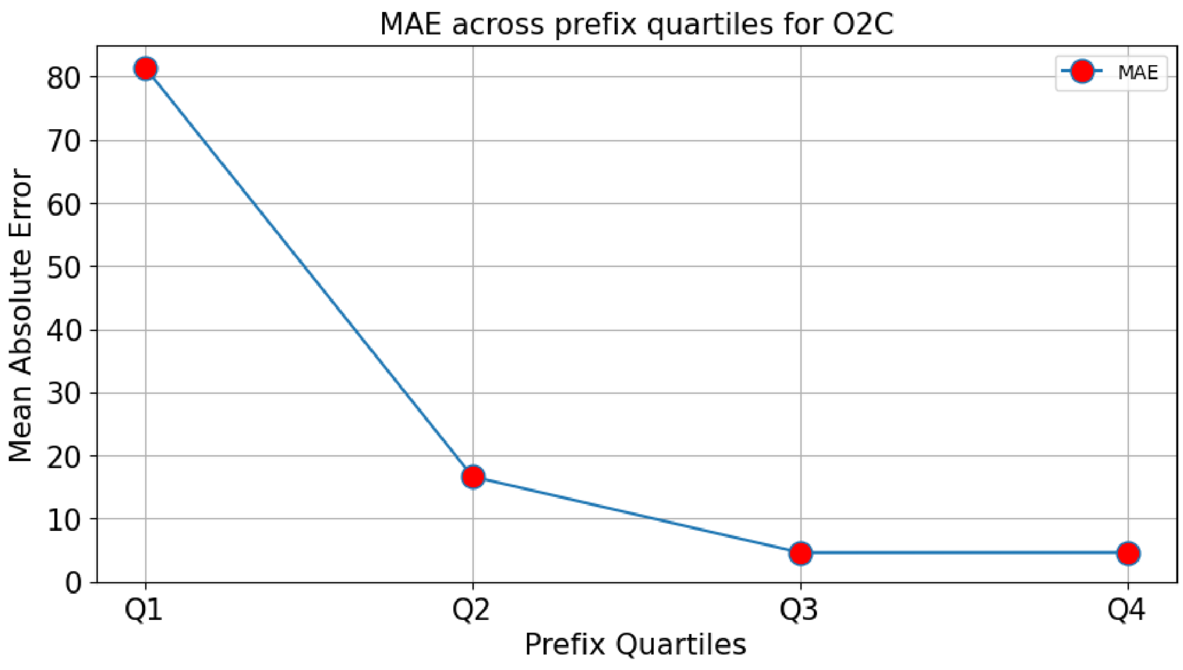


Figure 23: Q1 contains all prefixes <= 2.25, Q1< Q2 <=4.5, Q2< Q3 <= 6.75, Q3< Q4 <= 9 showing performance of GCN for time prediction of O2C dataset

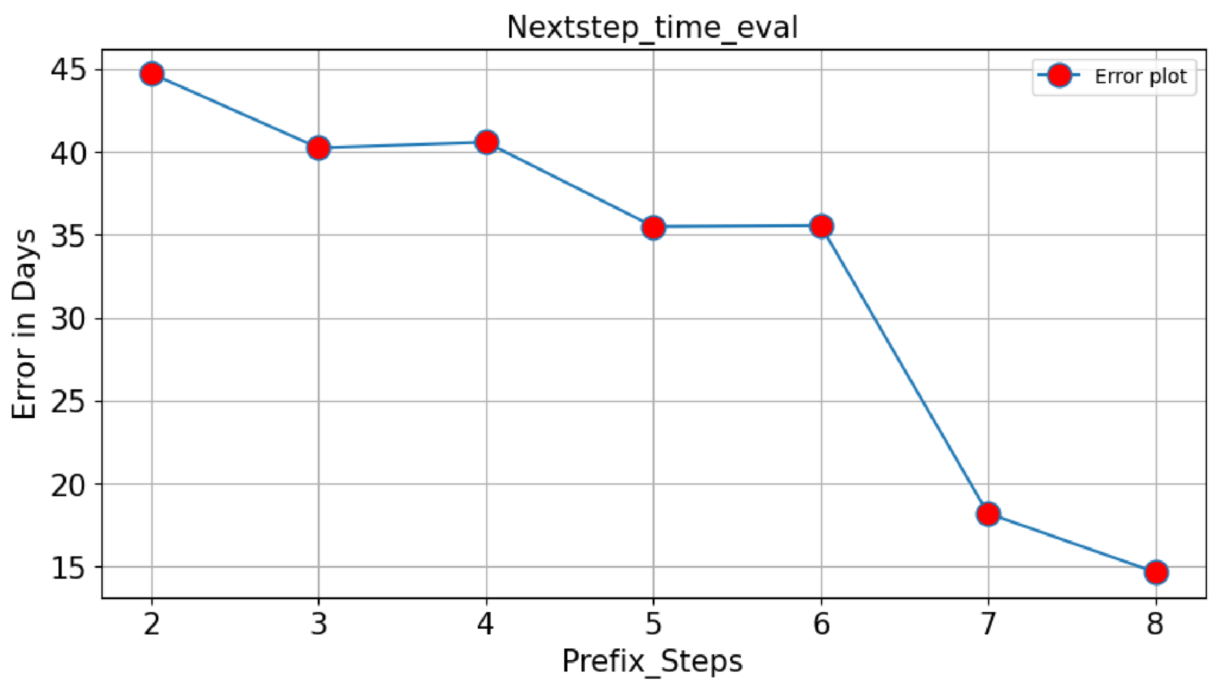


Figure 24: Plot showing performance of LSTM for time prediction of O2C dataset on different traces.

For the Time prediction of the O2C dataset, the LSTM achieves lower MAE for the majority of cases compared to GCN. GCN model has a higher MAE of 81 in Q1 with decreasing trend in Q2 and further quartiles whereas, in LSTM, the MAE is 46.01 to 30.24 for small, and medium-size prefixes and very low to less than 20 for longer prefixes. The drastic change in MAE for longer prefixes might have been influenced as a result of a smaller number of cases. Additionally, the overall MAE for the LSTM model is 33 days and 44 for GCN.

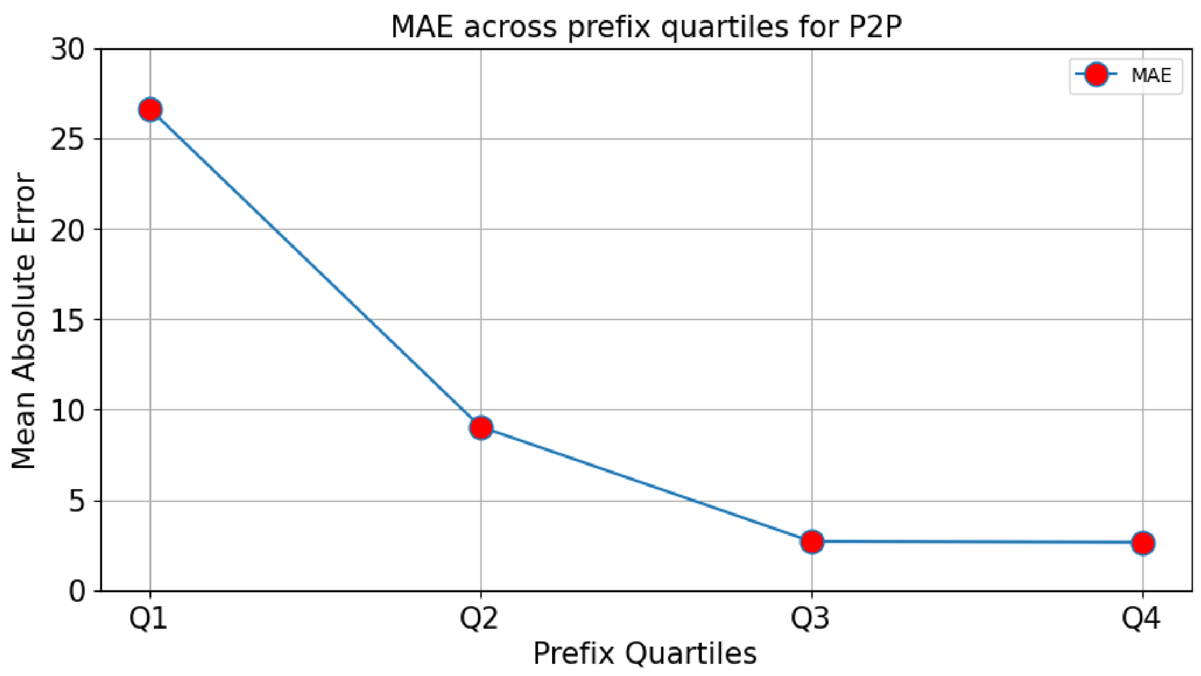


Figure 25: Plot showing performance of GCN for time prediction of P2P dataset on different traces. Q1 contains all prefixes <= 3.5, Q1< Q2 <=7, Q2< Q3 <= 10.5, Q3< Q4 <= 14.

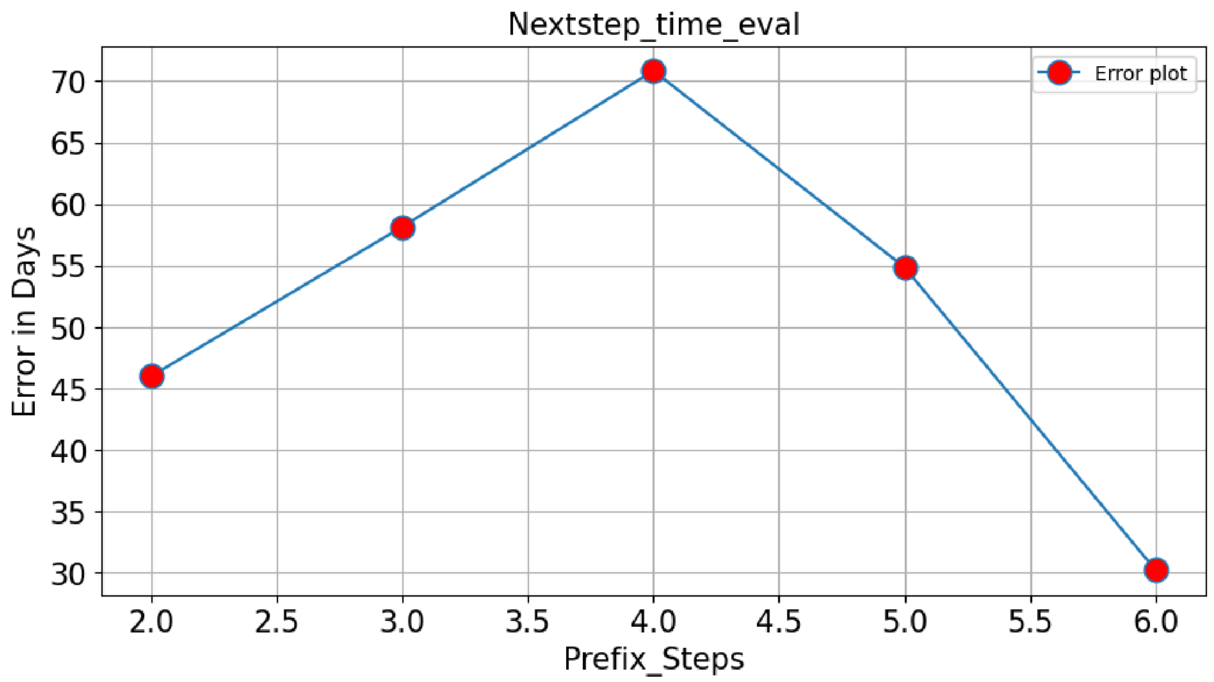


Figure 26: Plot showing performance of LSTM for time prediction of P2P dataset on different traces.

For the P2P dataset, the MAE is much lower for the GCN model across all prefix lengths compared to the LSTM model. The overall MAE values are 21 days for GCN and 52 days for LSTM showing better performance of GCN.

A comparison between two deep learning models is accomplished using similar approaches in terms of usage of dataset preprocessing, extraction of features, and separation of training, validation, and test set. Both models show viable results given the complexity of the dataset, whereby the GCN model shows better performance in event prediction for O2C and time prediction for P2P. The LSTM model shows better performance in O2C time prediction and similar performance overall in event prediction for P2P.

7 Conclusion

Process prediction is one of the major challenges in business process mining and predictive process monitoring. Timestamp data from ERP systems for O2C and P2P processes contain valuable information about the process flow and efficiency. The challenge of extracting information from real-life event logs using two different types of the deep neural networks is proposed in this thesis. Furthermore, this thesis provides a pathway for the KIGA project in further development of deep learning approaches in event and time prediction.

Given the overall performance, it is important to compare the performance relating to the prefix length. For consistent performance across different prefixes, the LSTM model can be considered especially for event prediction. Also, the type of process to be evaluated needs to be considered while choosing a preferred model.

7.1 Limitations and Outlook

In addition to viable results shown by both models, some limitations can be addressed to improve the approach and results. The event prefix imbalance is one of the factors that influence the training of the model. Although results for varying lengths of prefixes are implemented and compared, a dataset with more balanced prefixes could provide detailed insight into the performance of the model. These factors which could nonetheless be implemented in further iteration of research for the KIGA project.

Bibliography

[1] Procurement Strategy Blogs, "GEP, Intelligence drives innovation," 10 July 2023. [Online]. Available: https://www.gep.com/blog/strategy/order-to-cash-vs-procure-to-pay.

[2] W. M. P. van der Aalst and J. Carmona, Process Mining Handbook, Aachen: Springer Cham, 2022.

[3] A. Wil M. P. van der, Process Mining. Discovery, Conformance and Enhancement of Business Processes, Heidelberg: Springer Berlin, 2011.

[4] K. M. HANGA, A Deep Learning Approach to Business Process Mining, Birmingham City University, 2023.

[5] N. D. C. Lewis, Deep Time Series Forecasting with Python: An Intuitive Introduction to Deep Learning for Applied Time Series Modeling, CreateSpace Independent Publishing Platform, 2016.

[6] Y. Yu, X. Si, C. Hu and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures," *Neural Computation,* vol. 31, no. 7, pp. 1235-1270, 2019.

[7] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Comput.,* vol. IX, no. VIII, p. 1735–1780, 1997.

[8] S. Wu, F. Sun, W. Zhang, X. Xu and B. Cui, "Graph Neural Networks in Recommender Systems: A Survey," *ACM Computing Surveys,* vol. 55, no. 5, pp. 1-37, 2022.

[9] W. L. Hamilton, Graph Representation Learning, Springer Cham, 2020, pp. 1-159.

[10] N. A. Asif, Y. Sarker, R. K. Chakrabortty, M. J. Ryan, M. H. Ahamed, D. K. Saha, F. R. Badal, S. K. Das, M. F. Ali, S. I. Moyeen, M. R. Islam and Z. Tasneem, "Graph Neural Network: A Comprehensive Review on Non-Euclidean Space," *IEEE Access,* vol. 9, pp. 60588-60606, 2021.

[11] J. Zhou, G. Cui, S. Hu, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li and M. Sun, "Graph Neural Networks: {A} Review of Methods and Applications," *AI Open,* vol. 1, p. 57–81, 2020.

[12] I. Venugopal, J. Tollich, M. Fairbank and A. Scherp, "A Comparison of Deep-Learning Methods for Analysing and Predicting Business Processes," in *International Joint Conference on Neural Networks (IJCNN)*, 2021.

[13] E. Rama-Maneiro, J. C. Vidal and M. Lama, "Embedding Graph Convolutional Networks in Recurrent Neural Networks for Predictive Monitoring," *IEEE Transactions on Knowledge and Data Engineering,* pp. 1-16, 2023.

[14] S. J. Leemans, E. Poppe and M. T. Wynn, "Directly Follows-Based Process Mining: Exploration & a Case Study," in *2019 International Conference on Process Mining (ICPM)*, Aachen, 2019.

[15] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F. Maggi, A. Marrella, M. Mecella and A. Soo, "Automated discovery of process models from event logs: review and benchmark.," *IEEE Trans Knowl Data Eng,* p. 31(4):686–705, 2018.

[16] I. Verenich, M. Dumas, L. Rosa, M. Fabrizio and I. Teinemaa, "Survey and Cross-benchmark Comparison of Remaining Time Prediction Methods in Business Process Monitoring," in *ACM Transactions on Intelligent Systems and Technology 10(4):1-34*, 2019.

[17] N. Dominic A, L. Johannes and F. Peter, "A systematic literature review on state-of-the-art deep learning methods for process prediction," *Artificial Intelligence Review,* vol. 55, p. 801–827 , 2022.

[18] M. Pingel, "A systematic literature review on process mining and machine learning in healthcare," Enschede, 2021.

[19] T. Nolle, A. Seeliger and M. Mühlhäuser, "nsupervised Anomaly Detection in Noisy Business Process Event Logs Using Denoising Autoencoders," in *Discovery Science*, 2016.

[20] J. Theis and H. Darabi, "Decay Replay Mining to Predict Next Process Events," *IEEE Access,* vol. 7, p. 119787–119803, 2019.

[21] N. Di Mauro, A. Appice and T. M. A. Basile, "Activity Prediction of Business Process Instances with Inception CNN Models," in *AI*IA 2019 -- Advances in Artificial Intelligence*, Cham, 2019.

[22] V. Pasquadibisceglie, A. Appice, G. Castellano and D. Malerba, "Using Convolutional Neural Networks for Predictive Process Analytics," in *2019 International Conference on Process Mining (ICPM)*, 2019.

[23] N. Tax, I. Verenich, M. La Rosa and M. Dumas, "Predictive Business Process Monitoring with LSTM Neural Networks," *International Conference on Advanced Information Systems Engineering. Springer,* p. 1345–1365, 2017.

[24] L. Lin, L. Wen and J. Wang, "Mm-pred: A deep predictive model for multi-attribute event sequence.," *Proceedings of the 2019 SIAM international conference on data mining,* pp. 118-126, 2019.

[25] M. Camargo, M. Dumas and O. Gonzalez-Rojas, "Learning Accurate LSTM Models of Business Processes," in *Springer International Publishing*, Cham, 2019.

[26] D. Sommers, V. Menkovski and D. Fahland, "Process Discovery Using Graph Neural Networks," in *2021 3rd International Conference on Process Mining (ICPM)*, 2021.

[27] P. a. M. G. R. X. {Philipp, J. Beyerer, S. Robert and J. Beyerer, "Analysis of Control Flow Graphs Using Graph Convolutional Neural Networks," in *2019 6th International Conference on Soft Computing & Machine Intelligence (ISCMI)*, 2019.

[28] A. Sorathiya, "Implementing Graph based Neural Network models for a process mining application," Deggendorf, 2023.

[29] A. Joshi, S. Karimi, C. Paris and C. R. MacIntyre, "Does Multi-Task Learning Always Help? An Evaluation on Health Informatics Tasks," in *Proceedings of the The 17th Annual Workshop of the Australasian Language Technology Association*, Sydney, 2019.

[30] M. Crawshaw, "Multi-Task Learning with Deep Neural Networks: A Survey," 2020.

[31] G. Uwe and M. Khaled, "SAP News Center," 02 June 2021. [Online]. Available: https://news.sap.com/2021/06/sapphire-now-rise-with-sap-for-modular-cloud-erp/.

[32] K. Wolfgang, M. Jonas, R. Maximilian and S. Johannes, "Machine Learning in Business Process Monitoring: A Comparison," *Bus Inf Syst Eng,* p. 261–276, 2020.

[33] W. Kratsch, J. Manderscheid, M. Röglinger and J. Seyfried, "Machine learning in business process monitoring," *Bus Inf Syst Eng.,* 2020.

[34] A. Khan, H. Le, K. Do, T. Tran, A. Ghose, H. Dam and R. Sindhgatta, "Memory-augmented neural networks for predictive process analysis," *arXiv preprint arXiv:180200938,* 2018.

[35] A. Metzger and A. Neubauer, "Considering non-sequential control flows for process prediction with recurrent neural networks.," *Bures T, Angelis L (eds) 44th Euromicro Conference on Software Engineering and Advanced Applications (SEAA 2018).,* p. 268–272, 2018.

[36] W. M. Van der Aalst, M. Pesic and M. Song, "Beyond Process Mining: From the Past to Present and Future.," *International conference on advanced information systems engineering,* pp. 38-52, 2010.

[37] Y. Belghaddar, N. Chahinian, A. Seriai, A. Begdouri, R. Abdou and C. Delenne, "Graph Convolutional Networks: Application to Database Completion of Wastewater Networks," *Water,* vol. 13, no. 12, 2021.