Project Report

On

Process Discovery & Event Prediction using Deep Learning

A CNN-based Approach to Log Analysis

Under the guidance of

Dr. Joerg Evermann,

Associate Professor of Information Systems,

Faculty of Business Administration,

Memorial University of Newfoundland.

By

Siddhant Dani,

202385596,

Master of Data Science,

Memorial University of Newfoundland.

Index

Abstract

This project explores the fusion of Convolutional Neural Networks (CNNs) with process mining, a novel approach that enhances predictive accuracy and enables deeper analysis of business workflows. Using event logs from the Process Discovery Contest 2023, we set out to create CNN models capable of predicting the next event in a sequence of business activities. Our process was not limited to model building; it involved meticulous data cleaning, model training, and a rigorous evaluation phase. The results were promising, with an average training accuracy of 70.65% across 96 models. The true test, however, was the performance of these models on new, unseen data. They did not disappoint, demonstrating a precision of 0.9275, recall at 0.9251, and an F1-score of 0.9183, thereby proving their reliability and adaptability to various business scenarios.

Of course, the journey wasn't without its bumps. We faced challenges like lengthy training times and the initial struggle to predict probabilities accurately. Handling sequence termination tokens was another hurdle. But we didn't let these challenges slow us down. Instead, we found creative solutions—optimizing the code and tweaking our methodologies—which significantly improved the models' efficiency and accuracy. The result? This contribution adds to the growing field of process mining and machine learning and offers practical tools for real-world business process optimization.

1. Introduction

The complexity of modern business processes, coupled with the exponential growth of digital trace data, has positioned process mining as a crucial discipline for organizations striving to optimize their operations. By bridging the gap between traditional data mining and business process management, process mining offers tools that uncover, monitor, and enhance real-world processes by analyzing event log data. The ultimate goal is to extract actionable insights that drive better decision-making, improve performance, and ensure compliance.

In recent years, integrating deep learning techniques into process mining has garnered significant attention, especially in event log analysis and predictive process monitoring. Techniques such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks, renowned for their effectiveness in handling sequence data, have been employed to predict upcoming events, classify event sequences, and uncover underlying process models. This synergy between deep learning and process mining opens up promising avenues for enhancing the accuracy and efficiency of process analysis tasks.

However, applying deep learning to process mining has its challenges. Key issues include

- managing the variability inherent in event logs,
- addressing incomplete or noisy data and
- ensuring that models generalize effectively across diverse processes and datasets.

Moreover, interpreting deep learning models within the context of process mining remains a complex endeavour, often necessitating a hybrid approach that combines traditional process mining techniques with advanced machine learning models.

This project seeks to overcome some of the challenges in applying deep learning to process mining by developing an innovative approach that leverages CNNs and advanced classification techniques for process discovery and event prediction. The proposed methodology is not just innovative, but it is also rigorously evaluated using real-world event logs from the Process Discovery Contest 2023 and 2024, providing empirical evidence of its effectiveness. Additionally, the project explores various strategies for feature engineering, such as utilizing aggregated probabilities and custom kernels for classification, to further enhance model performance.

The motivation behind this project is to push the boundaries of what can be achieved through the fusion of process mining and deep learning. By integrating cutting-edge techniques, this project contributes to academic research and offers practical solutions that organizations can adopt to gain deeper insights into their processes. Ultimately, this leads to more informed decision-making and optimized process performance.

Through this project, we aim to demonstrate that when deep learning is thoughtfully integrated, it can significantly enhance the capabilities of process mining tools. The outcomes of this work are expected to not only contribute to the expanding body of knowledge in both process mining and machine learning but also set the stage for more sophisticated and effective process analysis tools in the future, inspiring the audience with the potential of this integration.

2. Prior work

In the domain of predictive process monitoring, deep learning models such as Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks have shown remarkable potential. These models are particularly effective in predicting process outcomes and handling sequential data. However, their application in real-world scenarios presents unique challenges that must be addressed to fully utilize their capabilities.

A prominent theme in the literature is the comparison between CNNs and LSTMs in predictive monitoring. For example, Nicola Di Mauro, Annalisa Appice, and Teresa M. A. Basile [1] explore the use of Inception CNN models for predicting the next activity in business process instances. They argue that while LSTMs have traditionally dominated sequential modeling, Inception CNNs offer a compelling alternative. By employing multiple convolutions with different kernel sizes, Inception CNNs capture both local and global sequence patterns, leading to superior prediction accuracy and computational efficiency. This makes Inception CNNs particularly suitable for real-time applications in process mining.

Building on this, Hans Weytjens and Jochen De Weerdt [2] critically examine CNNs and LSTMs in their study "Process Outcome Prediction: CNN vs. LSTM (with Attention)." Their findings reveal that CNNs not only achieve similar accuracy to LSTMs but also significantly outperform them in computational speed, making CNNs the preferred choice for real-time scenarios. Moreover, the study highlights CNNs' robustness across various hyperparameters, making them ideal for large datasets like BPIC 2017

On the other hand, the study by Tax et al. [3], titled "Predictive Business Process Monitoring with LSTM Neural Networks," investigates the strengths of LSTMs. The authors demonstrate that LSTMs excel in predicting various business process aspects, including the next activity, case continuations, and remaining cycle time. However, challenges such as handling repeated activity traces were noted, suggesting further research to optimize LSTM performance in specific contexts.

Expanding the application of deep learning beyond business processes, Ashrafi et al. [4] propose an innovative model that combines process mining with deep learning to predict mortality in coronary artery disease (CAD) patients. Utilizing electronic health records (EHRs) and the Decay Replay Mining (DREAM) algorithm, their model generates time-sensitive data that enhances predictive accuracy when processed by a neural network. The study’s results highlight the model's effectiveness, achieving an AUC score of 0.871, though the authors call for additional research to address data availability and completeness.

Similarly, Pishgar et al. [5] present a model integrating process mining with deep learning to improve mortality prediction in hospitalized COVID-19 patients. Leveraging Timed State Samples (TSS) and the DREAM algorithm, this model captures the temporal dynamics of patient health data, significantly improving predictive accuracy. Their research demonstrates the model's superior performance, especially in real-time predictions within the first 72 hours of hospital admission, emphasizing the critical role of temporal information in healthcare predictions.

Returning to business processes, Venugopal et al. [6] in their paper "A Comparison of Deep-Learning Methods for Analyzing and Predicting Business Processes," evaluate various models, including Graph Neural Networks (GNNs), CNNs, and LSTMs. The study reveals that simpler models like Multi-Layer Perceptrons (MLPs) can sometimes outperform more complex ones, especially at different stages of process prediction. This suggests that careful model selection based on specific dataset characteristics is crucial for optimizing predictive accuracy.

Moreover, Mehdiyev, Fettke, and Evermann [7] introduce a multi-stage deep learning framework designed to predict the next event in business processes. This approach combines feature hashing,

stacked autoencoders, and deep feedforward neural networks to handle high-dimensional data effectively and improve prediction accuracy. Tested on real-life datasets, this framework outperforms existing methods, emphasizing the importance of data preprocessing and hyperparameter selection.

In another novel approach, Xia, Xing, Ye, and He [8] propose a deep learning framework integrating CNNs and LSTMs for predictive process monitoring. Their method converts event logs into image data, fusing features using CNNs before feeding them into an LSTM network for prediction. By addressing challenges like overfitting and sparse data, their framework demonstrates improved accuracy and training efficiency compared to traditional methods.

Advancing process mining further, Lu, Chen, and Poon [9] develop an LSTM-based model to repair missing activity labels in event logs. Their approach uses both prefix and suffix sequences, along with additional attributes, to predict and correct missing labels. The results show that this method consistently outperforms existing techniques, enhancing the quality of process models derived from incomplete event logs.

Similarly, Mehdiyev, Evermann, and Fettke [10] propose a multi-stage deep learning model for predicting business process events. This model utilizes n-grams, feature hashing, and stacked autoencoders to transform event log data into a predictive feature matrix. Their findings highlight the model's superior predictive accuracy, particularly in handling imbalanced datasets, and offer a robust framework for future business process management applications.

Exploring yet another approach, Al-Jebrni, Cai, and Jiang [11] introduce the use of one-dimensional CNNs (1D CNNs) for predicting the next event in business processes. They suggest that 1D CNNs provide computational efficiency and improved performance over traditional Recurrent Neural Networks (RNNs) and LSTMs. Their model treats event traces similarly to sentences in natural language processing, leading to higher accuracy, precision, and recall in real-world datasets.

Additionally, Park and Song [12] present a comprehensive method for predicting business process performance using deep neural networks (DNNs). Their approach includes constructing an annotated process model from event logs, generating a process representation matrix that captures spatial and temporal dependencies, and building prediction models using CNNs, LSTMs, and LRCNs. Through real-life case studies, they demonstrate superior accuracy in predicting both short-term and long-term process performance, highlighting the importance of spatiotemporal correlations in enhancing process execution.

In conclusion, Evermann, Rehse, and Fettke [13] explore the use of Recurrent Neural Networks (RNNs), specifically LSTMs, for predicting process behavior. Their innovative approach treats process event logs as sequences of words, improving prediction precision without relying on explicit process models. The study indicates that this method significantly outperforms traditional approaches, offering a powerful tool for business process management.

Finally, Kamala and Latha [14] discuss the integration of process mining with deep learning techniques to enhance the prediction of business process outcomes. Their study emphasizes the use of deep fuzzy neural networks to improve forecasting accuracy, particularly in complex industrial settings like chemical processes. The approach is highlighted for its ability to automate the discovery and enhancement of process models, making it a valuable tool for industrial optimization and process improvement.

In summary, the integration of CNNs and LSTMs within process mining and predictive monitoring continues to offer significant advancements while also presenting new challenges. As these studies demonstrate, the careful selection and application of deep learning models are essential for optimizing predictive accuracy and computational efficiency across various domains.

3. Methodology and Data Processing Framework

3.1 Dataset and Preprocessing

Dataset Overview

The dataset for this project comes from the Process Discovery Contest 2023 (PDC 2023), which poses a unique challenge with its complex and varied event logs. The dataset includes:

- **384 Training Logs:** Used to train the model, these logs contain sequences of activities that represent different instances of a business process.
- **96 Test Logs:** These are set aside to evaluate how well the trained model performs.
- **96 Base Logs:** Serving as a reference point, these logs are used to compare with the test logs.

Each event log is stored in the IEEE XES format, which is a standard in process mining for recording and managing event logs.

Understanding Event Logs

Event logs play a crucial role in process mining because they capture the detailed execution of business processes. Each log is made up of multiple traces, where each trace represents a sequence of activities (or events) within a single instance of a process. The key components of an event log include:

- Case ID: A unique identifier for a process instance, like an order number. (e.g. trace1)
- Activity: The specific task performed, such as "t1" or "t11."

For example, in an XML-based XES file, a trace might be structured as shown in Figure 1.

```
<?xml version="1.0" encoding="UTF-8" ?>
<!-- This file has been generated with the OpenXES library. It conforms -->
<!-- to the XML serialization of the XES standard for log storage and -->
<!-- management. -->
<!-- XES standard version: 1.0 -->
<!-- OpenXES library version: 1.0RC7 -->
<!-- OpenXES is available from http://www.openxes.org/ -->
<log xes.version="1.0" xes.features="nested-attributes" openxes.version="1.0RC7">
    <extension name="Concept" prefix="concept" uri="http://www.xes-standard.org/concept.xesext"/>
    <global scope="event">
        <string key="concept:name" value="__INVALID__"/>
    </global>
    <classifier name="Event Name" keys="concept:name"/>
    <string key="concept:name" value="pdc2023_000000 [1000|1|0]"/>
    <trace>
        <string key="concept:name" value="trace 1"/>
        <event>
            <string key="concept:name" value="t1"/>
        </event>
        <event>
            <string key="concept:name" value="t11"/>
        </event>
        <event>
            <string key="concept:name" value="t12"/>
        </event>
        <event>
            <string key="concept:name" value="t40"/>
        </event>
        <event>
            <string key="concept:name" value="t3"/>
        </event>
        <event>
            <string key="concept:name" value="t5"/>
        </event>
        <event>
            <string key="concept:name" value="t33"/>
        </event>
        <event>
            <string key="concept:name" value="t29"/>
        </event>
    </trace>
</log>
```

Figure 1: Event Log

Preprocessing Steps

Preprocessing event logs is an essential step to ensure the data is ready for use in the Convolutional Neural Network (CNN) and other machine learning models. The following steps were taken to clean, standardize, and format the data:

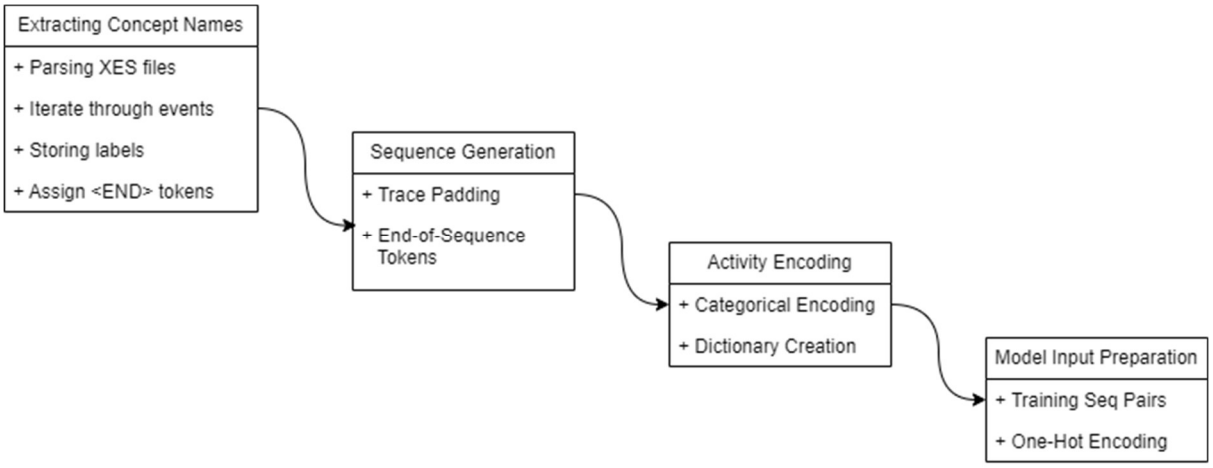

Figure 2: Preprocessing Steps

1. **Extracting Concept Names:**

- **Parsing XES Files:** The event logs were parsed using Python's xml.etree.ElementTree module to navigate through the XML structure.
- **Iterating Through Events:** Each trace and event were processed to extract the concept:name attribute, which represents the activity label.
- **Storing Activity Labels:** These labels were collected into sequences that reflect the order of activities in each trace.
- **End-of-Case Marker:** An <END> token was added to each sequence to mark the end of the process instance.

These sequences of concept names were then used as input for the model, helping it learn activity patterns and make predictions.

2. **Sequence Generation:**

- **Trace Padding:** Since the length of event sequences varies across traces, they were padded to a uniform length of 32 events. Pre-padding was applied to the beginning of each sequence to ensure consistency in the input data.
- **End-of-Sequence Tokens:** An <END> token was added to each trace to signal its completion, helping the model distinguish between ongoing and completed sequences.

3. **Activity Encoding:**

- **Categorical Encoding:** Each unique activity in the logs was converted into a numerical format through categorical encoding. This step involved mapping each activity name to a unique integer, which the CNN could then process as input.
- **Dictionary Mapping:** A dictionary was created to map each activity to its corresponding integer code. This mapping was crucial for both training the model and interpreting predictions during testing.

4. **Model Input Preparation:**

- **Training Sequence Pairs:** For training the CNN, sequences of activities were paired with their subsequent activity, creating input-output pairs. For instance, the sequence ["t1", "t11", "t12"] would be paired with the next activity ["t40"], allowing the model to learn the transition patterns between activities.
- **One-Hot Encoding of Labels:** The labels, which represent the next activity in a sequence, were one-hot encoded. This conversion turned the labels into a binary matrix, where each column represented a possible activity, and the correct activity was marked with a 1.

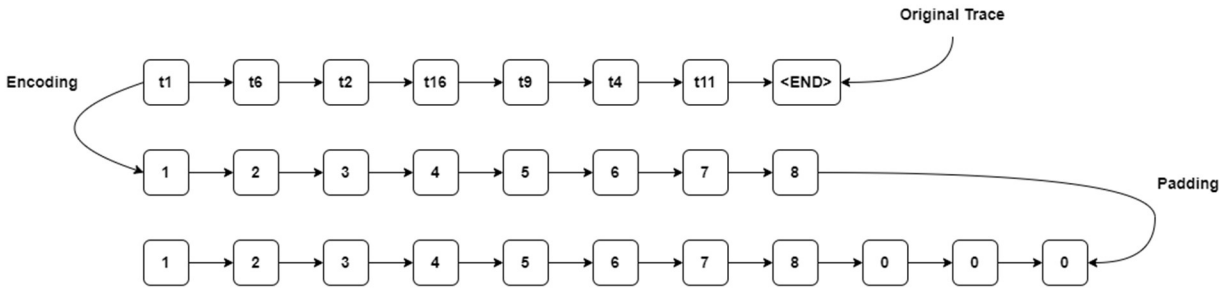

Figure 3: Encoding and Padding

These preprocessing steps were essential in transforming the raw event logs into a structured format that the CNN could efficiently process, enabling the model to learn and predict the sequences of activities in business processes.

3.2 Model Architecture and Training

In this project, Convolutional Neural Networks (CNNs) play a pivotal role in analysing and predicting sequences of activities within business process logs. What makes CNNs particularly well-suited for this task is their ability to detect intricate local patterns in data. This capability is crucial when dealing with event logs, where the order of activities is key to understanding the process. By identifying these patterns, the CNN effectively predicts the next step in a process, providing actionable insights.

One of the strengths of deep learning, and CNNs in particular, is the ability to learn complex patterns directly from the data, bypassing the need for manual feature extraction. This is especially valuable in the realm of process mining, where event logs can be extensive and intricate. With deep learning, we can automatically uncover the underlying patterns that dictate how processes unfold, enabling us to make more accurate predictions.

Model Architecture

The CNN model in this project is carefully designed to efficiently handle sequences of activities extracted from event logs. The architecture is both powerful and straightforward, comprising several key layers:

- Embedding Layer: This layer converts activity codes into dense vectors, which the model can easily process.
- Convolutional Layer: Here, the model detects patterns within the sequences by applying filters across the data, helping it to recognize important features.
- Pooling Layer: To streamline the data, this layer reduces its size, keeping only the most crucial features for further analysis.
- Dense Layers: These layers take the refined features and process them, ultimately producing a probability distribution over possible next activities.
- Output Layer: Finally, the model uses this information to predict the most likely next activity in the sequence.

This architecture strikes a balance between simplicity and effectiveness. It allows the model to achieve high accuracy in its predictions while remaining accessible to those who might not be deeply familiar with the complexities of deep learning.

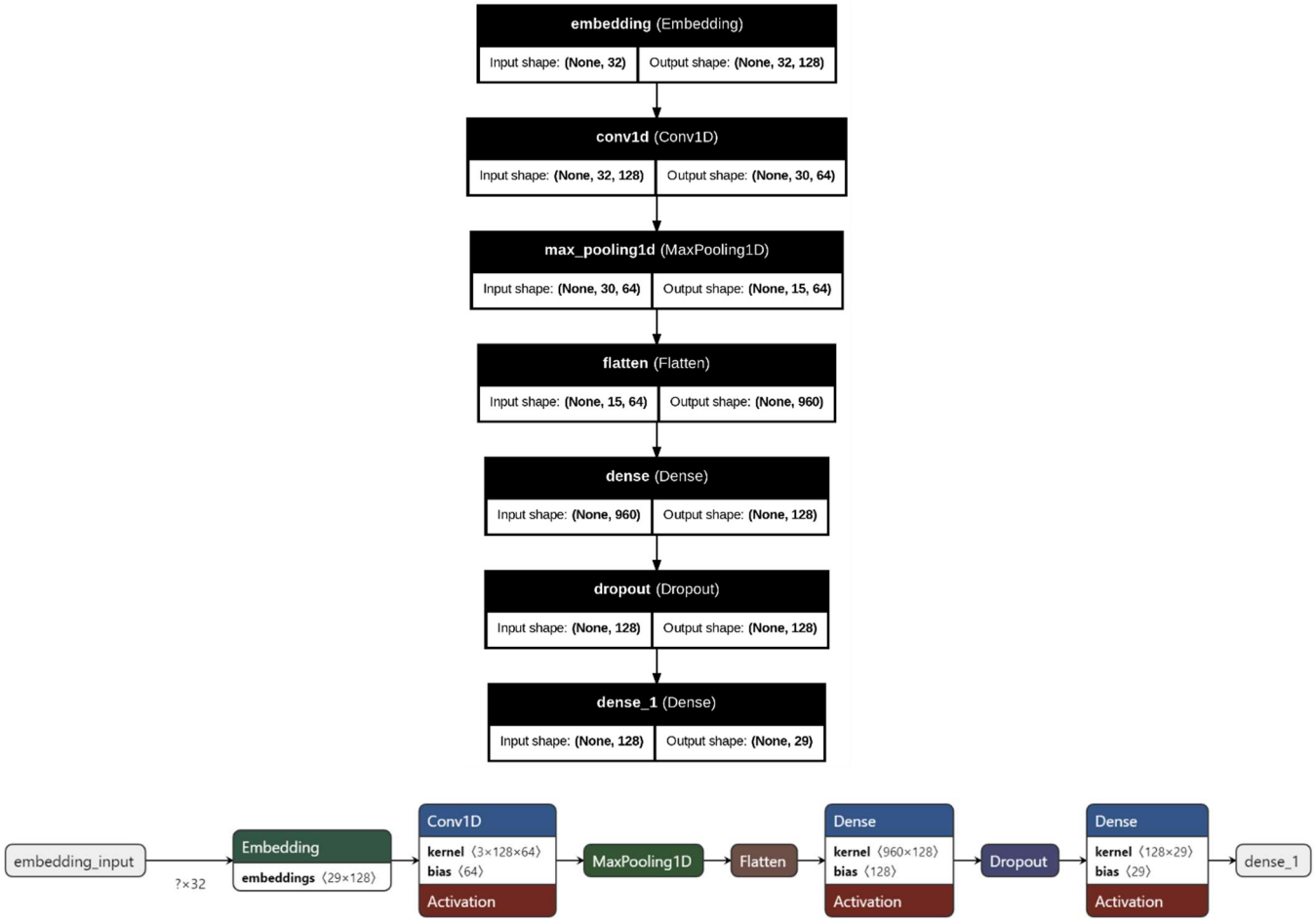

Figure 4: Proposed CNN Architecture

Training Process

The training process for the Convolutional Neural Network (CNN) model is meticulously structured to ensure thorough data preparation and optimal model performance. The following steps outline the procedure, detailing the progression from raw input traces to a fully trained and deployable process model.

Step 1: Input Traces

The training process begins with the input traces, which are sequences of activities directly extracted from event logs. These traces serve as the foundational data for the CNN model, as they encapsulate the sequential order of events within a business process. This sequential information is critical for the model to derive meaningful learning and predict subsequent activities effectively.

Step 2: Batching and Sequence Generation

Given the substantial size of the dataset, it is impractical to feed all traces into the model simultaneously. To manage this efficiently, a sequence generator is employed.

- Batching: The sequence generator divides the dataset into manageable batches, with each batch containing a fixed number of sequences (for example, 32 sequences per batch). This batching technique optimizes memory usage and expedites the training process by allowing the model to process multiple sequences concurrently.
- Sequence Generation: The sequence generator is also responsible for preparing the sequences. Each sequence is a subset of a trace, typically comprising a predetermined number of activities (e.g., 32 activities per sequence). To ensure uniform input sizes, padding is applied to sequences shorter than the maximum length. Padding involves adding placeholder values (often zeros) to the beginning of shorter sequences so that all sequences maintain the same length. These padded sequences, paired with their corresponding subsequent activities, are then fed into the model to facilitate learning.

Step 3: CNN Model Training

Once the data is suitably prepared, it is introduced to the CNN model. The CNN processes these input sequences, learning to identify patterns that indicate which activity is likely to follow in the sequence. The training process involves iteratively adjusting the model's weights based on the discrepancies between its predictions and the actual outcomes.

- Model Training: The training phase is conducted over multiple epochs, with each epoch representing a complete pass through the dataset. The model utilizes an optimization algorithm, such as Adam, to minimize the loss function, thereby enhancing the accuracy of its predictions.
- Early Stopping: To mitigate the risk of overfitting—where the model performs exceptionally well on training data but poorly on unseen data—early stopping is implemented. This technique monitors the model's performance on a validation set and terminates the training process if improvements cease, ensuring that the model retains its ability to generalize effectively to new data.

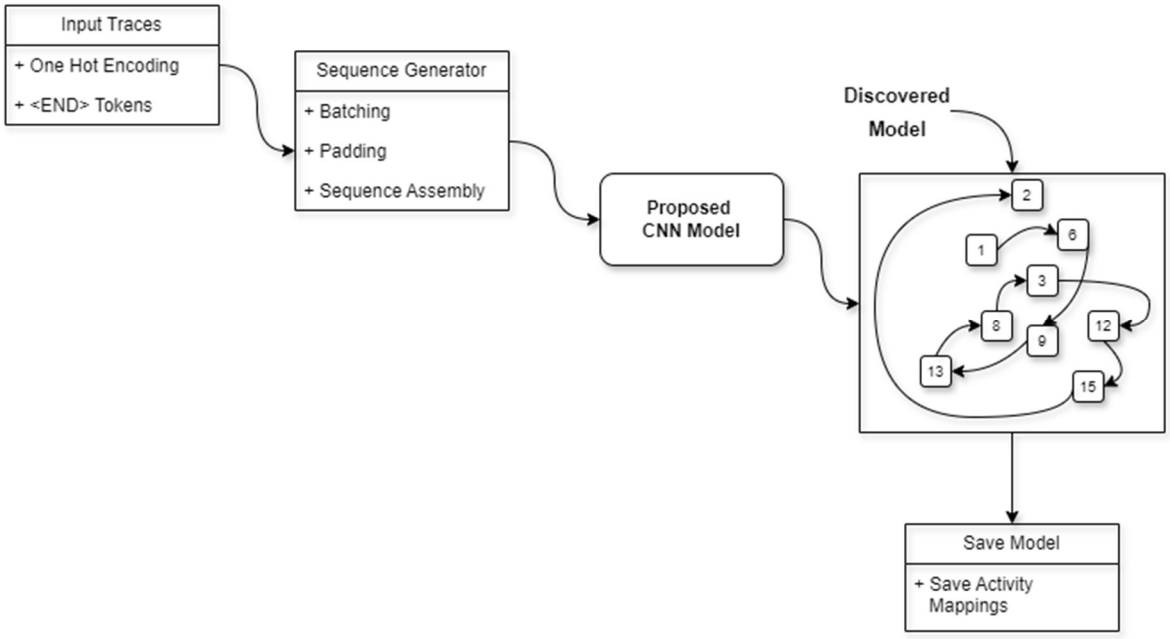

Figure 5: Training Process

Step 4: Discovered Process Model

Upon completion of the training, the CNN model is capable of generating a discovered process model. This model encapsulates the learned patterns and relationships between activities, enabling the prediction of future activities based on an input sequence.

- Probability Distributions: The CNN model outputs a probability distribution across all potential next activities, providing a quantified measure of the likelihood of each activity occurring. This distribution is instrumental in making informed predictions, supplemented by a degree of confidence.

Step 5: Model Preservation

The final step in the training process involves saving the trained model. By preserving the model's weights and configuration, the learned patterns are retained, allowing the model to be deployed for future predictions without necessitating retraining.

- Saving Activity Mappings: Alongside the model, the mappings between activity labels and their corresponding indices are also saved. This step is critical for future applications, as it ensures that new sequences can be properly encoded and decoded when the model is applied to unseen data. The activity mappings file serves as a reference that links the categorical representations used during training to the actual activity labels, thereby maintaining consistency in predictions.

4. Experimental Setup

I. Training Stage

To effectively manage the extensive data and computational demands of the ICPM 2023 Process Discovery Contest, a strategically designed experimental setup was implemented. This setup ensured that each Convolutional Neural Network (CNN) model was trained and optimized effectively, facilitating reliable predictions across a diverse and expansive dataset.

Strategic Model Training

The substantial size of the dataset necessitated a methodical approach to model training. To address this challenge, a strategy was employed whereby one CNN model was trained for every four logs, culminating in a total of 96 distinct models. This approach allowed for a distributed computational load and focused training on manageable subsets of data.

- 50 Epochs of Training: Each of the 96 models underwent training for 50 epochs. This duration was carefully selected to provide sufficient training time while mitigating the risk of overfitting. The 50 epochs ensured that the models could thoroughly learn from the data, capturing essential patterns without becoming overly specialized.
- Consistent Performance: The average training accuracy across all 96 models was approximately 70%. This consistency indicates that the models were effectively trained to generalize well across different subsets of data, maintaining a high level of predictive accuracy.

Optimization Techniques

Several key optimization techniques were employed to enhance the model's performance during the training phase:

- Hyperparameter Tuning: Critical hyperparameters, such as learning rate, batch size, and the number of epochs, were meticulously tuned to optimize performance. The Adam optimizer was utilized to dynamically adjust the learning rate, promoting efficient convergence throughout the training process.
- Early Stopping: Early stopping was implemented as a precaution against overfitting. By closely monitoring the validation loss, training was halted when improvements ceased, thereby ensuring that the models retained their ability to generalize effectively to new data.
- Dropout Regularization: Dropout was applied to the dense layers to further mitigate the risk of overfitting. By randomly disabling a portion of the neurons during each training step, dropout encouraged the models to learn more robust and generalized features.

Computing Environment

The experiments were conducted on a system equipped with an Intel Core i7 8th Gen processor, 16 GB of RAM, and an NVIDIA GTX 1050 Ti GPU. This setup provided the necessary computational power to efficiently handle the training of 96 CNN models on a large-scale dataset, ensuring that the training process was both time-efficient and resource-effective.

- Hardware: The Intel Core i7 8th Gen processor, coupled with 16 GB of RAM, provided ample processing power and memory for managing large datasets and executing complex computations.
- GPU Acceleration: The NVIDIA GTX 1050 Ti GPU played a crucial role in accelerating the training process, enabling faster computations for convolutional operations and significantly reducing overall training time.

Model Preservation

After the training phase, each model, along with its weights and configuration, was preserved. This practice ensures that the models can be readily applied to future predictions without necessitating retraining. Additionally, activity mappings were stored to maintain consistency in predictions across different models, thereby facilitating their practical application in real-world scenarios.

This experimental setup, underpinned by a robust computing environment and a strategic training approach, ensured that each model was trained both efficiently and effectively. The consistent performance across the dataset underscores the success of this methodology, providing a strong foundation for the evaluation discussed in the subsequent section.

II. Testing and Classification Process

Following the training of the CNN models, the next crucial phase involved testing and classification to assess their accuracy in predicting the next event in a sequence. This phase was carefully structured to ensure rigorous evaluation and reliable results, forming the backbone of the model's validation.

- ***Test Data Preparation***

The test logs, which were meticulously kept separate from the training data, were prepared by extracting sequences of activities using a custom function. Each activity within a trace was encoded into a numerical format consistent with the training phase. To maintain uniform input dimensions, sequences were padded to meet the required length of 32 activities per sequence. This padding was essential to ensure consistency and compatibility when feeding the sequences into the CNN models.

- ***Model Prediction and Probability Distribution***

Once the test data was prepared, each sequence was passed through the trained CNN model. The model generated a probability distribution across all potential next activities, providing a nuanced measure of the likelihood for each event following the given sequence. Rather than producing a single prediction, the model's output for each sequence was a set of probabilities, each reflecting the model's confidence in various possible next activities. These probability distributions served as the foundation for the subsequent classification process.

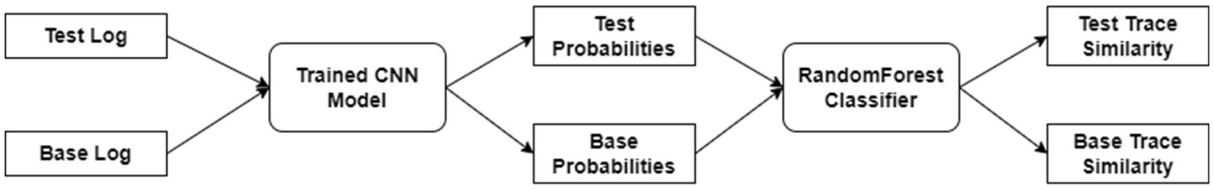

Figure 6: Testing & Classification Process

- ***Feature Extraction for Classification***

The next step was to prepare features for the classification phase. The mean and variance of the predicted probabilities were calculated, representing the overall behavior of the model's predictions. These features were critical in training a secondary classifier—a RandomForestClassifier—that used them to differentiate between traces in the test logs and those in the base logs.

- ***Classification Using RandomForestClassifier***

A RandomForestClassifier was employed to classify the traces based on the features derived from the probability distributions. The classifier was trained on features from both the test logs and the base logs, with the objective of distinguishing between the two based on their respective similarities to the learned process model. The classifier's role was to evaluate how closely each test trace aligned with the expected process model, assigning a similarity score that indicated the fit of the sequence.

- ***Evaluation of Classification Results***

The classification results were then compared against a ground truth to evaluate the model's performance. The classification was binary, labeling each trace as either "positive" (indicating a close match to the process model) or "negative" (indicating a divergence). Key metrics such as precision, recall, and F1-score were calculated to quantify the model's effectiveness. These metrics provided critical insights into the model's accuracy in classifying sequences correctly and its ability to generalize to new, unseen data.

- ***Saving the Ground Truth Logs***

To preserve the classification results, ground truth logs were generated and saved in the XES format. These logs included the sequences along with their respective classifications, providing a comprehensive record of the testing phase.

The testing and classification processes were pivotal in validating the model's effectiveness and ensuring its ability to predict future events within a business process accurately. By integrating deep learning with traditional machine learning techniques, this project achieved a comprehensive and sophisticated approach to event prediction and trace classification, laying a solid foundation for the evaluation of results in subsequent sections.

5. Evaluation & Results

5.1 Evaluation Metrics

In this project, evaluating the performance of the CNN models was crucial to understanding their effectiveness in predicting the next event in a sequence of business process activities. The evaluation metrics selected for this analysis provide a comprehensive view of the models' predictive capabilities, focusing on both their accuracy and their ability to handle different types of classification challenges.

Introduction to Metrics

Evaluation metrics are crucial for quantifying how well a model performs in predictive tasks. For this project, we focused on accuracy, precision, recall, F1-score, and the confusion matrix, each providing a unique perspective on the model's effectiveness.

- Accuracy: Measures the overall correctness of the model's predictions by calculating the proportion of correct predictions out of all predictions made.
- Precision: Evaluates the quality of the positive predictions, indicating how many of the predicted positive instances were actually correct.
- Recall: Assesses the model's ability to capture all actual positive cases, highlighting its sensitivity.
- F1-Score: Balances precision and recall, providing a single metric that is especially useful when dealing with imbalanced data.

Metric Calculation

These metrics were calculated by comparing the model's predictions with the ground truth labels in the test data:

- **Accuracy =** $\frac{\boldsymbol{True\ Positives + True\ Negatives}}{\boldsymbol{Total\ Predictions}}$
- **Precision =** $\frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives + False\ Positives}}$
- **Recall =** $\frac{\boldsymbol{True\ Positives}}{\boldsymbol{True\ Positives + False\ Negatives}}$
- **F1-score = 2 X** $\frac{\boldsymbol{Precision\ X\ Recall}}{\boldsymbol{Precision + Recall}}$

These metrics collectively provide a comprehensive view of the model's performance, highlighting both its strengths and areas for improvement.

5.2 Model Performance

Training Performance

The training phase involved training 96 models, each on a specific subset of the dataset. On average, the models achieved a training accuracy of approximately 70.65%, as depicted in the figure below.

This chart illustrates the training accuracy of each of the 96 models, with the blue line representing individual model accuracies and the red dashed line indicating the mean accuracy. The variation in accuracies across models can be observed, with some models achieving over 80% accuracy, while others dip closer to 60%.

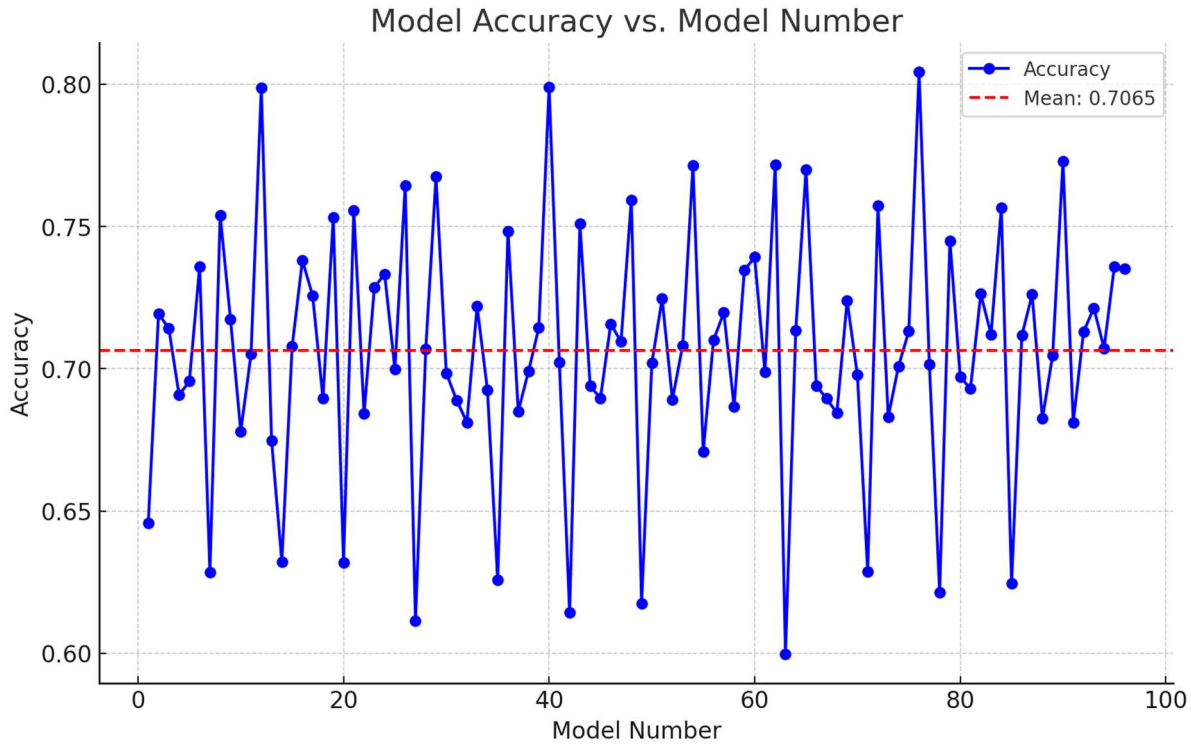

Figure 7: Training Performance

***Notable Variations and Potential Causes**:*

- **Dataset Complexity**: Certain models trained on more complex logs with diverse event sequences exhibited lower accuracy, possibly due to the increased difficulty in capturing intricate patterns within the data.
- **Data Imbalance**: Logs with a skewed distribution of activities might have led to some models being more proficient in predicting frequent activities while struggling with less common ones.
- **Overfitting**: Models that showed particularly high accuracy on the training data may have overfitted, learning specific patterns that did not generalize well to new data. This overfitting is often a concern in deep learning models when they are too closely tailored to the training set.

Despite these variations, the average accuracy of 70.65% indicates that, on the whole, the models were successful in learning the key features of the process sequences, though with some expected differences based on the unique characteristics of the subsets they were trained on.

Testing Performance

The testing phase provided an in-depth evaluation of the models' performance on unseen data, focusing on key metrics such as Precision, Recall, and F-score. The results, as visualized in the figure below, offer a clear picture of how well the models generalized beyond the training data.

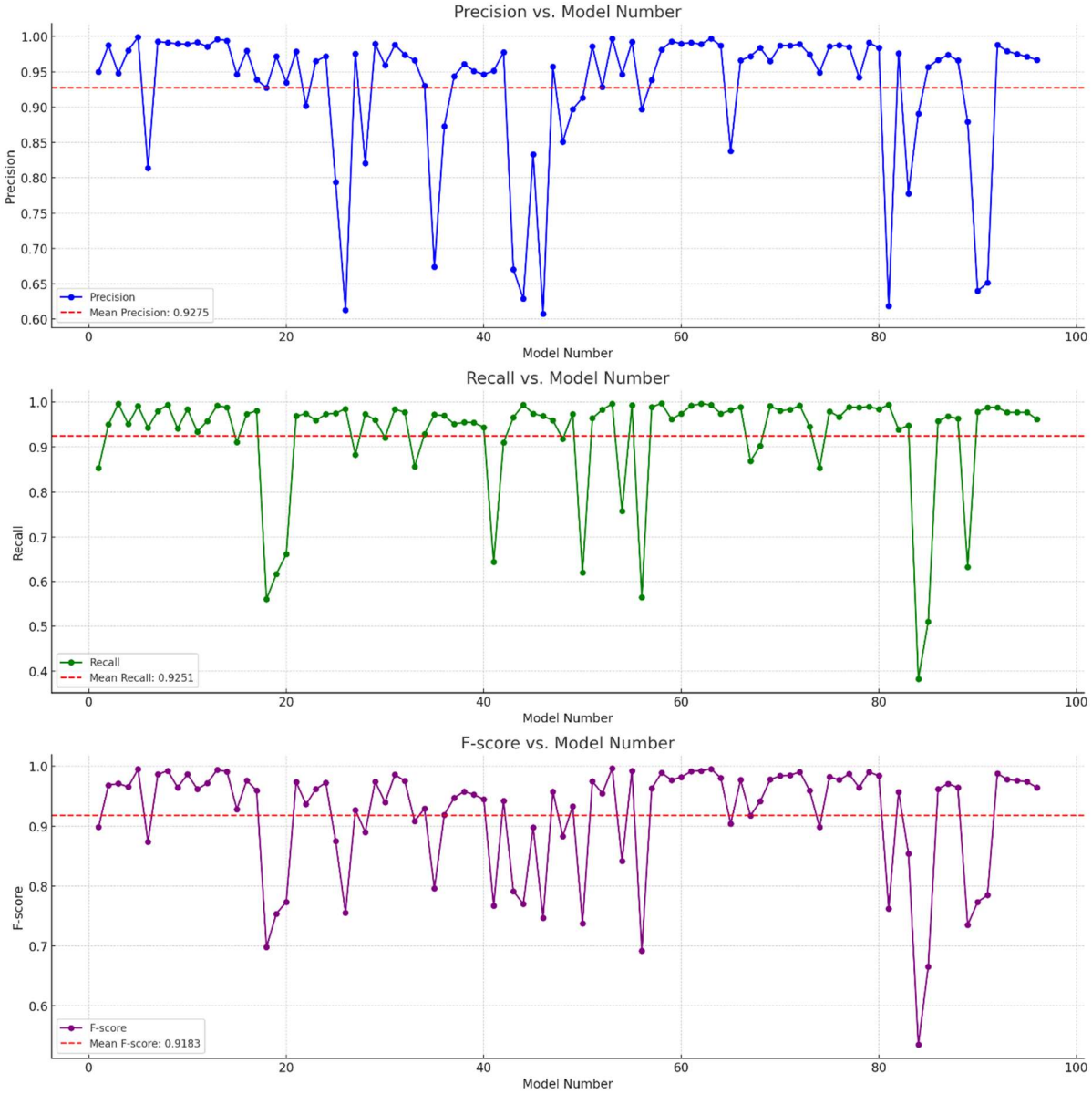

Figure 8: Testing Performance

Key Observations:

1. **Precision**:
    - **Average Precision**: 0.9275
    - **Highest Precision**: 1.0, observed in several models, indicating perfect prediction accuracy for positive cases in those instances.

- **Lowest Precision**: 0.60, observed in a few models. This drop suggests that some models struggled with making precise positive predictions, likely due to the complexity or imbalance in the test logs they encountered.

2. **Recall**:
    - **Average Recall**: 0.9251
    - **Highest Recall**: 1.0, observed across many models, demonstrating that these models were highly effective in identifying all relevant positive cases.
    - **Lowest Recall**: 0.55, indicating that some models missed a significant portion of actual positive events, which could be attributed to more challenging or less frequent sequences in the test data.
3. **F-score**:
    - **Average F-score**: 0.9183
    - **Highest F-score**: 0.99, which reflects models that maintained a good balance between precision and recall.
    - **Lowest F-score**: 0.65, observed in models where both precision and recall were relatively low, suggesting difficulties in predicting accurate sequences in those specific cases.

***Analysis**:*

- **Consistency Across Metrics**: Most models maintained high precision, recall, and F-score, indicating robust performance. However, the occasional drops in these metrics suggest areas where the models encountered sequences that were either too complex or not well-represented in the training data. The consistency of high scores in a majority of the models underscores the effectiveness of the CNN-based approach in event prediction and classification.
- **Impact of Dataset Diversity**: The variations in precision and recall across different models can be linked to the diversity and complexity of the event logs. Models that were tested on logs with more diverse or less frequent sequences tended to show lower precision and recall, indicating the need for more nuanced handling of such cases in future work.
- **Overall Performance**: The high average values for precision (0.9275), recall (0.9251), and F-score (0.9183) reflect the models' strong ability to differentiate between test and base logs, making accurate predictions in the majority of cases. This consistency in high performance across multiple metrics confirms the models' generalization capabilities, even when faced with unseen data.

In summary, the testing phase results demonstrate that the trained models performed exceptionally well in predicting events in business processes. The high average scores across all metrics suggest that the models are not only accurate but also reliable in a variety of scenarios, though certain models did exhibit variability that highlights potential areas for further improvement.

6. Challenges & Solutions

Throughout the project, several challenges arose that required innovative solutions to ensure the success and accuracy of the model. Below, the key challenges are discussed along with the solutions that were implemented to address them.

1. Time Consumption

Challenge:
The entire process, from training models to extracting probabilities and performing classification, proved to be extremely time-consuming. Initially, training the models alone took about a week, largely due to the size of the dataset and the computational demands. This was followed by an additional day for the classification process, making the overall timeline for completing a full cycle excessively long.

Solution:
To tackle this issue, extensive discussions were held, and several code optimizations were made. These changes drastically reduced the time required for the process, bringing it down from over a week to approximately three days. Furthermore, the utilization of the CAIR Cluster—a high-performance computing resource—played a crucial role in handling the extensive data processing requirements efficiently. This allowed the project to maintain feasibility within the given time constraints while improving the speed and efficiency of the operations.

2. Initial Methodological Challenges

Challenge:
The project initially aimed to predict the probability of each event in a sequence and then determine the sequence's fit to the model based on these probabilities. However, this method encountered significant problems as the multiplied probabilities often resulted in values so small that they were nearly zero, making it difficult for the classifier to function correctly. This approach was particularly problematic in handling the complexity of event sequences.

Solution:
The approach was revised to predict probabilities over the entire sequence rather than focusing on individual events. This method involved using aggregated probabilities and statistical features like mean and variance for classification. The shift not only simplified the prediction process but also maintained the overall performance. As a result, the time required for predictions was reduced from a full day to just four hours, improving both the efficiency and effectiveness of the prediction process.

3. Handling <END> Tokens in Sequence Processing

Challenge:
Initially, the models did not adequately consider the <END> tokens that signify the end of a sequence. Ignoring these tokens led to models predicting subsequent events even when a sequence should have logically ended, resulting in incorrect predictions and decreased model accuracy.

Solution:
The team identified this issue and developed a code logic that properly incorporates <END> tokens during both training and prediction. This allowed the models to accurately determine when a sequence should terminate, which in turn improved the overall prediction accuracy. By accounting for the

sequence termination, the models were able to provide more realistic predictions that closely mirrored actual process flows.

Addressing these challenges through strategic solutions not only enhanced the efficiency and accuracy of the models but also underscored the importance of adaptability in tackling complex data science problems. The improvements made during this project serve as a valuable learning experience, demonstrating how technical obstacles can be overcome with thoughtful analysis and innovative approaches.

7. Conclusion

This project has successfully demonstrated the integration of Convolutional Neural Networks (CNNs) with process mining techniques to predict and classify business process events. Through rigorous training, testing, and evaluation phases, the models developed in this study achieved significant accuracy in predicting the next steps within a sequence of business process activities, underscoring the potential of deep learning in enhancing process mining capabilities.

The CNN models, trained across a large and diverse dataset from the Process Discovery Contest 2023, consistently performed well, with an average training accuracy of 70.65% across 96 models. The testing phase further validated these results, showing high precision (0.9275), recall (0.9251), and F1-scores (0.9183) across the models. These metrics indicate that the models were not only accurate in their predictions but also robust in handling various complexities and challenges inherent in the event logs.

Several key challenges were encountered throughout the project, including the time-consuming nature of the training and classification processes, the initial methodological hurdles related to probability prediction, and the handling of sequence-ending tokens. Each of these challenges was addressed through innovative solutions, such as optimizing the code for faster execution, refining the prediction approach to consider entire sequences rather than individual events, and incorporating logic to handle sequence terminations effectively. These improvements significantly enhanced both the efficiency and accuracy of the models.

In conclusion, the project has achieved its objectives of demonstrating that CNNs can effectively be applied to process mining for event prediction and classification. The integration of deep learning techniques with traditional process mining approaches has proven to be a powerful combination, offering substantial improvements in predictive accuracy and process understanding. The insights gained from this study provide a strong foundation for future work, which could further refine these methods and explore their application to a wider range of business processes and event logs.

This project not only contributes to the academic understanding of process mining and deep learning integration but also offers practical tools that can be employed in real-world scenarios to optimize business processes and improve decision-making.

8. Learning Outcomes

1. Event Log and Sequence Data Handling

Throughout this project, I gained a deep understanding of how to handle and process complex event logs and sequence data. I learned how to extract meaningful sequences from raw data and how to prepare these sequences for analysis by machine learning models. This involved encoding activities, managing sequence lengths through padding, and efficiently batching sequences for model training. These skills have enhanced my ability to work with large, unstructured datasets and apply them effectively in predictive modelling, making me more proficient in data handling and preprocessing tasks.

2. Process Mining and Models

The project significantly improved my knowledge of process mining techniques and the underlying process models that guide business workflows. I learned how to extract valuable insights from event logs using process mining tools like PM4Py, and how to model business processes for predictive analysis. This experience deepened my understanding of process flows and how they can be analyzed and optimized for better decision-making. The integration of process mining with deep learning models like CNNs provided me with a robust framework for analysing complex business processes.

3. Process Discovery with CNN/LSTM and PM4Py

A key technical aspect of the project was exploring the use of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) networks for process discovery. I delved into the architecture of these neural networks, learning how they can capture patterns and dependencies in sequential data, which is crucial for predicting the next events in a process. Additionally, I gained hands-on experience with PM4Py, applying advanced process mining techniques to real-world data. This experience has given me a solid foundation in both deep learning and process mining, equipping me with the skills to tackle complex predictive modelling tasks.

4. Command Line and Shell Scripting on Linux

Working on this project also allowed me to sharpen my skills in using command line interfaces and shell scripting on Linux. Given the extensive data processing and model training required, I had to automate various tasks using shell scripts, manage file operations, and control job executions in a Linux environment. This hands-on experience has not only streamlined my workflow but also made me more proficient in handling large-scale data processing tasks, enhancing my ability to operate efficiently within a Linux ecosystem.

5. Next Event Prediction

A major learning outcome of this project was understanding the mechanisms involved in predicting the next event in a sequence using probability distributions. I learned how to build and optimize models that predict future events based on historical data, using the output from CNN models. This included understanding how to extract features like mean and variance from these predictions and applying them

to improve process outcomes. This knowledge is directly applicable to various real-world scenarios where predicting future events can lead to more informed decision-making.

6. Overcoming Challenges and Optimization

Throughout the project, I faced several challenges, such as the time-consuming nature of the process, handling sequence-end tokens, and initial methodological hurdles. Overcoming these challenges taught me the importance of flexibility and innovation in problem-solving. I learned how to optimize processes through code refinement and strategic adjustments, significantly reducing the time required for model training and classification. These experiences not only enhanced my technical skills but also provided me with valuable insights into the importance of iterative improvements in complex projects.

9. References

1. Di Mauro, N., Appice, A., & Basile, T. M. A. (2019). Activity Prediction of Business Process Instances with Inception CNN Models. In *AI*IA 2019 - Advances in Artificial Intelligence: XVIII International Conference of the Italian Association for Artificial Intelligence, Bari, Italy, November 19–22, 2019, Proceedings* (pp. 348-361). Springer. https://doi.org/10.1007/978-3-030-35166-3_25

2. Weytjens, H., & De Weerdt, J. (2020). Process Outcome Prediction: CNN vs. LSTM (with Attention). In *BPM 2020 Workshops, LNBIP 397* (pp. 321-333). Springer. https://doi.org/10.1007/978-3-030-66498-5_24

3. Tax, N., Verenich, I., La Rosa, M., & Dumas, M. (2017). Predictive Business Process Monitoring with LSTM Neural Networks. In *CAiSE 2017 - Advanced Information Systems Engineering* (pp. 477-492). Springer. https://doi.org/10.1007/978-3-319-59536-8_30

4. Ashrafi, N., Abdollahi, A., Placencia, G., & Pishgar, M. (2024). Process Mining/Deep Learning Model to Predict Mortality in Coronary Artery Disease Patients. *medRxiv*. https://doi.org/10.1101/2024.06.26.24309553

5. Pishgar, M., Harford, S., Theis, J., Galanter, W., Rodríguez-Fernández, J. M., Chaisson, L. H., Zhang, Y., Trotter, A., Kochendorfer, K. M., & Darabi, H. (2022). A process mining-deep learning approach to predict survival in a cohort of hospitalized COVID-19 patients. *BMC Medical Informatics and Decision Making*, 22:194. https://doi.org/10.1186/s12911-022-01934-2

6. Venugopal, I., Töllich, J., Fairbank, M., & Scherp, A. (2021). A Comparison of Deep-Learning Methods for Analysing and Predicting Business Processes. *International Joint Conference on Neural Networks (IJCNN)*. https://doi.org/10.1109/IJCNN52387.2021.9533742

7. Mehdiyev, N., Fettke, P., & Evermann, J. (2017). A Multi-stage Deep Learning Approach for Business Process Event Prediction. *2017 IEEE 19th Conference on Business Informatics (CBI)*, 119-128. https://doi.org/10.1109/CBI.2017.46

8. Xia, C., Xing, M., Ye, Y., & He, S. (2022). A Process Mining Framework Based on Deep Learning Feature Fusion. *Proceedings of the 41st Chinese Control Conference (CCC)*, 7412-7418.

9. Lu, Y., Chen, Q., & Poon, S. K. (2022). A Deep Learning Approach for Repairing Missing Activity Labels in Event Logs for Process Mining. *Information, 13*(5), 234. https://doi.org/10.3390/info13050234

10. Mehdiyev, N., Evermann, J., & Fettke, P. (2018). A Novel Business Process Prediction Model Using a Deep Learning Method. *Proceedings of the 2018 IEEE International Conference on Business Informatics (CBI)*, 119-128. https://doi.org/10.1109/CBI.2018.100

11. Al-Jebrni, A., Cai, H., & Jiang, L. (2018). Predicting the Next Process Event Using Convolutional Neural Networks. *Proceedings of the 2018 IEEE International Conference on Business Informatics (CBI)*, 332-339. https://doi.org/10.1109/CBI.2018.100

12. Park, G., & Song, M. (2020). Predicting Performances in Business Processes Using Deep Neural Networks. *Decision Support Systems, 129*, 113191. https://doi.org/10.1016/j.dss.2019.113191

13. Evermann, J., Rehse, J.-R., & Fettke, P. (2017). Predicting Process Behavior Using Deep Learning. *Decision Support Systems, 100*, 129-140. https://doi.org/10.1016/j.dss.2017.04.003

14. Kamala, B., & Latha, B. (2022). Process Mining and Deep Neural Network Approach for the Prediction of Business Process Outcome. *International Conference on Communication, Computing and Internet of Things (IC3IoT)*. https://doi.org/10.1109/IC3IoT53935.2022.9767941