Business Process Anomaly Classification Utilizing Sensor Data

Master thesis by Erik Schwan
Date of submission: November 30, 2021

1. Review: Prof. Dr. Max Mühlhäuser
2. Review: Dr. Timo Nolle
Darmstadt

TECHNISCHE UNIVERSITÄT DARMSTADT

Computer Science Department
Telecooperation Lab
Smart Proactive Assistance

Erklärung zur Abschlussarbeit gemäß §22 Abs. 7 und §23 Abs. 7 APB der TU Darmstadt

Hiermit versichere ich, Erik Schwan, die vorliegende Masterarbeit ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die Quellen entnommen wurden, sind als solche kenntlich gemacht worden. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Mir ist bekannt, dass im Fall eines Plagiats (§38 Abs. 2 APB) ein Täuschungsversuch vorliegt, der dazu führt, dass die Arbeit mit 5,0 bewertet und damit ein Prüfungsversuch verbraucht wird. Abschlussarbeiten dürfen nur einmal wiederholt werden.

Bei der abgegebenen Thesis stimmen die schriftliche und die zur Archivierung eingereichte elektronische Fassung gemäß §23 Abs. 7 APB überein.

Bei einer Thesis des Fachbereichs Architektur entspricht die eingereichte elektronische Fassung dem vorgestellten Modell und den vorgelegten Plänen.

Darmstadt, 30. November 2021

E. Schwan

Zusammenfassung

Diese Arbeit zielt darauf ab, die Möglichkeiten zur Verbesserung von Process-Mining-Modellen durch Nutzung von Sensorinformationen zu untersuchen. Durch die aktuelle Digitalisierung im Zuge der vierten industriellen Revolution (Schwab, 2017) werden Sensoren für Unternehmen allgegenwärtig. Die Kernaufgabe des Process Mining besteht darin, betriebliche Abläufe in Unternehmen anhand aller verfügbaren Daten zu analysieren. Aus diesem Grund ist es naheliegend, die Sensordaten, welche über die physische Welt erfasst wurden, in die Process-Mining-Modelle einzubeziehen.

Um die Vorteile von Sensordaten für das Process Mining zu untersuchen, werden in dieser Thesis siebzehn verschiedene Prozessmodelle mit zwei Datensätzen für die Aufgaben der Prozessvorhersage und der Klassifizierung von Anomalien bewertet. Der erste Teil dieser Thesis beschreibt den Aufbau eines speziellen Simulationssystems, welches zur Erzeugung von Ereignisprotokollen mit Sensorattributen für diese Forschung entwickelt wurde. Beide Datensätze der Thesis werden mithilfe des Simulationssystems generiert. In einem der Datensätze sind die Sensorattributdaten ebenfalls simuliert, während sie in dem anderen Datensatz aus realen Sensordaten stammen. Für die Simulation natürlich vorkommender Fehler in den Datensätzen, können zwei verschiedene Anomalien den Kontrollfluss und sieben Sensoranomalien die Messungen verändern. In dieser Arbeit wird ein statistisches Prozessmodell auf der Grundlage einer RNN-Architektur verwendet, um einen Prozess aus einem Ereignisprotokoll zu lernen. Zur Verbesserung dieses Prozessmodells werden verschiedene Sensordarstellungen aus dem Bereich der Signalverarbeitung sowie dem Bereich des maschinellen Lernens als Vorverarbeitungsmethode bewertet.

Das vorgeschlagene Modell mit einer normalisierten RNN- oder normalisierten CNN-Sensordatenvorverarbeitung schafft es, die Prozessdarstellung sehr effektiv zu verbessern. Außerdem erweist sich dieses Modell als robust gegenüber Eingangsrauschen, da es den Prozess aus einem Datensatz erlernen kann, der 30% Sensoranomalien und 30% Kontrollflussanomalien enthält. Das endgültige Modell zeigt, dass die Einbeziehung von

Sensorattributen die Prozessvorhersage und die Klassifizierung von Anomalien im Vergleich zu einem Modell ohne Sensordaten deutlich verbessern kann.

Abstract

This thesis aims at studying possibilities to leverage sensor information about events for process mining tasks. With the current transformation of businesses during the fourth industrial revolution (Schwab, 2017), an increasing amount of data about the physical world is captured by sensors. While process mining wants to analyze operational processes in businesses from all available data, it might be useful to include the sensor data into the process mining models.

To study the benefits of sensor data for process mining, seventeen different process model are evaluated with two datasets for the tasks of process prediction and anomaly classification. The datasets are generated by a simulation system dedicated to event log generation with sensor attributes which are provided in this thesis. In one of the datasets, the sensor attribute data is simulated whereas it is drawn from real-world sensor data for the other dataset. Noise in the dataset is introduced by two different anomalies in the control flow and seven sensor anomalies.
This thesis employs a statistical process model based on an RNN architecture to represent a process contained in an event log. To improve the process model, various possible sensor representations from the signal processing domain as well as the machine learning field are evaluated as a pre-processing method.

The proposed model with a normalized RNN or normalized CNN sensor data pre-processing manages to improve the process representation most effectively. This model presents itself as robust against input noise by managing to learn the process from a dataset containing 30% sensor anomalies and 30% flow anomalies. The final model demonstrates that the incorporation of sensor attributes can improve the process prediction and anomaly classification results in comparison to a model without sensor data inference capabilities.

Contents

1. Introduction

Modern companies track the execution of processes with information systems to ensure a smooth operation of the business. To take full advantage of these systems, it is not only necessary to document the processes but also check for their conformance while they are executed. Processes can vary in the execution order of activities or in the processing of the activity itself. Anomalous processes appear to generate significantly different data compared to their correctly executed counterparts. The difficulty to detect a deviating pattern increases with the input size, variation in the format as well as the amount of noise in the data.

Traditionally, process mining research is mostly focused on the process execution based on event data of established infrastructure such as enterprise resource planning (ERP) or customer relationship management (CRM) systems. Hence, this research analyzes processes based on the available data in corporate databases with limited event insights. The resulting high-level process model has the benefit of being easily interpretable for humans but disregards a lot of detailed information in each process step. With the increasing digitalization of businesses, very detailed information of the process execution is captured. For example, smart production environments record comprehensive information within each process step from various sensor sources. This information is very insightful to estimate the execution status of a process and consequently monitor for deviations.

This thesis seeks to provide a process mining method that is able to utilize sensor event attributes. The actual contribution of this thesis to the scientific community is divided into three research questions listed below.

> ***RQ1**: How can process mining tasks with sensor event attributes be evaluated?*
>
> ***RQ2**: What is a effective represented of sensor data in a process model?*
>
> ***RQ3**: Are sensor event attributes helpful for anomaly detection or process prediction tasks?*

To be able to answer these questions, a fundamental understanding of process mining and the underlying technology is required. Therefore, Chapter 2 will present the history and current approaches to process mining with a focus on deep learning and its application in the process mining domain. To get insights into relatable methods and transferable knowledge, various research from the domains of process prediction, process anomaly classification, sensor data processing, and process mining evaluation will be presented in Chapter 3. Chapter 4 use all gathered information to present a dataset generator designed to create authentic process data with sensor attributes. The main process mining system, its evaluation setup as well as experiments on process mining tasks will be presented in Chapter 5. The chapter is followed by a discussion of the finding with suggestions for possible developments and finally summarized within a conclusion.

Details of the described software in this thesis can be observed in the public source code repository[1].

[1] https://github.com/ErikBird/SensorProcessMining

2. Background

This chapter connects the academic backgrounds of process mining with its research challenges. The first section 2.1 will introduce the reader to a technology with the name of *deep learning*, which will be fundamental for this thesis. Afterwards, section 2.2 will first cover the technical foundation as well as the historical background up to the current state of research of process mining. Section 2.3 will elaborate on the different task perspectives of deep learning based process mining compared to its traditional counterpart. Finally, Section 2.4 presents different anomalies that can occur in event logs with sensor event attributes.

2.1. Deep learning

The ever-growing integration of computer technologies in all areas of our life increases the amount of captured data as well. Machine learning is a discipline that aims to turn this data into knowledge. Various specific techniques with different characteristics have emerged in this field of study to achieve the goal of machine learning.

Deep learning (DL) is one of these specific machine learning techniques, based on artificial neuronal networks. DL has become very popular as it allows modelling complex structures by learning a hierarchy of simple concepts (Goodfellow et al., 2016). *Multilayer perceptron networks* (MLP) have proven to provide the greatest practical value of all types of neuronal networks (Bishop, 2013, p.226). Deep Learning methods based on MLPs manage to dramatically improve the state-of-the-art of prediction tasks in the field of natural language processing, computer vision or computational biology. In this section, a brief introduction of some Deep learning concepts of relevance for this thesis will be provided. Please refer to Bishop (ibid., chapter 5) for a comprehensive overview of the topic and implementation details.

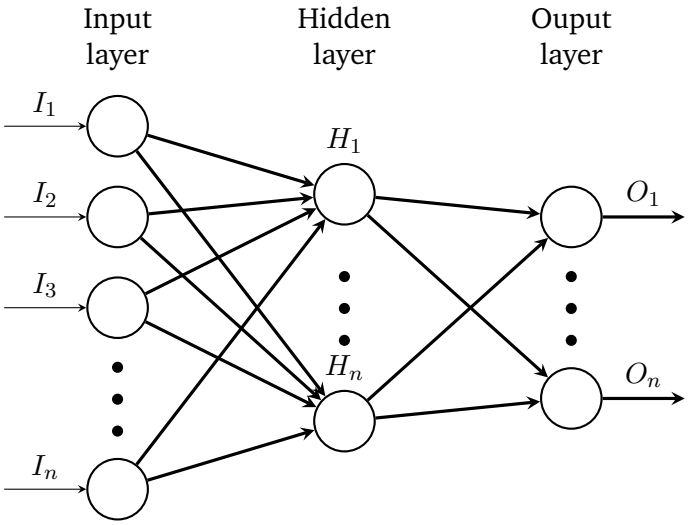


Figure 2.1.: Feed Forward Multi Layer Perceptron

The core functionality of an MLP can be best explained by a *feed forward network* (FFN) depicted in figure 2.1. A circle represents a neuron that is directly connected to all neurons of the subsequent horizontally aligned neurons, called a *layer*. The information in this network moves only forward, from the input nodes through the hidden nodes to the output nodes. The value of each neuron is determined as the weighted sum of all outgoing signals of previous neurons. A neuron propagates its value after it is projected by a non-linear function called *activation*. The connections between neurons are called *parameter* and their weight needs to be learned to have a functioning neuronal network. To achieve this, the backpropagation algorithm indicates how a machine should change its parameter weight in each layer from the representation in the previous layer (Rumelhart et al., 1986).

One more regularized form of an MLP is a *convolutional neural network* (CNN), which was initially developed for images (Lawrence et al., 1997). A convolutional layer slides a kernel of fixed size over the input features. This results in abstract feature maps of local areas of the input. Afterwards, the dimensionality of these feature maps is reduced by a pooling layer. CNN networks usually contain multiple consecutive convolutional layers to take advantage of hierarchical patterns in data. Each convolutional layer thereby increases the complexity of the data they are representing with little connectivity in the network. This enables a CNN to learn effectively with fewer parameters than an FFN if

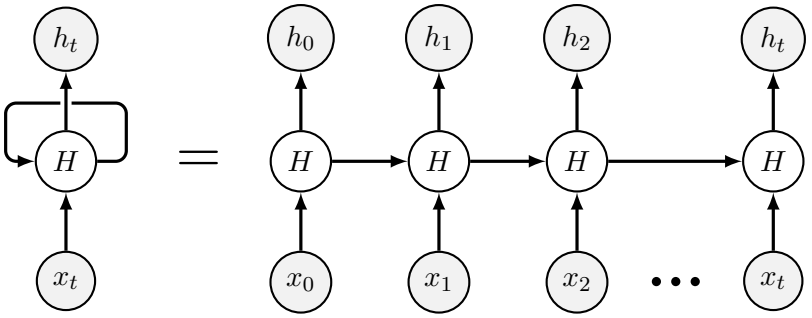


Figure 2.2.: Recurrent Neuronal Network

the data is locally coherent.

A *recurrent neuronal network* (RNN) is another special form of an MLP (Rumelhart et al., 1986) This network is designed to model the sequential characteristic in data. An RNN differentiates itself from an FFN by retaining its state while processing the next sequence of inputs. Therefore, the RNN can relate the current input with their previous perceptions. In the most basic architecture, a recurrent neuron concatenates the output of the previous step to the input for a prediction. The training of such a network can be imagined as an FFN where each layer represents a time step as illustrated in figure 2.2. Long sequential input data can therefore create very deep networks. The backpropagation in these deep networks encounter the vanishing gradient problem (Hochreiter, 1991).
Due to this problem, it is difficult to learn long-term dependencies with a simple RNN. To increase the long term memory of RNNs, Hochreiter and Schmidhuber (1997) proposed to use a different model architecture called *long short-term memory* (LSTM). An LSTM unit is composed of input, output and forget gate as well as a cell that stores a state. The three gates control the in and outflow of information into the cell. The mechanisms of these gates allow LSTMs to remember information for long periods. Due to the exceptional performance of LSTMs, they became very popular for various sequential machine learning tasks (Graves and Schmidhuber, 2005; Mayer et al., 2006; Schmidhuber et al., 2005). Cho et al. (2014) proposed a different RNN mechanism called *gated recurrent unit* (GRU). A GRU is similar to an LSTM but manages to have fewer parameters with a similar performance (Chung et al., 2014).

The various model variants show how flexible deep learning can adapt to the target domains. In the next chapter, the domain of this research, *process mining*, will be presented followed by the current state of deep learning in process mining.

2.2. Process Mining

Processes are everywhere. In this work, a *process* is defined as a sequence of activities for achieving a certain goal. One participates in a process if one orders bread at the bakery or brushes their teeth.

Processes are ubiquitous in the business environment as well and describe how an enterprise will achieve its business goals. The quality of the business processes influences the performance of an enterprise (Leymann and Altenhuber, 1994, p.326). Thus, techniques to manage and support business processes are an active research area for decades (Georgakopoulos et al., 1995; Leymann and Altenhuber, 1994; Reinwald and Wedekind, 1992). These techniques rely on some form of documentation of the actual process execution. Therefore, data about the activities during the process execution needs to be recorded. This data is referred to as *event log*. The event log contains a table of events where each event refers to an activity. Event logs should store as much useful information about events as possible. In this thesis, this additional event information will be referred to as an *event attribute*.

Historically, Business Process Management research was concerned to assist integrating information technology into processes within an organization (Davenport et al., 2018). Therefore, human experts modelled the practised processes into explicit process models (Georgakopoulos et al., 1995; Hollingsworth and Hampshire, 1995). The process models enabled business reengineering practices (Georgakopoulos et al., 1995, p.130) and ensured the database integrity (Leymann and Altenhuber, 1994, p.328).

The first invented algorithmic mining approaches extract process models automatically from the event log (Agrawal et al., 1998). This process mining task is usually referred to as *process discovery* (van der Aalst et al., 2012). Automatic process discovery makes the process models more tangible since no human experts are needed to formulate them. Based on an existing process model, Van Der Aalst (2012) identified two other major tasks for process mining. The first task type is *process conformance*, where event logs are compared with a given process model. In the second task type, *process enhancement*, the process models should be improved or extended.

The performance of process mining for process conformance or enhancement always depends on the quality of the given process model. However, the process model formation has a conflict of objectives between accuracy and interpretability (Karolina Dziugaite et al., 2020). Process models from a representative event log often contain too much detail to be easily comprehensible. On the other hand, interpretable models often deviate

strongly from the real processes. It is not uncommon that only 40% of an event log are represented in the process model (Rozinat et al., 2009). Van Der Aalst (2012) criticises that the process model ofter shows the 'PowerPoint Reality' by masking away the variability of the real processes. Some of the process models get even simplified after their discovery, to become humanly interpretable (Chapela-Campa et al., 2019; De San Pedro et al., 2015).

Process Mining techniques based on explicit process models have the advantage that they are theoretically humanly interpretable. Yet accurate process models are often too complex to grasp easily. The next section is going to present a different process mining approach that is dedicated to embody complex processes but disregards a humanly interpretable process model entirely.

2.3. Deep learning based Process Mining

In recent years, deep learning-based methods gained popularity in process mining research and improved the state-of-the-art process mining performance (Tax et al., 2020). DL based process mining methods learn their internal representation as a hierarchy of concepts as described in section 2.1. The resulting model representation is very complex and not humanly interpretable. Sometimes the learned DL based process model is referred to as a *stochastic process model* (Camargo et al., 2021).

With the adaption of ubiquitous computing in processes monitoring, the event log complexity is growing. Each event can be documented with additional captured information bout its execution. These *event attributes* can be for example the time of execution, the responsible person or captured sensor data. Furthermore, the prevalence of Process-Aware Information Systems enables recording an extensive event log of the performed processes. Traditional process discovery algorithms become unsuitable for these big datasets due to their quadratic or even exponential run time requirements (Augusto et al., 2019). On the other hand, DL based process mining methods scale linearly to the available data and can represent very complex relationships in the event log without relying on handcrafted features. Since these characteristics are essential if high dimensional event attributes, such as sensor data, should be incorporated into the process mining models, this thesis will focus on DL based process mining methods.

The explicit knowledge about the process model is so deeply integrated into traditional process mining methodologies that the task classification by van der Aalst et al. (2012)

does not apply to DL based process mining techniques. Today, DL based process mining is used for some form of *process prediction* like *process simulation* (Camargo et al., 2019) or *predictive process monitoring* (Di Francescomarino et al., 2018) and *anomaly classification* (Junior et al., 2020; Nolle et al., 2019). In process prediction, the learned model is used to estimate a probability distribution of the next event, given some history of event logs. If the prediction also includes the event attributes, it is called a *multi-perspective* prediction (Böhmer and Rinderle-Ma, 2016).
The task of anomaly classification want's to detect and classify potential anomalies from an event log. This can be done by directly predicting the anomalies end to end (Junior et al., 2020) or by discriminating against a reconstruction error (Krajsic and Franczyk, 2020; Nolle et al., 2016, 2018b) or the distance between an event prediction and a given event (Nolle et al., 2018a, 2019). Details on the systems in the current literature to solve these tasks will be presented in the next chapter. Table 2.1 offers an overview of all process mining research tasks including DL based process mining.

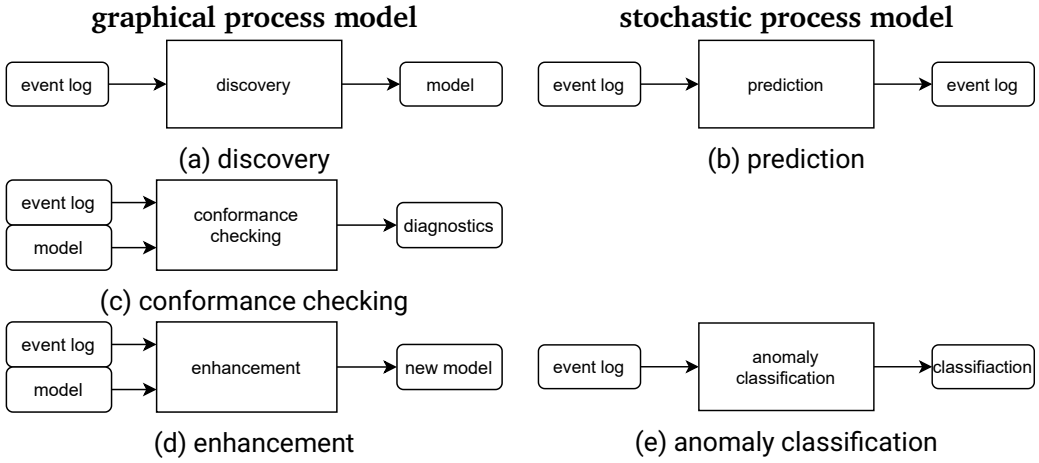


Table 2.1.: The basic types of process mining tasks explained in terms of input and output separated by methodology. Extension of van der Aalst et al. (2012, Fig.3)

2.4. Event Log Anomalies

To develop systems for process anomaly classification, it is crucial to understand how anomalies can occur in business processes in the first place. Anomalies are also referred

to as abnormalities, deviants, or outliers in the data mining and statistics literature (Aggarwal, 2017). These anomalies are usually caused by errors in the execution or documentation of a process but might also indicate a previously unknown underlying process.

According to Chalapathy and Chawla (2019), anomalies can be distinguished by the classes of *Point Anomalies, Contextual Anomalies* and *Collective Anomalies*.
Point Anomalies and their detection are extensively researched (Pang et al., 2021, chap. 8.4). They represent an irregularity in one or many consecutive data points. For example, it is suspicious that one received e-mail is written in a different language than all other mails.
Contextual Anomalies are data points that can only be considered as deviation given some specific (conditional) context. Increasing temperature data from an outside sensor might only be considered irregular if one knows that they are recorded during midnight.
Collective Anomalies can only be detected by observing a group of data points together. Each information in this group appears to be normal whereas the connection of the group of points exhibits an unusual structure. A single access log in an authorization system might appear normal. However, if the same account accesses various resources quickly and in alphabetical order, the access log indicates a hacked account.

Anomalies in an event log can occur in the control flow (i.e., the ordering of activities) or the data perspective of the event (i.e., the event attributes). Since datasets with natural anomalies in the event log usually don't describe the anomalies, explicit knowledge about the anomalies is mostly found in event logs with generated anomalies.

Control flow anomalies change the fundamental event sequence of the process. This has been done by *altering the activity order* (Böhmer and Rinderle-Ma, 2016), *adding a new activity*(Junior et al., 2020; Nolle et al., 2019), *skipping events* (Nolle et al., 2018b) or *repeating/doubling event sequences* (Nolle et al., 2016).

Data perspective anomalies are event log anomalies caused by issues in event attributes. Böhmer and Rinderle-Ma (2016) and Junior et al. (2020) modeled anomalies in the timing event attribute. Böhmer and Rinderle-Ma (2016) generated the anomaly *New Date*, where the attribute of the start timestamp is changed. Junior et al. (2020) used the anomalies *Early* and *Late* for events that are executed too early or too late.
Nolle et al. (2018b) introduced anomalies affecting the process execution by users outside their authority. This was done in a direct activity or a long-term dependent activity. Böhmer and Rinderle-Ma (2016) and Junior et al. (2020) both modeled the anomaly of adding a random incorret attribute to an event.

No process mining research so far has been directed towards data perspective anomalies based on sensor data. However, signal anomalies in sensor data is a viable research community on their own. Since sensor anomalies are an integral part of this thesis, in the following the survey about **sensor data anomalies** will be broadened outside of the process mining literature.

Leigh et al. (2019) and Teh et al. (2020) provide classification frameworks for sensor data anomalies. The framework of Leigh et al. (2019) is developed for anomalies occurring in water sensors. This framework also groups the anomalies into three groups by the difficulty to detect them. The classes presented by Teh et al. (2020) are derived from a literature review. Table 2.2 contrasts both classification frameworks.

Teh et al. (2020)	Leigh et al. (2019)
Outliers	Large sudden spike Clusters of spikes Small sudden spike Impossible values Out-of-sensor-range values
Missing data	Missing values
Bias	Constant offset (e.g. calibration error)
Drift	Sudden shifts Drift
Noise	High variability
Constant value	Low variability/persistent values
Uncertainty	
Stuck-at-zero	

Table 2.2.: Comparison of the sensor anomaly classification of the authors Leigh et al. (2019) and Teh et al. (2020).

The anomalies described by Teh et al. (2020) are generally more descriptive towards the informational interpretation of the anomaly. Anomalies stated by Leigh et al. (2019) express mostly the course of the sensor measurement without a deeper reflection on the content. This can be seen by the logical difference of the anomaly *noise* against *high variability*. Noise needs knowledge about the underlying signal without the noise to be differentiated whereas a high signal variability can also be the result of a very volatile signal.

Furthermore, Leigh et al. (2019) differentiates anomalies more fine-grained by their temporality. A short and strong *drift* is seen as *sudden shift*. Multiple *outlier* are labelled *clusters of spikes* or *small/large sudden spike* as single value.

Described sensor anomalies in this thesis will mostly follow the classifications of Teh et al. (2020) but incorporate the temporal differentiation of outlier by Leigh et al. (2019).

3. Related Work

As elaborated in section 2.3, deep learning models are very suitable for process mining tasks and offer the flexibility to extend the input data with high dimensional values such as sensor measurements. Therefore, this chapter will narrow the focus of the presented literature onto deep learning-based methods and their realization in the process mining domain. The two sections 3.1 and 3.2 will provide an overview of the current state of research for the two main tasks for deep learning based process mining, *process prediction* and *anomaly classification*. Afterwards, chapter 3.3 present methodologies in the research field of sensor data processing. Finally, the last section 3.4 describe current evaluation possibilities in the process mining field.

3.1. Process Prediction

The goal of process prediction is to estimate coming events in a process instance given previous event data from an event log. Multiple solutions for this task have been proposed, which vary not only in the exact task formulation but also in the fundamental deep learning model.

To predict the id of the next event, Mehdiyev et al. (2017) developed a multi-stage input pipeline. At first, n-grams of the activity names were created and concatenated into one label. This n-gram is then hashed into a binary representation to reduce the input dimension. Afterwards, the representation is further decreased via two auto-encoder layers and fed into an FFN. The model finally predicts the id of the next event as a one-hot classification task.

Di Mauro et al. (2019) propose a CNN for next event prediction. To be able to represent the event data as CNN input, the activity id and temporal attributes of multiple timesteps are concatenated to form a two-dimensional input. The multi-layer CNN is trained with

multiple one-dimensional convolution sizes in parallel. The experiment measures a performance improvement in comparison with LSTM networks. Similar results have been made by Pasquadibisceglie et al. (2019), which use only the activity is for the CNN inputs. Heinrich et al. (2021) proposed a gated convolutional neural network and a key-value-predict attention network for the same task. It was found that both networks perform very well for the next event prediction but each showed superior performance on different datasets. This suggests that different models can capitalize on specific dataset characteristics and therefore need to be adapted to excel.

The most popular approach for process prediction with deep learning is the use of RNN networks since they can model the sequential nature of the task natively. Evermann et al. (2017) used an LSTM network for the process prediction task. This research also joins multiple names to create the input. In this method, the activity name and the name of the resource associated with the event were combined and projected into an embedding space for each timestep. The prediction label of this method is a vector in the same embedding space. Thereby the output not only predicts the next activity but also the name of its event resource. However, they also acknowledged that this methodology is only feasible for small numbers of distinct resource-activity pairs.

Camargo et al. (2019) and Hinkka et al. (2020) developed their models with a focus on task scalability. Therefore, Camargo et al. (2019) developed an LSTM model for next event prediction. To reduce the dimensionality of the input, all categorical event attributes were grouped. Afterwards, an additional model to map all groups and the activity ids into an embedding space is learned. Continuous attributes, such as the elapsed time between activities is not projected into the embedding space. These attributes have shown to be most indicative if they are Log-normalized and scales into a [0, 1] range. The benefit of the normalization of continuous event attributes into the [0, 1] range is also attested by Schönig et al. (2018).

Hinkka et al. (2020) used a one layer GRU RNN model for the next activity prediction task. The model is built upon the model in Hinkka et al. (2019) which improves the training performance by filtering infrequent activities. To remain efficient, Hinkka et al. (2020) decreased the input size by clustering the categorical event attributes and using these attribute cluster encodings as input vectors. Only one-hot encoded activity ids and the category vectors were used as input for each timestep. This results in only a minor performance improvement of the predictions but a significant decrease in the training duration.

Tax et al. (2017) proposed LSTM based model to predict not only the next event but also its timestamp. The activity IDs of the input data were provided with one-hot encoding.

Additionally, three time-based input features served as input for each timestep. The duration of the activity, the time within the day and time within the week to model the office hours.

Lin et al. (2019) proposed *MM-Pred*, an RNN architecture with multiple components. The model first encodes categorical attributes and the event id with separate LSTMs. Afterwards, the modulator component combines the resulting vector representations into one hidden representation. The combination is based on a weight vector which indicates the relevance of each attribute for the prediction. Finally, a two-layer LSTM decodes the hidden representation to its labels separately for the next event id and its attributes.

No process prediction method so far tried to incorporate high dimensional event attributes in general or sensor data specifically.

3.2. Process Anomaly Classification

Anomaly classification systems based on deep learning methods are commonly implemented with reconstruction or prediction-based approaches in the process mining domain. These systems first learn a process prediction model or a denoising reconstruction model from a given event log. As described in the previous section, process prediction systems learn to predict the next event, given the previous events. Reconstruction models usually input the event of the whole instance and predict the same instance as output. Both approaches classify abnormal events by discriminating the data based on the distance between the given label to its predicted counterpart. Anomaly detection based on a distance discriminator has the benefit that no labelled anomalous data is required for the development of the system.

Nolle et al. (2018b) proposes such a reconstruction-based anomaly detection system. The model is represented by an autoencoder that denoises event case data. The case data is represented by the event id next to the event attribute of the responsible user of all events. The input classes are one-hot encoded and padded to the size of the longest case. To measure the reproduction distances, multiple strategies were proposed. Either the mean squared error over the whole case reproduction is measured or each input information is measured individually. The second option enables to locate the source of the anomaly. Evaluated against multiple alternative methods, datasets and anomaly proportions, the suggested method presents itself with leading performance.

Krajsic and Franczyk (2020) suggests a similar system for online anomaly detection based on autoencoder. The systems differentiate slightly since only the event id information is integrated into this system and the reproduction error can only be computed on the whole case.

Krajsic and Franczyk (2021) proposed a anomaly detection system based on a bidirectional lstm variational autoencoder with self-attention mechanism. The event data is encoded via the graph embedding node2vec and all numeric attributes are normalized. The proposed process prediction model is trained on data without anomalies. Finally, the anomaly detection capability of the system is evaluated on a dataset with 10% anomalies containing three control flow anomalies and two data perspective anomalies of temporal nature. Events are classified as anomalous if the error distance of the predicted event and the input is above-average.

Nolle et al. (2018a) proposes a sequence-to-sequence recurrent neural network for a prediction-based multi-perspective anomaly detection system. Nolle et al. (2019) extended this system with two additional variants of the model. For all models, the event data is first preprocessed by projecting the nominal attributes and the activity names into an embedding space to reduce their input dimension. In the simplest variant, the model predicts the next event data, containing the activity id and all its attributes solely on the input of the previous events. The second variant has two stages to predict at first the control flow and afterwards the event attributes. To predict the control flow, a network like the simplest variant predicts only the id of the next activity. The inner state of this network then gets access to the actual next activity id and predicts all event attributes with this information. The third variant also enables each part of the event attribute prediction network to access attribute information of the next event. The available information is the whole event data except the ones currently predicted in this part of the network.
Anomalies in these systems are not detected by discrimination of the mean squared reproduction error but by an anomaly score. This score represents the cumulative probability of the predicted probability distribution greater than the probability of the given data. Nolle et al. (ibid.) further proposes multiple heuristics to derive the anomaly threshold of the final discriminator automatically.

It can be seen that anomaly detection systems based on distance discrimination of a reconstruction or prediction model are viable approaches in process mining. The proposed approach based on RNN networks has the benefit that it entails a model to solve the process prediction task as well. In addition, RNN models are more flexible towards dynamic input data length and can detect anomalies for each finished event. This enables the detection of anomalies in an online setting without long latency before the anomaly can be

detected.

The various systems provide multiple strategies to integrate event attributes into the classification models. Yet, none of the presented research integrates high dimensional event attributes in their models. Nevertheless, these systems will serve as inspiration for the method developed in this thesis to be adapted for sensor event attributes.

3.3. Sensor Data Processing

The main goal of this thesis is the incorporation of sensor data in process mining tasks. So far, sensor data has never been represented with neuronal networks for process prediction or process ming tasks. Therefore, this section will present how sensor data is commonly represented in neuronal networks in other disciplines.

Kammerer et al. (2019) presents a system to detect anomalies in sensors of a manufacturing process in two settings. The first setting describes the machine sensors of a pharma packing machine whereas the second setting details the environmental sensors of a 3D printing machine. For the packing machine, the data was preprocessed via a fast-Fourier transformation. Anomalies were detected by a Euclidean distance threshold or various machine learning methods.

Tanuska et al. (2021) proposed an FFN network to monitor bearings within a production line. Therefore, continuous temperature measures are fed into the network for a binary classification task.

Lee et al. (2017) reported the best results for the task of anomaly classification from multivariate sensor signals by a multi-layer CNN. Therefore, the receptive field of the CNN is slid along the time axis for the extraction of the faulty features.

van Wyk et al. (2020) also suggests the usage of a CNN to detect anomalies in real-time. However, in this system, the CNN is combined with a Kalman Filter and a χ^2 detector, a very popular technique in signal processing to read noisy data. In this so-called CNN-KF model, the data is first processed and classified by a CNN to detect anomalies on its own. All sensor readings that are classified as normal by the CNN gets processed by the Kalman model to further detect anomalies. This two-stage process shows that a combination of NN based methods with the traditional sensor processing technique can provide a symbiosis.

According to the literature survey of Mohammadi et al. (Fourthquarter 2018), most time-series Internet of Things (IoT) data are modelled by LSTM or other RNN networks.

Dong (2019) deployed a deep LSTM with a deep CNN in parallel to predict multiple sensor anomalies in aircrafts. The LSTM and CNN outputs are collectively used as input for an FFN. The whole model is learned as a classification task end-to-end.

Similar to the anomaly classification models presented in the previous section, Malhotra et al. (2016) proposes a reconstruction based anomaly detection model. The system is composed of an LSTM autoencoder model with a discriminator based on the euclidean distance. The threshold is computed by a maximum likelihood estimation with already known anomalous cases. The final model can reliably detect anomalies in four real-world datasets with input lengths up to 500 time-steps.

It can be seen that there exist various ways to represent sensor values in deep learning. The spectrum reaches from FFN to CNN, RNN models up to combinations of the methods with classical sensor processing techniques such as the Kalman filter or Fourier transformations. So far, none approach has managed to present itself as a standard which might also be caused by the fact that its suitability might depend on the properties of the sensor data (Solomakhina et al., 2014).

3.4. Process Mining Evaluation

The quality of machine learning models depends on the dataset they extrapolate from. Additionally, ML models based on neuronal networks require those datasets to have an appropriate size to be applicable.

In the process mining domain, there exist some widely-used datasets such as the helpdesk dataset (Verenich, 2016) or the BPI-challenge datasets (Dees and Dongen, 2016; Steeman, 2013; B. van Dongen, 2011, 2012, 2014, 2015, 2017, 2019; B. van Dongen and Borchert, 2018). These datasets are recorded from the real world, hence presenting authentic scenarios for research evaluation. Events within these datasets contain various attributes with numeric, temporal or categorical nature. Unfortunately, none of these events is annotated with IoT data. Since this thesis wants to showcase the use of IoT data for process mining, none of the traditional datasets can be used.

The combination of sensor data and process mining is becoming increasingly popular in the terms of Industrial IoT or smart manufacturing in the industry. High-Tech industries such as the semiconductor manufacturing (Dua and Graff, 2017) or the paper industry (Ranjan et al., 2019) provide data about their manufacturing including complex sensor data. However, the manufacturing processes contained in those datasets turn out to have no variance in their control flow. Therefore, these datasets are too simple to serve as a dataset for process mining.
The only public dataset with a complex control flow and a potential of time-series sensor data is the anonymized production data from *Bosch Production Line Performance* (2021). However, further analysis of the event attributes revealed that only short numeric sequences co-occur in the dataset. Considering the dataset is anonymized, the data points could not be categorized as time-series. Therefore the dataset is unsuitable for the evaluation of this research as well.
The scarce availability of public smart manufacturing datasets indicates that IoT is still very early in the industry adaption.

Another possibility to evaluate process mining tasks is the use of synthetic data by event log simulators. The effectiveness of synthetic event logs for process mining evaluation have been shown multiple times (Krajsic and Franczyk, 2021; Nolle et al., 2016). The simulator provides the advantage, that the reference model of the process is explicitly defined and all challenging characteristics of the event log can be controlled.

The GUI tool PLG2 by Burattin (2015) provides the possibility to simulate complex processes with sequential, parallel or iterative control flows. Furthermore, PLG2 supports the generation of multiperspective event streams and concept drifts. Event attributes can be categorical, temporal or numeric and can be defined as arbitrary complex. PLG2 even supports adding noise to the data. Skydanienko et al. (2018) proposes a tool with a similar scope of operation.

Sánchez Ferreres (2018) proposed to use natural language as an interface for a process simulator. The process model is extrapolated by NLP techniques before it is simulated.

Multiple simulation tools have been proposed on top of the Process Mining Framework (B. F. van Dongen et al., 2005). The tools have similar generative capabilities as PLG2 but generate distributions of event logs by varying the underlying models (Jouck and Benoît Depaire, 2016; Jouck and Benoit Depaire, 2019; Mitsyuk et al., 2017). Mitsyuk et al. (2017) generate the models based on a tree representation whereas Jouck and Benoit Depaire (2019) samples from a population of event logs.

However, none of the simulators provides the possibility to represent interdependencies between past event attributes and control flow. All event attributes in the generators are drawn by some function. The value of the event attributes will not influence the flow of the process afterwards. This constrains the impact and authenticity of attributes in synthetic multi-perspective event logs.

Since no available dataset provides an appropriate test case for process mining with sensor event attributes, a new dataset generator has been developed. The next chapter will present this generator and showcase how the limitations of the datasets and systems, presented in this section, has been overcome.

4. Synthetic Dataset Generator

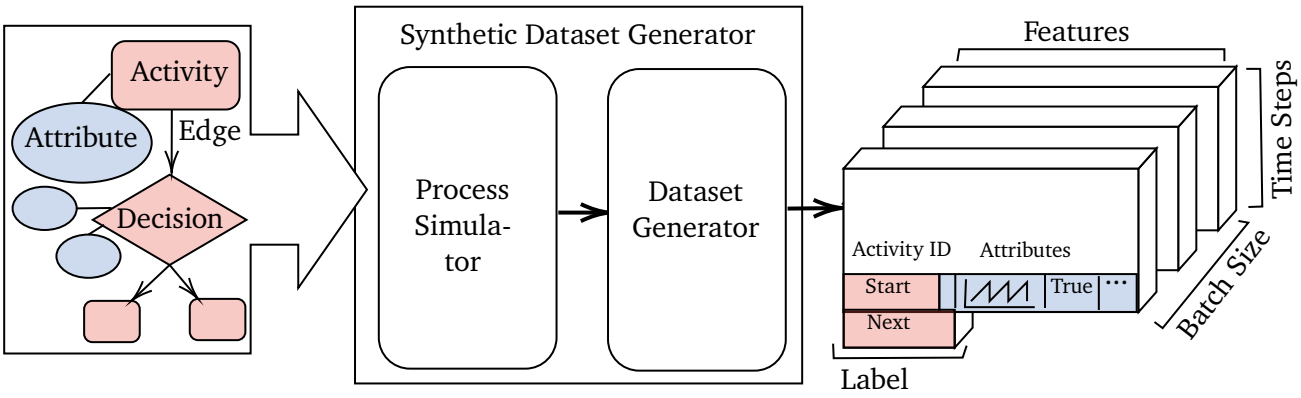


Figure 4.1.: Input and output of the Synthetic Dataset Generator with its sub-systems

This chapter explains the technical architecture of the software to generate the dataset, the research was conducted with. During the development of the system, it was crucial to be able to evaluate the influence of isolated variables. However, real datasets are very complex, often contain undocumented influences and are hard to customize. Synthetic datasets, on the other hand, can be generated to provide an ideal input for the desired experiment. The influences of the independent variable can therefore be attributed directly.

The *Synthetic Dataset Generator* is composed by two sub-systems, the *Process Simulator* and the *Dataset Generator*. The task of the *Process Simulator* is to generate an event log by simulating a defined process graph. The *Dataset Generator* takes the event log and converts it into the desired dataset for the experiment. The system is visualized in figure 4.1.
In the following, both systems will be presented in detail.

4.1. Process Simulator

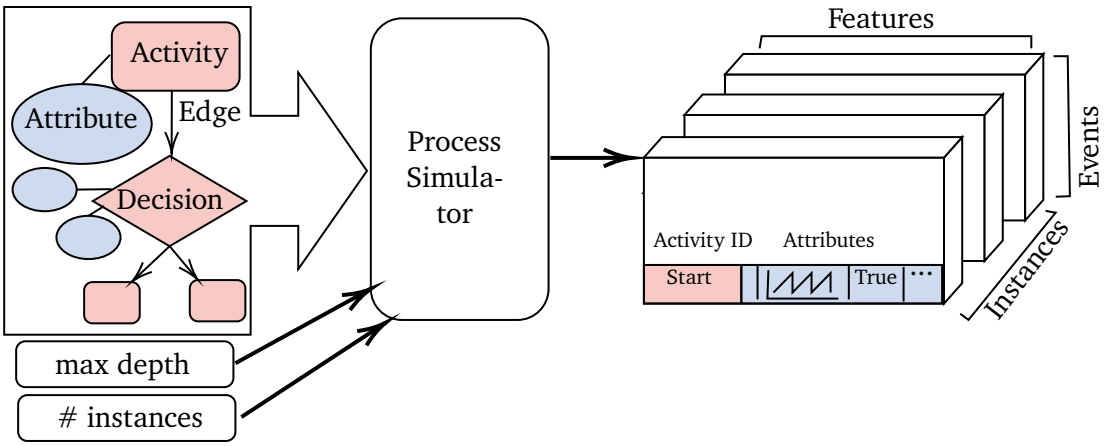


Figure 4.2.: Input and ouput of the Process Simulator as part of the synthetic data system.

Figure 4.2 shows the information which is processed in the simulation system. The Process Simulator is configured by the process graph, the number of instances to generate and the maximal instance depth. This parameter ensures that the number of events in one instance is maximal of this depth size. If this limit is passed, the instance is yielded even though it did not reach the end of the process graph. The instance length can be limited to avoid some very long instances in cyclic process graphs to which the whole batch need to be padded for the dataset generation.

The simulator yields the event log as a list of instances where each instance is composed of the event data. The activity id in the event data is represented with one embedding. All additional attributes need to be numeric since no additional embedding will be applied to the data.

As described in section 3.4 no process simulator so far has incorporated prior event attributes in the determination of the control flow. This simulation system will implement this feature as it is required to generate an appropriate dataset for this thesis. The next section will lay out how the control flow of the simulated process is determined.

4.1.1. Cotrol Flow Management

Definintion of the Process Graph

To define a process graph P in the simulator, one has to configure the edges E and vertices V of the desired direct process graph. The start and end nodes are already predefined.

$$P = (V, E, \phi_A, \phi_D)$$

Each *process step* $s \in V$, except the start and end node, can be a representation of an *activity* A or a *decision* D.

$$V = A \cap D$$

An *activity* ϕ_A is a step in the process graph that *results always in the same predetermined subsequent node*. Therefore, *only one edge* can originate from a *activity* and can not be directed onto itself.

$$\phi_A : E \to (x, y) | \forall x \in A \land y \in V : x \neq y \land (x \in A \nexists z \in A : (z, y) \in E)$$

On the other hand, a *decision* ϕ_D can point towards *numerous proceeding nodes*. In the simulator, a decision can be an arbitrarily complex function that returns the node id of the subsequent process step.

$$\phi_D : E \to (x, y) | \forall x \in D \land y \in V$$

Information about the execution or the result of the activity can often be captured by data. The system represents all activity data as *attributes* Ψ. Each activity $a \in A$ can be described by numerous attributes:

$$x_{1,a}, \ldots, x_{n,a} \in \Psi_a$$

Determination of the Control Flow

Decisions are the only varying factor of the control flow in process instances. The outcome of a decision is determined by an arbitrarily complex function from which the subsequent process step is going to be drawn. This decision function has access to the attribute values of activities before the decision. Thus, decisions might have to attribute independent outcome distributions but can also depend on their outcome on prior attributes.

In this thesis, a decision is either drawn solely from an internal probability distribution (*independent decision*) or the decision outcome depends entirely on prior attribute information (*conditional decision*).

$$independent\ decision: \qquad \phi_{D,ind}(x|x_1,\ldots,x_n) \Leftrightarrow \phi_{D,ind}(x) \qquad \forall x_1,\ldots,x_n \in \Psi$$

$$conditional\ decision: \qquad \phi_{D,cond}(x|x_1,\ldots,x_n) \Leftrightarrow y \qquad \forall x_1,\ldots,x_n \in \Psi$$

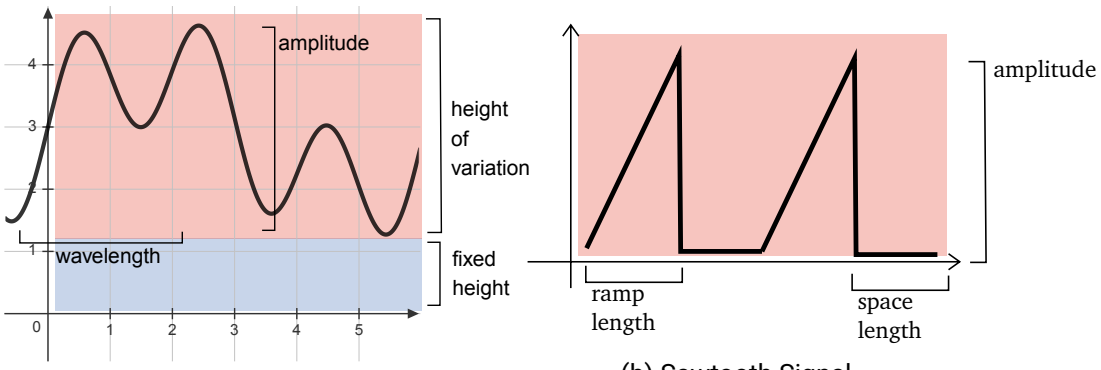
Attributes are represented by arbitrary complex functions from which the attribute information is going to be drawn. Each time the node is visited while the simulation, the function is executed and thus the attribute is updated. To have an authentic simulation, the attribute function usually depends on some kind of random sampling since no process step is executed the same way. Examples of attribute functions can be found in chapter 5.3.

Since this thesis is focused on the incorporation of attribute values during the flow prediction of processes, mostly conditional decisions are simulated.
It has to be noted, that the conditional decisions are only deterministic if one can perfectly marginalize over the attributes. This assumes a perfect interpretation end extrapolation of the important attribute information, the decision depends on. However, the noisiness and complexity of real-world data make this interpretation of attributes challenging.
A model to automate this challenge will be proposed in chapter 5.1.
In the next section, tools to simulate attribute data in the form of sensors will be presented.

4.1.2. Sensor Simulation

Since this research relies on sensor data, the simulator provides some tools to assist in the generation of the desired sensor attributes. The functions *perlin noise* and *spaced sawtooth* help to simulate the core sensor signal whereas the function *transition* help to add additional characteristics to the signal. Additionally, all implemented sensor anomalies, which will be presented in the next section, can be applied to the signal.

The simulated signal can be generated by an arbitrarily complex function. One of the provided signal simulators is the spaced sawtooth signal wave. It has a non-sinusoidal



(a) Perlin Noise

(b) Sawtooth Signal

Figure 4.3.: Parameter to configure the generated signal

waveform with upward ramps and a sharp drop of the signal followed by an option space before the next ramp start. The signal can be configured by the ramp length and amplitude as well as the length of the empty space between the signals.

The second provided signal simulator is the Perlin noise function by Perlin (1985). The function is popular for its pseudo-random appearance while all of its visual details remain in a similar size. With Perlin noise, it is possible to create a random signal while being able to readily control its properties. This helps to generate signals that simulate the property of the measuring sensor while showing some variance in its manifestation. The Noise Signal can be configured by a Parameter for the Amplitude, the Frequency and the number of octaves. The Frequency determines the reciprocal of the wavelength whereas the amplitude describes the maximal height of its crests. Octaves describe the amount of small detail in its variation. These are generated by adding generating the same noise with the half frequency and half amplitude to the signal in a recursive fashion. The octaves describe the recursion depth and result in fractal-like signal details. Furthermore, one can configure the height of the variation as well as the minimum value of the generated data. The height of the variation describes a range in which the generated signal is scaled into. Since the signal height, determined by the amplitude, is not guaranteed to be of the full amplitude size in the resulting sample, this step might be useful.

Most parameters for both signal generators can be observed in figure 4.3. These parameters to be configured to receive an authentic signal depending on the emulated circum-

stance. Once the signal looks authentic, the number of data points to yield during each emulation can be configured.

The *transition function* provides the possibility to alter the generated signal to fade in or out of the sample. This characteristic can be used to simulate sensor measurements with a starting or stopping phase e.g. if something needs to be heated up to a working temperature. It is implemented by multiplying the signal with the natural logarithm.

With the help of these tools, it is possible to generate an event log from a process model with sensor attributes and complex dependencies between the process flow and its event attributes. The next section will describe, how this event log is converted into a dataset with the desired experiment properties.

4.2. Dataset Generator

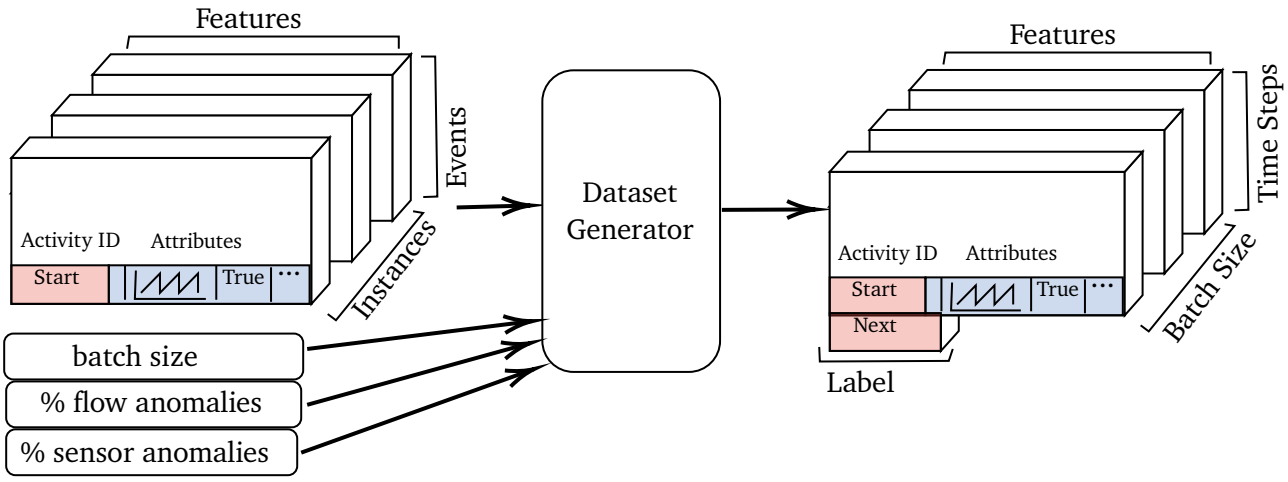


Figure 4.4.: Parameter to configure the generated signal

The *dataset generator* has two major roles, applying anomalies on the event log and shaping the log into a dataset for a prediction task. Figure 4.4 shows the input and output formats of the data and the dataset generator arguments. Due to the separation of the process simulator and dataset generator components the same simulated event log can

be converted into multiple datasets with different anomalies. Therefore, the influence of the anomalies can be measured more accurate since the underlying event log satisfies the ceteris paribus requirement.

Two kinds of anomalies can be applied to the event log, flow anomalies and sensor anomalies. Flow anomalies alter the process flow in the event log whereas sensor anomalies change event attributes with sensor data. The next two subsections will describe the anomalies in each class in greater detail. Anomalies are applied to the event log before the log is converted into the dataset format.

During the conversion of the event log into the dataset format, all process instances are converted into a next event prediction task by a sliding window function. The time step depth of the window is set to the length of the longest instance in the event log. All instances are zero-padded to this size.
At this step, it is possible to filter certain inputs and labels based on a subset of the process graph activities. The experiment can thereby be tailored to examine only certain aspects of the simulated process. For example, this filter enables the generation of a dataset where only events after decisions have to be predicted.
Afterwards, the generated dataset is shuffled and batched to the given desired batch size. This dataset provides the proper shape for the next event prediction task from the given event log.

4.2.1. Flow Anomalies

The probabilty for each case to be anomalous is configured by an argument. Each randomly selected instance can only be altered by one anomaly. All anomalies have an equally split chance to be applied. The system implements two anomalies:

Skip: *A event sequence of up to 3 events is deleted.*

Rework: *A event sequence of up to 3 events is duplicated.*

The anomalies are inspired by Nolle et al. (2019). As described in section 2.4, additional anomalies to alter the activity order or insert a random event has been proposed. Extending the system with these anomalies is an interesting task for future research work.

4.2.2. Sensor Anomalies

Section 2.4 also offers an overview of classification frameworks for sensor anomalies. This system implements all described sensor anomalies that don't need prior knowledge about the sensor like anomaly *impossible value*. The implemented sensor anomalies are:

Drift: *positive or negative gradual change in values to the signal*

Noise: *Values oscillate considerably*

Spike: *One Anomalous value is higher or lower than surrounding data*

Offset: *Values are wrong by some constant*

Spike Cluster: *Multiple spikes in a short period of time*

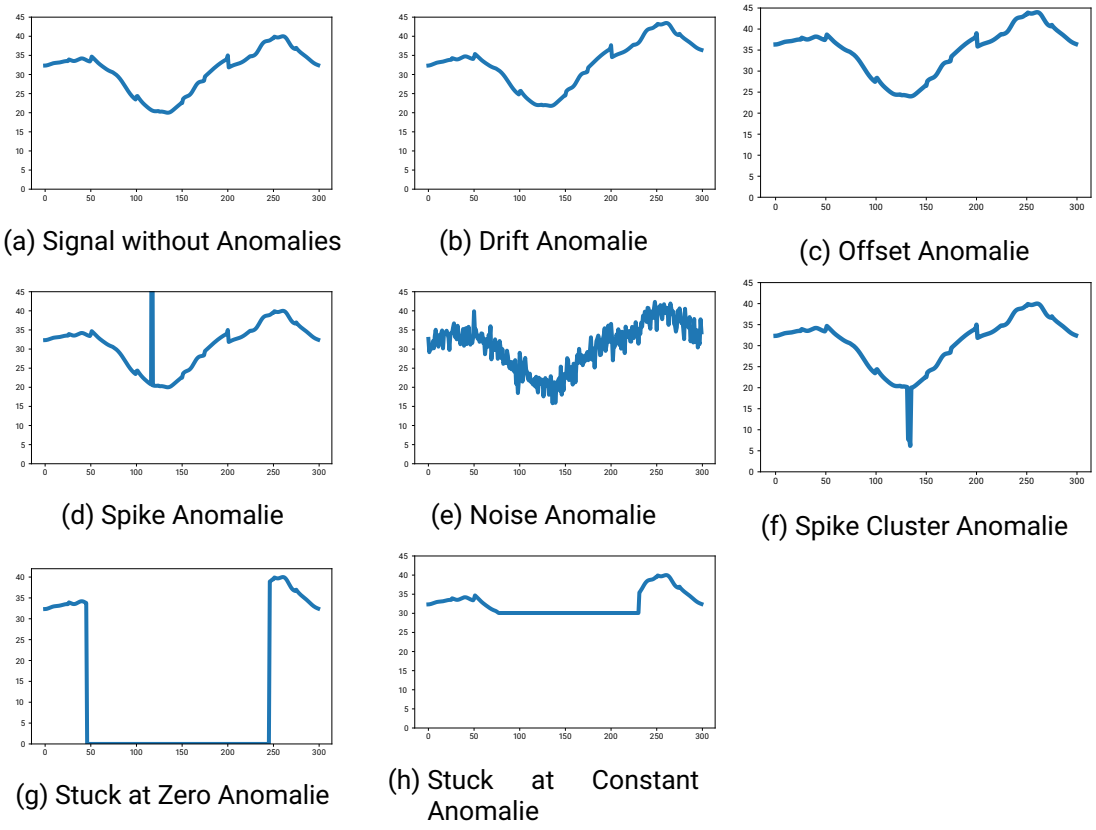
Stuck at Zero: *Sudden period of zero values*

Stuck at Constant: *Sudden constant period of the last value in a period with variance*

One argument determines the number of anomalies. A random function with the given chance applies the sensor anomaly to each chosen event attribute. The anomalies are selected by a uniform chance.
Each attribute will only be altered by one anomaly. It has to be noted, that that the original signal can already contain anomalies. This can be the case for sensor data that is drawn from a real-world dataset or the anomaly is part of the data simulation. In this case, multiple anomalies may be contained in the sensor data. The dataset generator will only be responsible for one anomaly per sensor data at most.

Table 4.5 shows a visualization of each anomaly applied to a given signal. The anomalies are implemented in the following logic:

The **drift** anomaly randomly selects if it is positive or negative. A gradual linear curve with, by default, a peak value of 10% of the maximal signal value is computed with the same length as the signal. This curve is added or subtracted by the signal, depending on whether the drift is positive or negative.



(a) Signal without Anomalies

(b) Drift Anomalie

(c) Offset Anomalie

(d) Spike Anomalie

(e) Noise Anomalie

(f) Spike Cluster Anomalie

(g) Stuck at Zero Anomalie

(h) Stuck at Constant Anomalie

Figure 4.5.: Generated Signal in comparison with each implemented anomalie applied individually. The signal 4.5a is generated with perlin noise.

The **noise** anomaly are randomly drawn values from a gaussian distribution with a mean of 0 added to the signal. The variance of the distribution is maximally 0.1 or 10% of the maximal variance in the input signal.

For the **spike** anomaly, a random spike index in the time series is chosen. Furthermore, an amplitude factor between 5 and 5.5 is chosen uniformly. With equal chance, the value at the chosen index is either divided or multiplied by the amplitude factor generating a spike in a positive or negative direction.

To achieve the **offset** anomaly, 10% of the maximal sensor value is added to all values in the time series.

The **spike cluster** anomaly chooses a range between 1% and 5% of the time series values to apply the anomaly. Again, it is randomly determined if the spike is positive or negative. Positive spike cluster multiplies the values in the cluster range by 150% to 180%. Negative spike cluster multiplies with a factor between 0.2 and 0.5 in the range.

The **stuck at zero** anomaly is generated by choosing two index values. The start index is randomly chosen in the first half of the sensor range whereas the end index is randomly drawn in the last half of the sensor range. All values between the selected start end index are set to zero.

The **stuck at constant** anomaly selects the start and end index similar to the *stuck at zero* anomaly. But the values of the whole selected range are set to the first value of the range.

The exact implementation details can be found in the appendix in section A.1. The following chapter will explain the core process mining system of this thesis and showcase its performance in experiments. For the experiment evaluation, two datasets are defined with the *synthetic dataset generator*, presented in this chapter. Hence, the applicability of this dataset generator to generate authentic datasets will be showcased.

5. Sensor-data assisted Process Mining

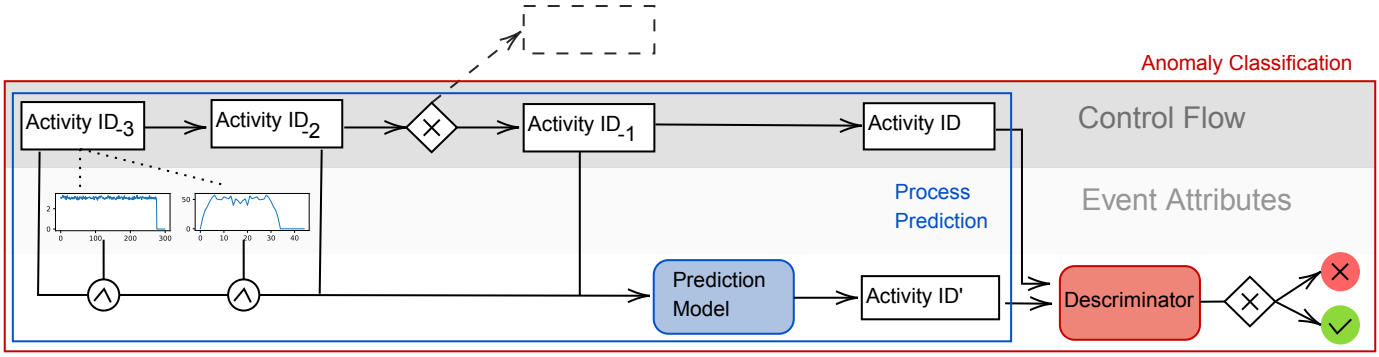


Figure 5.1.: System architecture of the process mining system system of this thesis. The components in the blue box are needed to use the system for a process prediction task whereas anomaly detection needs the components in the red box.

This chapter presents a process mining system that can be used effectively for process prediction and anomaly classification tasks while being able to extrapolate from sensor event attributes. To achieve this goal, an anomaly classification system based on a process prediction model has been developed. The fundamental concept is similar to the methods presented in section 3.2. The system is composed of multiple components from which only a subset is needed for the process prediction task.

The process prediction model is trained to predict the probability of the next event in a given event log. This model can be used for various process prediction tasks like forecasting or simulation.

To be able to detect anomalies, the process prediction model is extended with a discriminator component. At first prediction model forecasts how a process should be exe-

cuted. The discriminator then compares this prediction and the actual data to determine whether the execution was anomalous.

Figure 5.1 depicts all components of the presented system graphically. In the following, all main components *the process simulator, the prediction model* and *the discriminator* will be presented in greater detail.

5.1. Prediction Model

The performance of the prediction model is not only important for an effective process prediction but also influences the effectiveness of the anomaly classification system. A more effective process prediction enables to discriminate anomalous events more precise.

The literature review in section 3.1 has shown that RNN based models can effectively be applied to process mining tasks. RNN models are very flexible towards dynamic input and outputs by adapting to the desired temporal sequence size. Furthermore, they can be used for *process prediction* as well as *anomaly classification* tasks.

The proposed system architecture is inspired by the proposed system by Nolle et al. (2019). All event inputs are pre-processed and concatenated in an input pipeline and fed into an RNN network for the next event prediction task. The core RNN network is represented by a two-layer GRU with batch normalization Ioffe and Szegedy (2015) after each layer to avoid overfitting. The output of the RNN network is fed into a fully connected output layer (FN) to predict the one-hot encoded id of the next activity with a softmax activation function.

The input pipeline of the model distinguishes three different types of inputs: *event ids, event attributes* and sensor event attributes. Event ids and event attributes are not changed in the input pipeline. The input of each event id is expected to be one-hot encoded. Event attributes are considered to be numeric values with only one variable.
Sensor event attributes on the other hand are expected to be numeric input vectors and can have a very complex pre-processing setup. These different pre-processing possibilities for sensor attributes will be presented in the following. At the end of the input pipeline, all features are cast into a 32-bit float representation and concatenated to provide a uniform input for the RNN network. Figure 5.2 shows the system architecture of the described network.

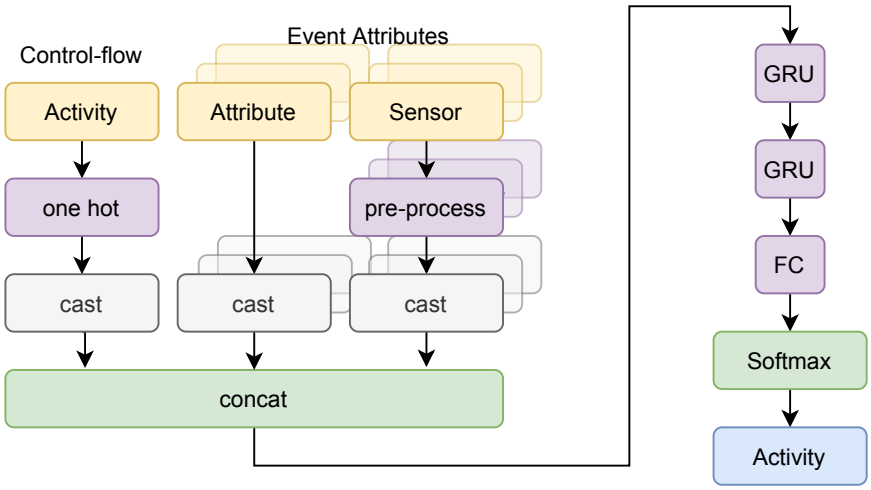
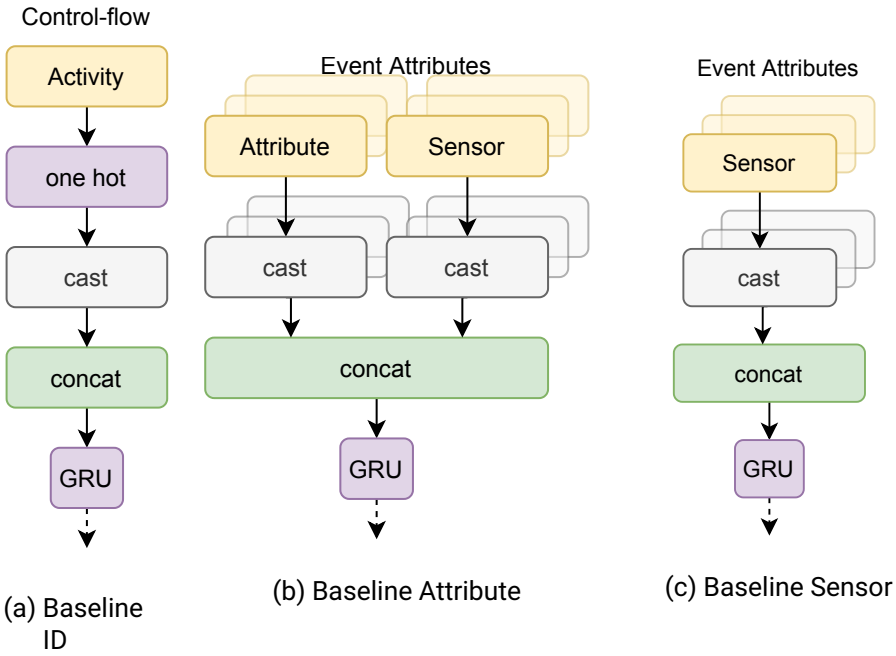


Figure 5.2.: Prediction Model Architecture for Next Event Prediction.

The described input pipeline is altered in three exceptional settings to compute a baseline performance. For each baseline, the prediction model deliberately disregards certain inputs.
The first baseline, *Baseline ID*, contains only information from the control flow of the process. This entails only event ids omits the event attribute and sensor event attribute values.
On the other hand, the *Baseline Attribute* can only predict based on attribute and sensor attribute values. This baseline has no access to the event ids of the corresponding attributes.
Finally, *Baseline Sensor* will only infer from sensor attributes and disregards the event ids and regular event attributes entirely. The sensor attributes in the baseline senor and baseline attribute are not altered in form of a pre-processing.

The baselines are designed to identify the influence of each input type and represent the model performance with restricted capabilities. Figure A.6 depicts the architecture of the input pipeline of the three baselines.



(a) Baseline ID

(b) Baseline Attribute

(c) Baseline Sensor

Figure 5.3.: Input pipeline architectures of the baseline experiment setups

The input vectors of the sensor attributes can be very large and entail anomalies. Therefore, deliberate pre-processing of the data might help the RNN network to extrapolate useful information for the process prediction task.

Section 3.3 presents various possibilities to model sensor values in deep learning with varying success in different domains. To detect the best strategy for the process mining tasks, a breadth-first search on possible methods is conducted. Therefore, multiple heterogeneous pre-processing configurations have been developed. The authors in section 3.3 found useful sensor representation with CNNs, RNNs, FFNs, Kalman and Fourier transformations.
Sensor representations based on Fourier transformations are disregarded in this thesis since such transformations require prior knowledge about the sensor spectrum. This thesis on the other hand aims to develop a system with end-to-end capability and no manual adaptation to the dataset. However, the preprocessing of the normalization is added since many authors reported them useful in section 3.1.

Figure 5.1 gives an overview of all implemented pre-processing layers. The figure exhibits

only the sensor attribute pre-processing part of the input pipeline. All pre-processing configurations always contain the event id and event attributes as well.

The *RNN* pre-processing is represented by a single layer GRU with 10 units and a hyperbolic tangent function as activation.

The *CNN* pre-processing is implemented with a single layer CNN with a fixed size of 10 filters and a kernel size of 5.

Two additional CNN based pre-processing methods are implemented as a deep network to learn a hierarchical sensor representation. Both networks are represented by three convolutional layers with a filter size of the previous input size divided by three and a kernel size of 3. The in the *Deep CNN* model, each convolutional layer is followed by a max-pooling layer whereas the *Deep CNN no Pool* pre-processing employs no pooling.

For the *Deep FFN* pre-processing, a two-layer feed-forward network is defined. Each layer contains 50 units and therefore results in an embedding with 50 dimensions.

The *Kalman* pre-processing is more complex. Each senor value is first filtered by a Kalman filter (Kalman, 1960). Afterwards, the Pearson correlation (Pearson, 1895) of the filtered signal with the original input is computed. Since a Kalman filter smoothes a signal from outliers and noise, a higher correlation indicates less noise and outliers in the signal. Only this correlation coefficient is then yielded from the pre-processing layer.

The *Normelization* pre-processing normalizes the signal into the [-1,1] range and appends the initial mean signal value. The initial sensor value dimension remains the same or is rather extended by the mean value. In this way, the absolute signal values can be retained to some extend while having a higher focus on the signal trend by the normalization.

Finally, an additional method by the name of *Linear* pre-processing is provided. This pre-processing is similar to the implementations of the attribute and sensor baselines in that it does not change the input. The signal is just passed through the layer into the RNN without any change of the initial signal.

All model variations are optimized with an Adam optimizer (Kingma and Ba, 2017) on categorical crossentropy loss. Additionally to the Momentum technique (Sutskever et al., 2013) used in Adam, the learning rate is reduced by 0.8 on a plateau of 2 epochs. The models are all trained on a duration of 20 epochs with a mini-batch size of 128. Exact model parameters and configurations for all variations of the proposed model can be found in the appendix at A.2.

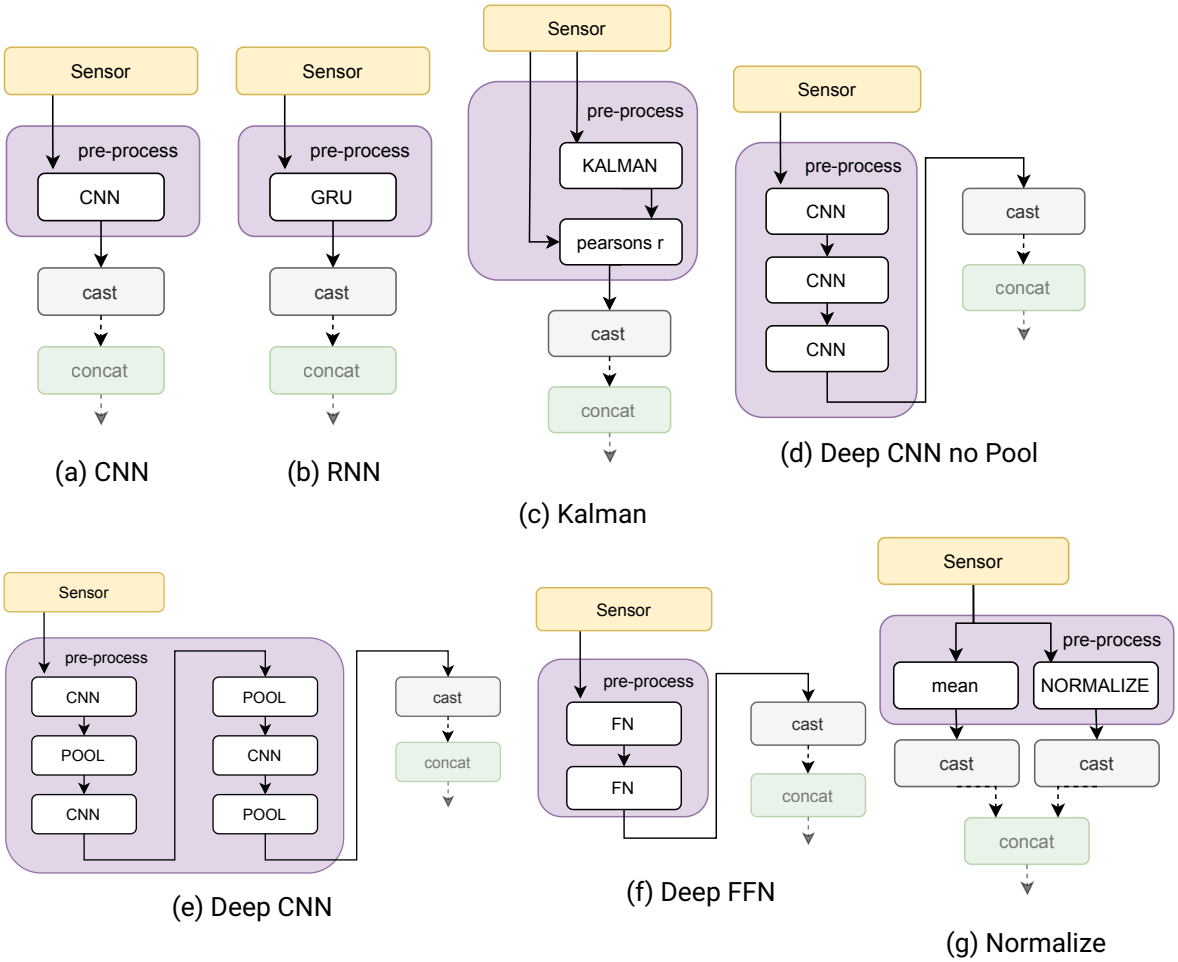


Table 5.1.: Configuration of the Model with different PreProcessing Layer

5.2. Discriminator

As described in section 3.2, a common way to detect anomalies in the process mining literature is discrimination based on a prediction or reconstruction error. The system in this thesis will follow this methodology and discriminate against the predictions of the prediction system presented in the previous section. It is assumed that anomalous next

events have a low probability in the predicted probability distribution of the next event. Therefore, they can be detected by measuring the vector distance between the probability distribution and the encoded given event. To measure this distance, the loss function of the mean squared error is employed.

$$\text{MSE}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} (y_i - \hat{y}_i)^2.$$

Finally, the predicted events need to be discriminated into anomalous and normal class predictions based on the measured distance. To achieve this, a threshold value τ is set as lower bound for an anomaly distance, classifying all events with higher distance as an anomaly. The determination of this τ can be done by a simple heuristic suggested by Krajsic and Franczyk (2021) and Nolle et al. (2018b). The τ is set to the average MSE of the given dataset, which can be scaled by some factor as suggested by Nolle et al. (2018b). For the anomaly classification experiment, no scaling factor is applied.

$$\tau = \frac{\sum_{i=1}^{n} \text{MSE}(y, \hat{y})}{n}$$

With this discriminator function, the prediction system can be extended for an anomaly classification task without the need for labelled anomalies.

5.3. Datasets

To serve as an appropriately challenging evaluation setup, three requirements for the dataset have been identified:

***R1**: Sensordata should vary and be authentic.*

***R2**: Sensordata within multiple activities should be combined to be interpreted.*

***R3**: The control flow of the process should influence the process outcome.*

Name	Process	#Gate-ways	#Train Instances	#Test Instances	#Train Events	#Test Events	Maximal Sequence Length
H6000	Hospital	4	6000	2000	50307	16821	10
H10000	Hospital	4	10000	2000	83885	16821	10
L6000	Laboratory	3	6000	2000	84702	28170	30
L10000	Laboratory	3	10000	2000	140519	28170	30

Table 5.2.: Details of all generated datasets for the experiments

R1 specifies that the applied model on the dataset measures the applicability to a sensor data scenario in the real world without overfitting one certain scenario. *R2* ensures that it is possible to represent a collective anomaly from the event attributes. *R3* makes sure, that an applied control flow anomaly as defined in subsection 4.2.1 also results in an anomaly from a logical perspective. If this requirement is not given, a rework anomaly might result in an event log that is logical without flaws. This requirement is often satisfied if the activities in the process change the physical environment of the process, for example during a manufacturing process.

As described in section 3.4, no existing dataset is suitable for the evaluation of this thesis. Hence, the process including its sensor values has been simulated to generate the *Laboratory Dataset*. The dataset contains process data from a 3D printing laboratory and will be described in detail in section 5.3.1.

To evaluate the system on real-world data as well, a second dataset, the *Hospital Dataset*, has been generated containing only real-world sensor attributes. This dataset showcases the flow of a patient with potential heart problems in a hospital. The dataset details are presented in section 5.3.2.

Table 5.2 gives an overview of the metadata of all generated datasets of the conducted experiments.

5.3.1. Laboratory Dataset

This simulated process describes the manufacturing process of a 3d printing laboratory utilizing lithography based 3d printing. The process is derived from the manufacturing of a hearing aid in audiology.

This specific process was chosen since the technology supplier in this industry are obligated to publish very precise process specifications in form of instructions (*Gebrauchsanweisungen* 2021; *Gebrauchsinformationen Audio* 2021).

The process starts with a customer meeting, where the ear is scanned and the hearing aid is modelled. This model will then be converted into a printable format and printed. Afterwards, the resin has to be washed away from the print and the print needs to be post-cured to get the desired properties. If these properties are met, the production might be completed. Otherwise, the product has to be reworked or printed again. But the process can also continue if a product has been produced that needs to be cast. In this case, only the cast shell has been produced so far. This shell has to be prepared for the casting process by the application of an additional cast isolation layer which additionally needs to be dried. After the actual casting process, the cast is polymerized in a pressure chamber to remove the remaining air. If all steps are met, the cast can be extracted from the shell without flaws and the process is completed. If the extraction did not work, the product has to be printed again.
Figure 5.4 shows the laboratory process in a diagram. The attribute values of the process are described in detail in table 5.3 and the decision details are documented in table 5.4.

The process involves multiple chemical reactions to create a product.
Since each consequential reaction has a very specified input to result in the desired outcome, *R2* is satisfied. *R3* is satisfied because the application of each step has a determined order to be successful. The process involves various machines with different parameters and functionalities, hence *R1* can be satisfied.

Attribute Name	**Description**	**Simulation**
product_type_0	Describes whether the product involves casting or not	Returns 0 indicating a moulding process in 50% of the cases, otherwise 1 indicating a casting process.
print_finished_0	Indicates if the print finished or was aborted	Returns 1 with a probability of 99% and indicate an aborted print with a 0 in 1% of the cases.
skipped_layer_0	Marks if layers were skipped while printing. This can be caused by unwanted artefacts inside the printer	Yields 0 with a random chance of 0.95 and 1 with a chance of 0.05.

sensor_print_temp_0	Expresses the temperature inside the 3D printer	At first, three-generation modes are drawn. Cold has a 3% chance, normal a 92% chance and hot a 5% chance. Each mode then draws a flexible heat variable from a uniform distribution. Cold and normal between 10-12 and hot between 13-20. Afterwards, the signal is generated by Perlin noise with an amplitude of 1, a wavelength of 400 and 5 octaves. Hot and normal have a fixed Perlin noise height of 30 and cold has a fixed height of 10. The variable heat serves as a variable height for the Perlin noise. Afterwards, the signal gets smoothed in at the beginning with the transition function before being returned.
sensor_load_curve_0	Signals the pull-off force per layer during the 3D print process	The load curve is generated by the sawtooth function with 30 periods and an amplitude of 20. With a chance of 5%, one of the ramps is anomalous and has an amplitude of 30-60. This generated signal is then distorted by adding the noise anomaly before being returned.

sensor_washing_0	Indicates the purity of the isopropanol during the washing	The isopropanol sensor for the washing is generated in three different modes. In 3% of the cases, short sensor values occur and are uniformly sampled between 50 and 170 seconds. Long washing can also occur with a probability of 3% with a wash duration time drawn between 530 and 620 seconds. Otherwise, the wash duration is considered normal and drawn between 200 and 500 seconds. The actual measured isopropanol level is drawn once from a uniform distribution between 0.85 and 1.0. A vector by the size of the selected wash duration filled with the drawn isopropanol level is then generated to simulate a measurement. Afterwards, a drift anomaly downwards with a strength of 10% is applied on the generated signal to simulate the decreasing isopropanol level during the wash process. The resulting sensor value is zero-padded to a vector with the size of the maximal generated wash duration of 620 to have the same output shape for all measurements. With a probability of 5%, a spike cluster anomaly is applied to the vector to simulate a broken sensor.

sensor_cure_energy_0	Measures the energy use during the cure activity	The cure curve is generated by uniformly selecting one energy value between 2 and 4 and duration between 250 and 300. The sensor measurement vector is then created by the size of the duration and filled with the energy value. This measurement is zero-padded by the maximum duration to the size of 300 and yielded.
sensor_dry_temp_0	This sensor signals the temperature during the dry process of the cast	With a probability of 5%, a short dry signal is generated. In this case, the duration is uniformly drawn between 120 and 540 seconds. Otherwise, the duration is between 600 and 1320 seconds. The sensor signal is then created with Perlin noise with the length of the duration. The fixed height of the Perlin signal is 30 with a variable height drawn uniformly between 5 and 25. The generated signal is then faded in and out with a bilateral transition function to simulate a heat up and cool down phase. Lastly, the signal is zero-padded to the maximal duration of 1320 seconds and returned.

sensor_poly_temp_0	Expresses the temperature during the polymerization of the cast	The polymerization duration is used for the sensor sensor_poly_temp_0 and sensor_pressure_0 since they monitor the same process in the same machine. In 2% of the cases, the polymerization duration is drawn between 10 and 24 seconds and is considered short. Otherwise, the duration is sampled between 25 and 45 seconds. For this duration, a signal is generated with Perlin noise. The noise has a fixed height and the variable height is drawn between 15 and 25. The signal is transitioned with a bilateral fading and zero-padded to the size of 45.
sensor_pressure_0	The sensor indicates the pressure during the cast polymerization	The polymerization pressure sensor is simulated in two modes. With a probability of 5% the sensor indicates that the chamber lid is not closed properly. In this case, the pressure is uniformly drawn from a distribution between 1 and 2.5. Otherwise, the pressure lies between 3 and 4 bar. The measured signal is created with a vector by the size of the previously generated polymerization duration and filled with the value of the drawn pressure. This signal is as well smoothed bilateraly with the transition function and zero padded to the maximum size of 45 variables.

Table 5.3.: Description and details of the simulation implementation of the attributes in the laboratory dataset.

Decision ID	Type	Involved Attributes	Description
LD1	conditional	sensor_washing_0	If the measured alcohol percentage signal contains cluster spikes, the next step should be *Fix Sensor*. Otherwise the decision should direct towards *Cure*.
LD2	conditional	sensor_cure_energy_0 product_type_0 print_finished_0 skipped_layer_0 sensor_print_temp_0 sensor_load_curve_0	If the surface area of sensor_cure_energy_0 is lower than 600 milli- joule, the process has to repeat the *Cure* activity. Given that the curing is flawless, the print might still have failed and need to be continued at *Print Prep* again by multiple conditions. The print failed if the maximum measured temperature of *sensor_print_temp_0* is higher than 40° and or lower than 20°. Furthermore, the print also failed if *print_finished_0* is false or *skipped_layer_0* is true. The print can also fail if *sensor_load_curve_0* indicates that the printer needed an abnormally high load force to separate the layers. Finally, the print failed if its wash duration measured in *sensor_washing_0* exceeds 500 seconds or is shorter than 200 seconds or has less than 80% alcohol purity. If all potential errors have been avoided, the process continues with the *Cast Isolation* activity if *product_type_0* indicates a cast product, otherwise, the process is completed and the *End* node reached.

LD3	conditional	sensor_dry_temp_0 sensor_poly_temp_0 sensor_pressure_0	The extraction fails, thus the process continues at *Print* if the dry temperature is lower than 40° or higher than 50° or the dry process is shorter than 1000 seconds. Furthermore, *sensor_pressure_0* should indicate a polymerization pressure of at least 3 bar and *sensor_poly_temp_0* should indicate a temperature above 45° for at least 25 seconds. If these requirements are met, the process completes after the extraction at node *End*.

Table 5.4.: Description of the decisions in the laboratory dataset. The column *Type* refers to the decision types defined in section 4.1.1.

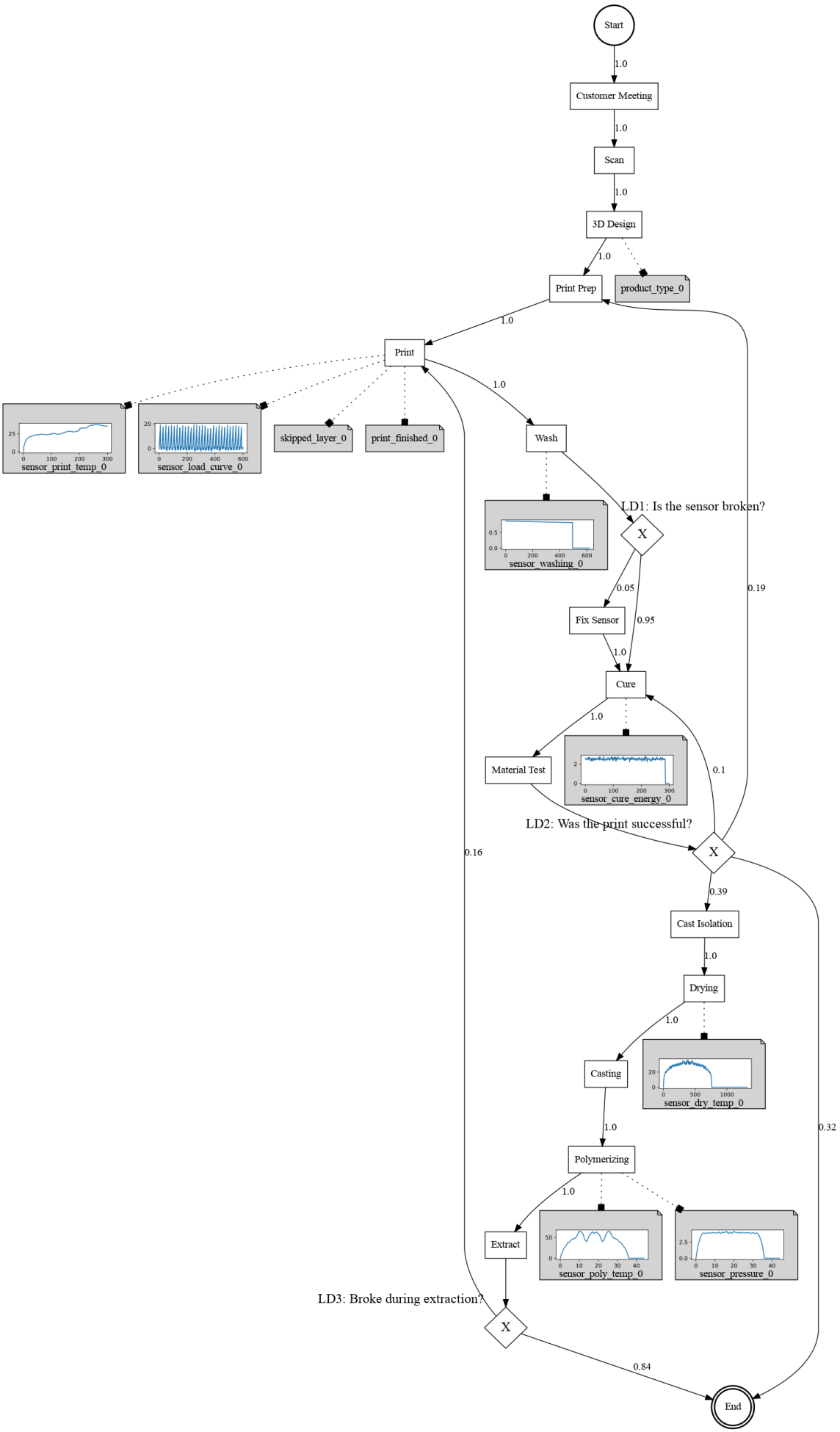


Figure 5.4.: Graphical representation of the laboratory business process model in BPMN notation (White, 2004). The numbers on the edges describe the conditional probability of the occurrence of the edge computed on 100000 events.

5.3.2. Hospital Dataset

This simulated process generates a dataset about the flow of a patient inside a hospital while being analyzed for heart problems. The simulator contains two different real-world sensor data. An Electrocardiography (ECG) Sensor, which collects electrical signals generated by the heart and an Arterial Blood Pressure (ABP) Sensor.
The ECG data originate from the MIT-BIH Arrhythmia Database Moody and Mark, 2001. The data is divided into train/dev/test data in which the test data is sampled from unrelated patients as the dev and training data origins. The original Dataset distinguishes four different anomaly classes and a normal class, but this research only distinguishes by abnormal and normal ECG readings.
The ABP sensor data originates from the MIMIC database (Saeed et al., 2002) and is preprocessed by Bote-Curiel et al. (2019). Since the ABP measurements are very long, the data was further divided into chunks of size 1000 and randomly shuffled. It has been made sure that the ABP datasets for train/dev/test setups all originate from different patients. All attributes in this dataset have been further described in table 5.5.

The Hospital process begins at the reception. If the patient has already been in the hospital, the data from the previous analysis are retrieved. Otherwise only a new patient record is created containing the patient age. Afterwards, a Nurse checks if the patient record indicates previous heart problems. In this case, the patient is directly transferred to the *Triage* activity. Otherwise, the patient has to wait. During the *Triage*, new ECG and ABP measurements are taken. If the patient shows no vital issues, he will be discharged. Otherwise, the patient gets a prescribed therapy and will be dealt with within the appropriate activity.

Since the dataset uses multiple real-world sensor data, *R1* can be satisfied. *R2* is satisfied since the ABP sensor values have to be combined to detect hypertension. *R3* might be mostly satisfied. The only activity which might be changed in the control flow without influencing the process outcome is the *Waiting* activity. This needs to be considered during the evaluation. All other activities and decisions satisfy *R3*.

Attribute Name	Description	Simulation
patient_age_0	Indicates the current patient age	The age can be drawn from two normal distributions. The old distribution has a mean of 80 with a standard deviation of 20 whereas the young distribution has a mean of 35. The age is dawn by 90% from the old distribution and 10% of the patient age from the young distribution.
sensor_abp_history_0	Contains an ABP measurement which is retrieved from the patient file.	Draws the next ABP measurement from the dataset and returns it.
sensor_ecg_history_0	Contains an ECG measurement which is retrieved from the patient file. The sensor might signal an anomalous heart function.	With a 20% chance, the ECG value is dawn from the abnormal ECG dataset, for the resulting 80% the sample is drawn from the normal dataset and returned.
sensor_ecg_0	Contains an ECG measurement which is newly measured. The sensor might signal an anomalous heart function.	With a 20% chance, the ECG value is dawn from the abnormal ECG dataset, otherwise, the sample is drawn from the normal dataset and returned.
sensor_abp_0	Contains a newly measured ABP sensor value.	Draws the next ABP measurement from the dataset and returns it.

Table 5.5.: Description and details of the simulation implementation of the attributes in the hospital dataset.

Decision ID	Type	Involved Attributes	Description
HD1	independent		With a 50/50% chance randomly decides weather patient information will be retrieved or not.
HD2	conditional	sensor_ecg_history_0	If the attribute *sensor_ecg_history_0* indicates a anomalous heart function, the decision directs to the *Triage* activity. Otherwise, the process continues with the *Waiting* activity.
HD3	conditional	sensor_ecg_0 sensor_abp_history_0 sensor_abp_0	If *sensor_ecg_0* indicates no anomalies and the abp sensors does not show any increase during the measurements, the patient is send to be discharged. Otherwise, the patient will be sent to get a therapy prescribed.
HD4	conditional	patient_age_0 sensor_ecg_0 sensor_abp_history_0 sensor_abp_0	If the doctor finds anomalies in the ECG data and the patient is young, he is directly transferred to the *Emergency Department*. Anomalies in ECG data in old patients are treated in the normal *Clinic*. If no anomalies are present in the ECG data but the ABP values have risen by 10% compared to the measurements in the patient record, the patient is prescribed with hypertension.

Table 5.6.: Description of the decisions in the hospital dataset. The column *Type* refers to the decision types defined in section 4.1.1.

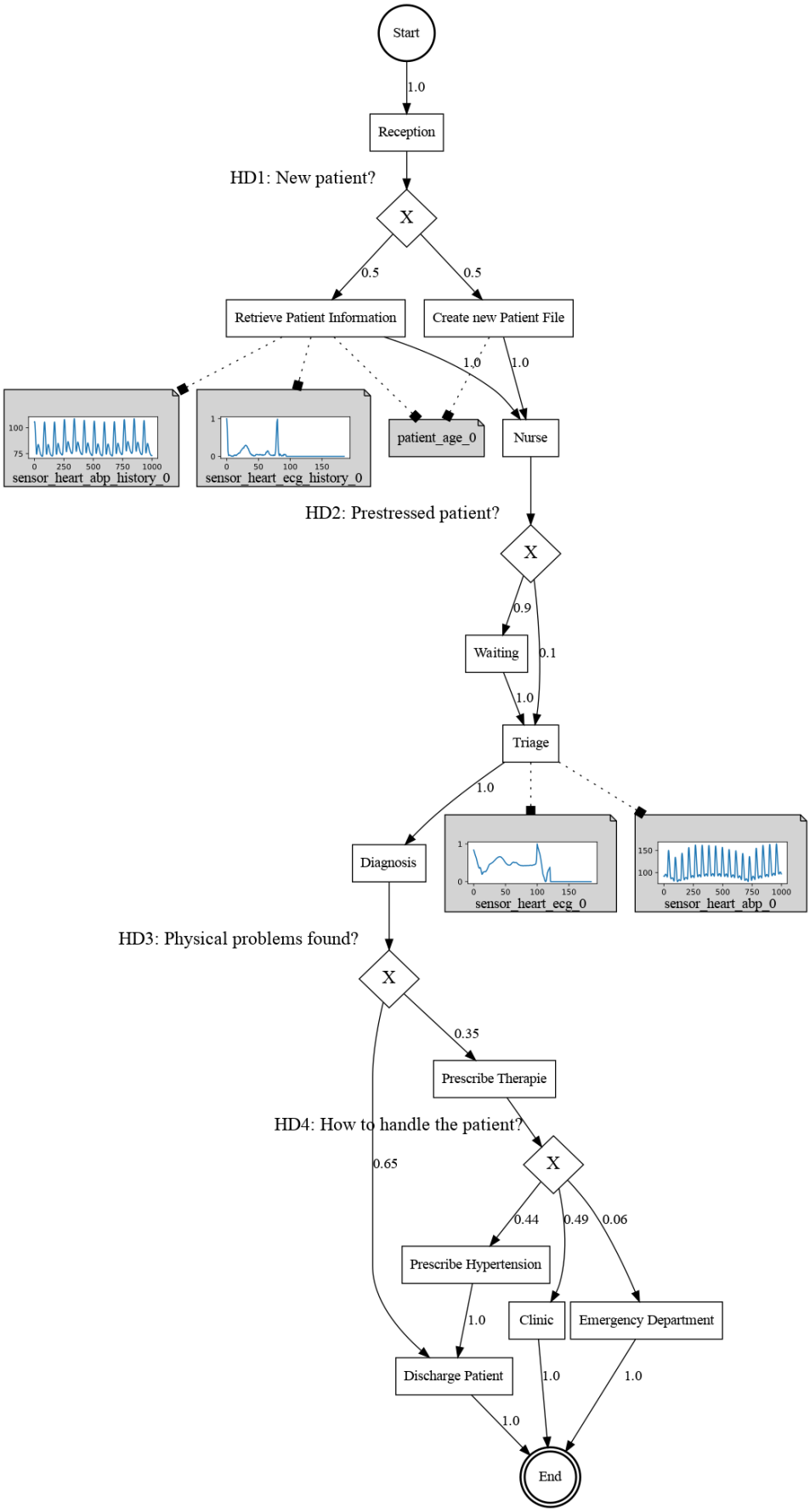


Figure 5.5.: Graphical representation of the hospital process model in BPMN notation (White, 2004). The numbers on the edges describe the conditional probability of the occurance of the edge computed on 100000 events.

5.4. Metrics

All binary classification experiments in this thesis are evaluated with the common metrics, *accuracy, precision, recall* and the F_1 statistic. *tp* refers to the true positive prediction, *fp* describes the false positive prediction whereas *fn* counts the false negative predictions.

$$\text{precision} = \frac{tp}{tp + fp}$$

$$\text{recall} = \frac{tp}{tp + fn}$$

$$F_1 = 2\frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

The precision measures the classifier ability to label a negative sample not as positive. Recall, on the other hand, represents how well a classifier detected all positive samples. The F_1 sore combines both measures by a harmonic mean. If the F_1 is 1, the classifier is perfect whereas a 0 indicates the worst possible classifier. The *accuracy* determines the percentage of correct predictions in the test setup.

$$\texttt{accuracy}(y, \hat{y}) = \frac{1}{n_{\text{samples}}} \sum_{i=0}^{n_{\text{samples}}-1} 1(\hat{y}_i = y_i)$$

Accuracy scores don't take the class distribution of the dataset into account and are therefore not as indicative for the evaluation of multiclass classification. However, the process prediction task of this thesis is a multiclass prediction problem and will therefore be evaluated with metrics adapted to this goal. The evaluation should account for the accuracy that would be generated simply by chance for imbalanced target classes. Therefore, the Kappa statistic (also known as Cohen's Kappa) will be additionally computed. This metric was invented to assess the agreement between two ratings (Cohen, 1960).

$$\texttt{kappa}\ \kappa = \frac{O - E}{1 - E}$$

O represents the observed accuracy. The expected accuracy based on the marginal totals is *E*. This statistic can result in values between -1 and 1 where 0 indicates no agreement

between the observation and the prediction. A κ of 1 on the other hand indicates a perfect match between the predicted and observed classes.

The *precision, recall* and the F_1 statistic are also adapted for a multiclass prediction to account for a class imbalance. y_l is the subset of predicted $(sample, label)$ pairs with the label l. $\hat{y}_l$ are the true $(sample, label)$ pairs with the label l. L represents the set of all classes.

$$\text{precision} = \frac{1}{\sum_{l \in L} |\hat{y}_l|} \sum_{l \in L} |\hat{y}_l| \, P(y_l, \hat{y}_l)$$

$$\text{recall} = \frac{1}{\sum_{l \in L} |\hat{y}_l|} \sum_{l \in L} |\hat{y}_l| \, R(y_l, \hat{y}_l)$$

$$F_1 = \frac{1}{\sum_{l \in L} |\hat{y}_l|} \sum_{l \in L} |\hat{y}_l| \, F_1(y_l, \hat{y}_l)$$

The statistics are calculated for each label and averaged out by the weight of their class. This weighting can result in an F_1 score outside of the range of the precision and recall measures.

5.5. Process Prediction

The following chapter will document the performance of the process prediction model as described in section 5.1 on the generated datasets presented in section 5.3. The model varies in the configuration of the sensor pre-processing or by the input scope for the three baseline configurations. All models are evaluated on the dataset L6000 and H6000 without anomalies, with 30% sensor anomalies, with 30% flow anomalies and with 30% sensor and flow anomalies. The scatterplot matrix in figure 5.6 depicts all experiment results. The kappa performance is in the centre of the evaluation since the process prediction task has no preference about the specificity or sensitivity of certain classes.
It can be seen that independent of the training data anomalies, the sensor baseline can infer the least information about the control flow. This can be improved if all attributes are incorporated with the baseline attribute for both datasets. Finally, the baseline id manages to predict the control flow most accurately among all baselines. All precise results of the experiments with their confusion matrices can be found in the appendix in section A.3.

These confusion matrices in figure A.12 depict the expected shortcoming of the attribute and sensor baseline of being unable to predict certain simple activity sequences that do not contain attributes. This is the case for the predicion of *3d_design* in the laboratory process or *hypertension* and *clinic* in the hospital process. Furthermore, the confusion matrices confirm the expectancy the ID baseline model mispredicts the activities that are referenced by a decision. These activities are *retrieve, create, wait, triage, prescribe, discharge, hypertension, clinic* and *emergency* in the Hospital Process. In the Laboratory Dataset, the activities after a decisions are *fix, cure, cast_isolation, print_prep, print* and *end*. On the other hand, the ID baseline model can learn trivial process flow segments which contain no decisions very well. It can be seen that the baseline ID even manages to predict L6000 with an Accuracy of 92.16% and H6000 with 86.93% Accuracy on the dataset without anomalies.
This indicates that both datasets have a large proportion of trivial process flows.

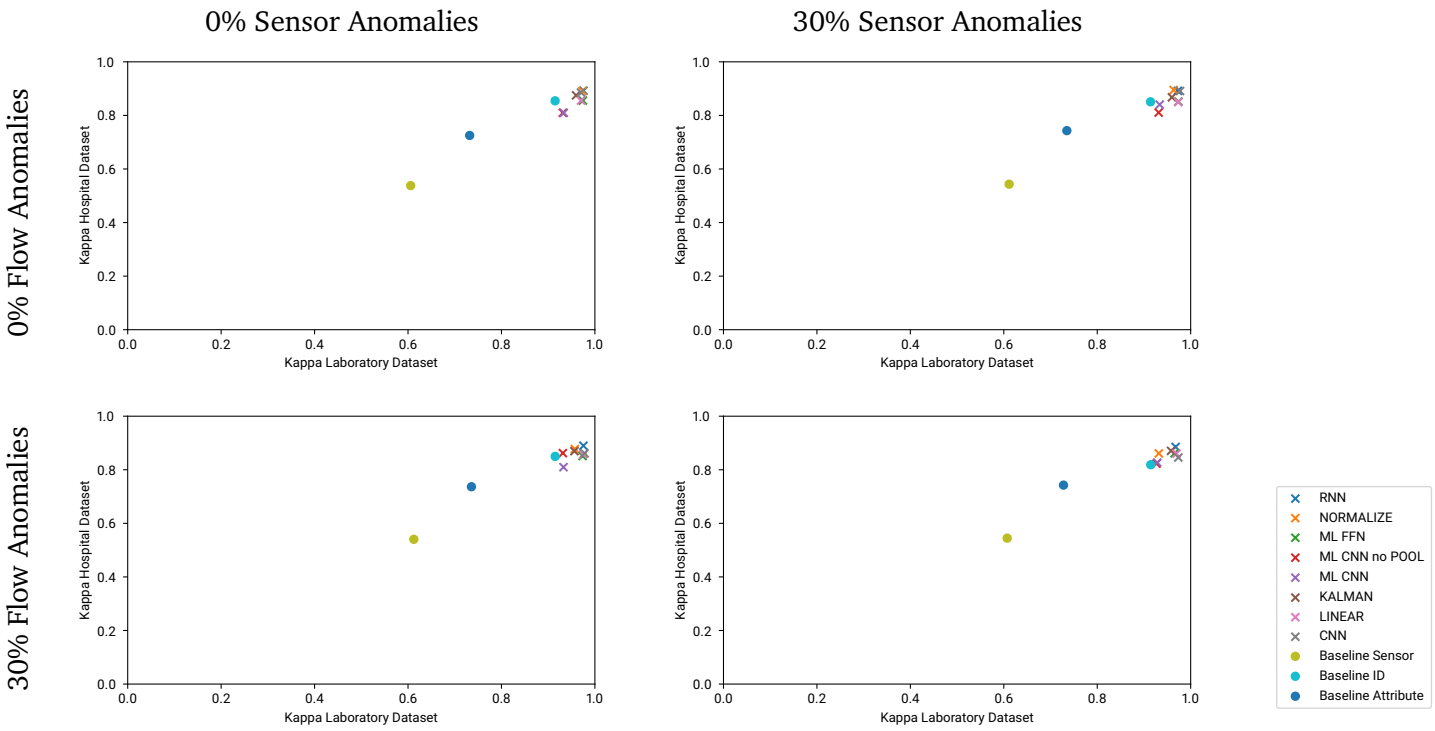


Figure 5.6.: Scatterplot matrix of the Kappa performance of various model configurations in comparison with the baseline on H6000 and L6000 with the annotated anomalies.

Figure 5.6 also reveals that the prediction model is very robust to anomalies in the training data. It is expected to see the influence of the sensor anomalies most isolated in the sensor baseline whereas the flow anomalies are expected to change the id baseline the most. Both baselines show no substantial loss in performance which can not be attributed to e.g. a different random initialization of the parameter.

To discard the influence of the trivial events, the process model is also evaluated to predict only events after a decision. By this test data filter, the performance of the prediction model on the difficult prediction tasks can be observed in isolation. Figure 5.7 shows a scatterplot matrix of the model performance on the test data containing only decision events with all anomaly combinations.

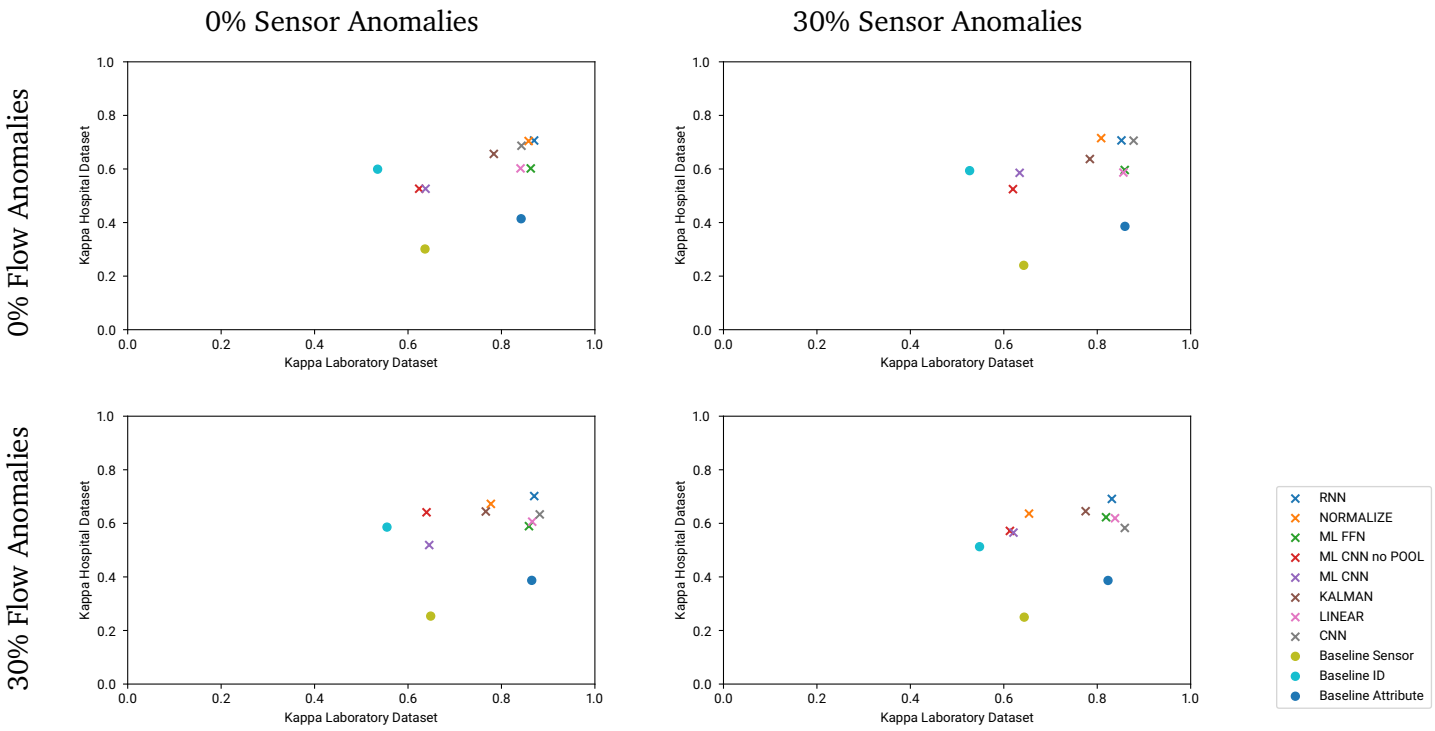


Figure 5.7.: Scatterplot matrix of the Kappa performance of various model configurations in comparison with the baseline on H6000 and L6000 decision events with the annotated anomalies.

Figure 5.7 indicate that most models with sensor attribute pre-processing perform on par to the attribute baseline for L6000 and excel on the dataset H6000. On the H6000 dataset, the baseline-id shows to be more indicative than the attribute baseline. Furthermore can be noticed that some pre-processing setups are more robust to anomalies than others. The *normalization* decreases its performance with each added anomaly whereas the *rnn, cnn, ml ffn* and the *linear* model present a relatively stable performance. In all experiments, the model with *rnn* or *cnn* sensor preprocessing manage to have leading performance.
The *CNN* model appears to be more susceptible to flow anomalies in the hospital dataset compared with the *RNN* model. Sensor data in the hospital dataset are sampled from real data and not normalized or aligned for the different timings of the measurements. This poses a challenge to the CNN since it infers from the spatial characteristic of the

data. Additional data entropy might make the CNN representations hard to learn for this dataset.
Both deep-convolutional models, on the other hand, show results with inferior scores to all prediction models and even undermatch the baselines for the experiment without anomalies on the decision events. This weak performance is also visible in the experiments in figure 5.6 against all test events. Since these models are deeper than most other models and with more parameters, the models might have under fitted the data indicating too little training data for the model complexity. The experiments have been repeated with the 10000 process instances datasets, H10000 and L10000, without anomalies and with 30% sensor and 30% flow anomalies. The results for all events and filtered for the decision events can be found in figure 5.8.

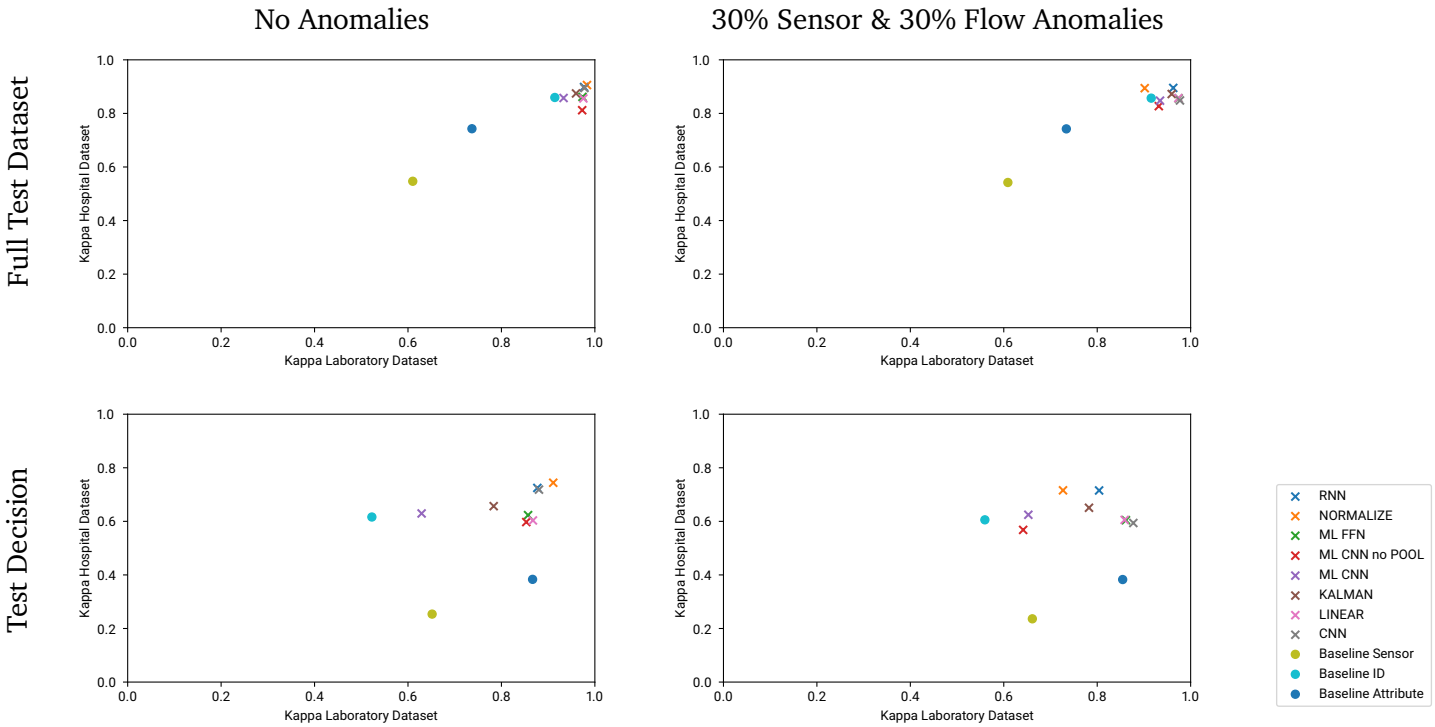


Figure 5.8.: Scatterplot matrix of the Kappa performance of various model configurations in comparison with the baseline on H10000 and L10000 as well as on their decision events with the annotated anomalies.

Again, the deep-convolutional models show to model the process the least well even on

the 10000 instances dataset. The *multi layer cnn with no pooling* model has even a weaker performance than the *baseline-id* on H10000 with anomalies. A similar weak performance can be observed by the *linear* model and the *multi layer FFN* sensor pre-processing model on the hospital dataset. The two models at least present themself as robust against added anomalies in the dataset. From these observations can be concluded that deeper model architectures are not preferable in this experimental setup.

The *normalization* model shows a leading performance on experiments with 10000 and 6000 instances if no anomalies are in the dataset. However, the model performance decreases significantly if the training data contain anomalies. The performance loss manifests itself the most for the laboratory dataset. The scatterplot matricies in figure **??** and figure **??** reveal that the model mispredict events of one activity often wrong. The normalization creates a less robust process model than other models modelling a modicum of the wrong process flows with a high probability from anomalous training data.

Next to the *normalization*, the *CNN* and *RNN* model supports the high-performance results of the 6000 instance experiments on the 10000 instances as well. Without anomalies, the *CNN* and *RNN* models have very similar performance for the hospital and laboratory dataset. The experiments with anomalies reveal that the *CNN* loses accuracy mostly for the hospital dataset whereas the *RNN* performs less well on the laboratory dataset. Both models show robust prediction accuracies for the opposite dataset.

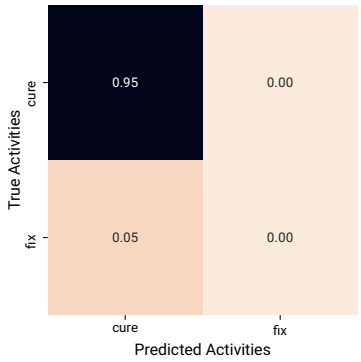
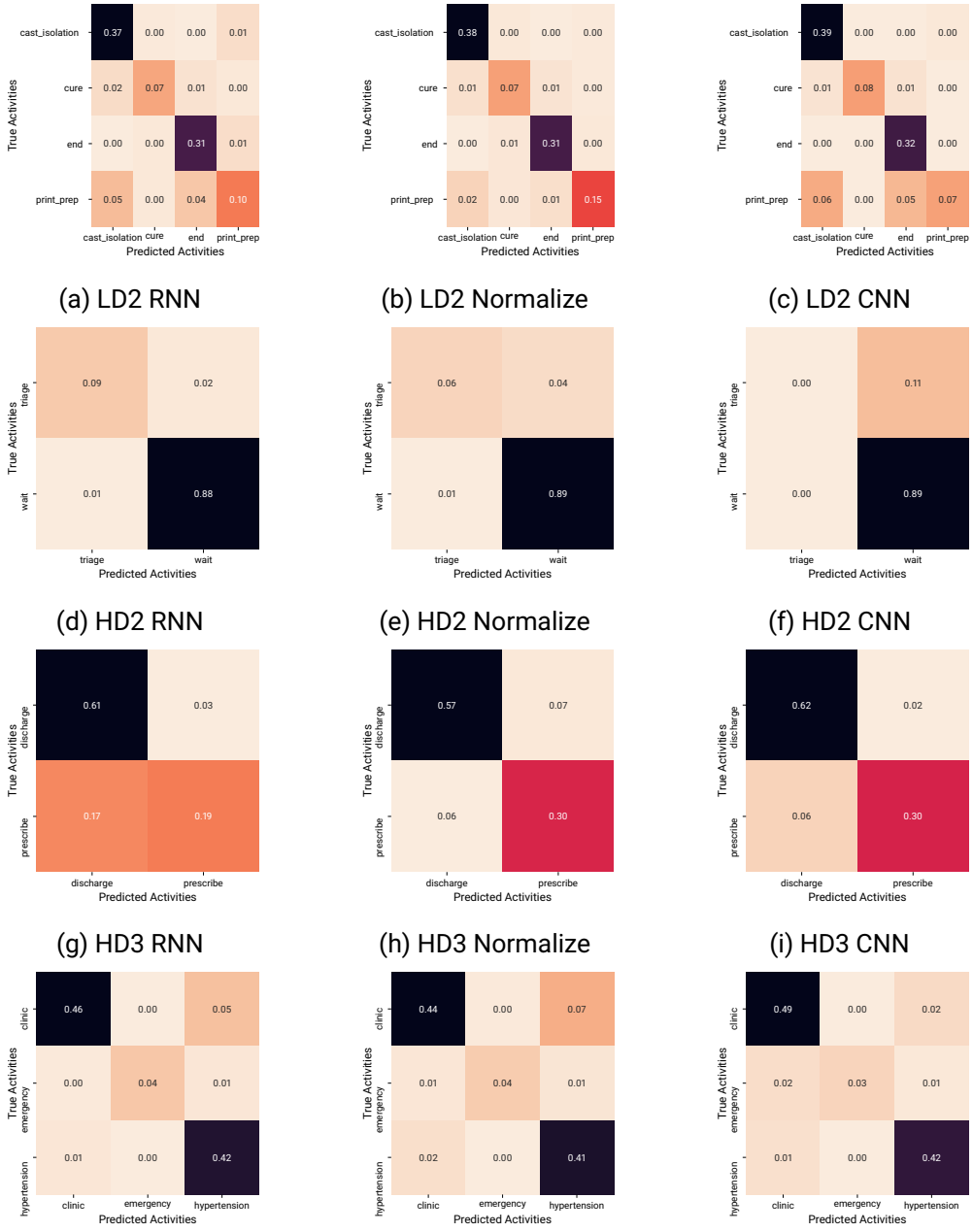


Figure 5.9.: Confusion Matrix of decision LD1 by the kalman model, trained on L10000 without anomalies.

It can also be seen that the model with access to the Pearson correlation of the Kalman

filtered signal (*kalman*) also improves the prediction of the model compared to the *baseline id*. However, the *kalman* model inputs all non-sensor attributes in addition to the Kalman value as well. Therefore, the performance gain compared to *baseline id* can not be fully attributed to the Kalman value.
In comparison with the *baseline attribute*, Kalman models the decisions of the laboratory dataset worse. It has to be noted that the Kalman model infers insights from sensor data only by the Pearson correlation value. The *baseline attribute* on the other hand has access to the unmasked sensor value and can therefore infer information that is not contained in the Kalman value.
As described in section 5.1, the Kalman filter intends to detect how many sensor anomalies are in the dataset. Decision LD1 requires the model to detect anomalies in sensor data to correctly predict the decision outcome. However, the Kalman filter model does not manage to detect anomalies for this decision as one can see in figure 5.9. If the anomaly in the sensor value is detected, the decision outcome of LD1 is *fix* which has never been predicted correctly.

On the other hand, the Kalman filter manages to improve the prediction performance against both baselines for the hospital datasets. This is an important outcome since the hospital dataset contains only real-world sensor data with natural occurring anomalies. The Kalman filter value might therefore be a useful feature if the dataset contains various anomalies. This insight is supported by the robust performance of the Kalman filter for all datasets with artificial anomalies.



(a) LD2 RNN (b) LD2 Normalize (c) LD2 CNN

(d) HD2 RNN (e) HD2 Normalize (f) HD2 CNN

(g) HD3 RNN (h) HD3 Normalize (i) HD3 CNN

(j) HD4 RNN (k) HD4 Normalize (l) HD4 CNN

Table 5.7.: Confusion matricies of LD2, HD2, HD3 and HD4 decision for the normalization, single layer cnn and rnn models on 10000 instances without anomalies.

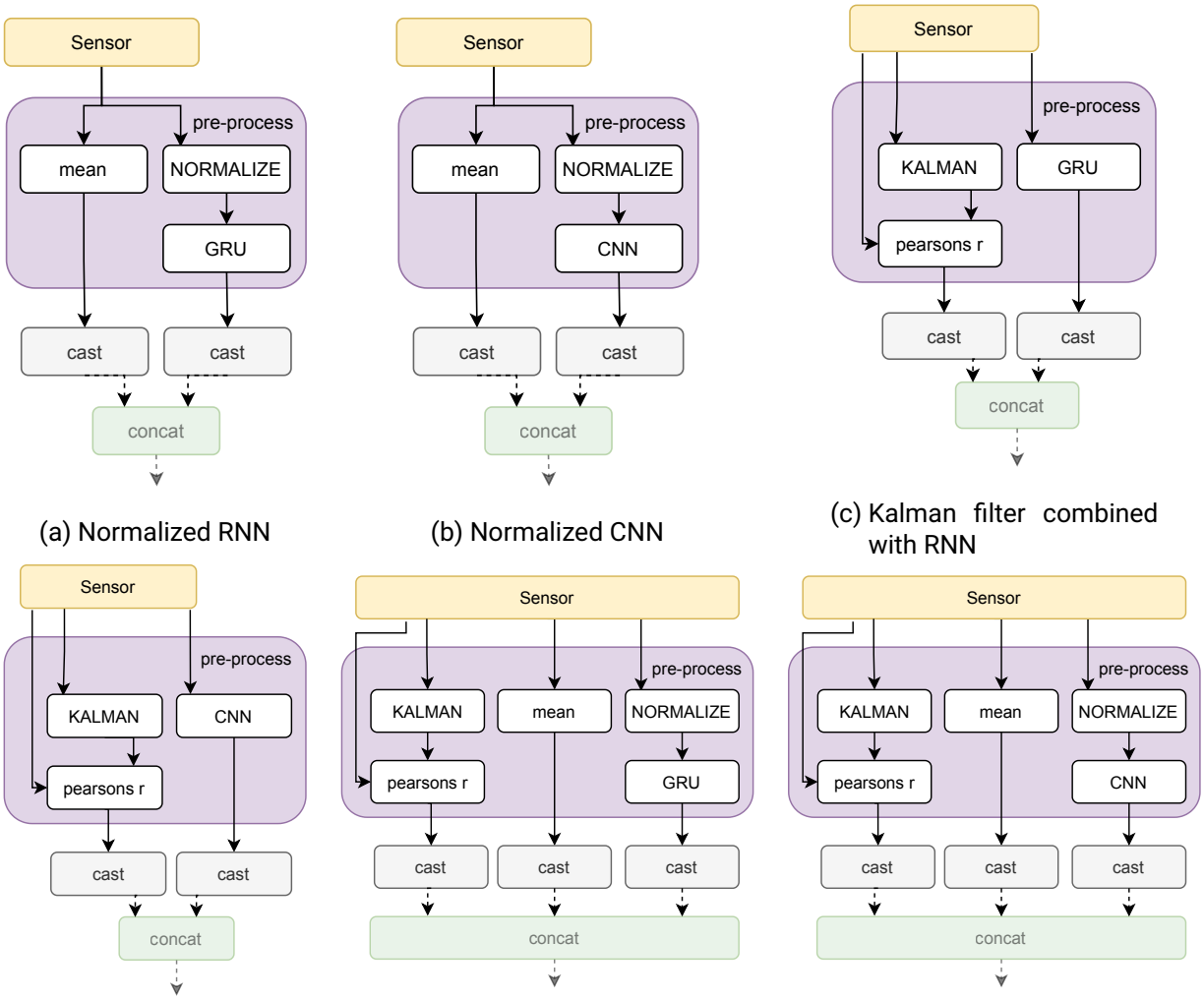
A comparison of the confusion matrices of the well-performing models in the appendix in section A.3 reveals that the models vary in their prediction strength and weaknesses. Confusion matricies for the most indicative decisions can be found in table 5.7 for the *CNN*, *RNN* and the *Normalization* model. The normalization manages to predict LD2 and HD3 more precise than the CNN or RNN architecture. On the other hand, the RNN model outperforms the normalization for HD2. The CNN is most precise for HD4.

It might be possible to use the opposed strength of the models to improve the prediction by building a symbiosis. The next chapter will present model variations which aim to combine the analysed model characteristics of this chapter and evaluate those on the same process prediction tasks.

5.6. Combined PreProcessing for Process Prediction

Since the *CNN* and *RNN* models showed the best overall performance in the previous experiments, this section tries to enhance them. The previous section also revealed that the *Normalization* model has opposite prediction strength and weaknesses to the *CNN* and *RNN*. Therefore, a combination of these pre-processing architectures suggests itself. Technically, it is possible to combine the normalization with the CNN or RNN layer by applying the layer successively on the input signal. Furthermore, it might be helpful to append the Kalman value as additional input since it has been shown to be useful for the Hospital Dataset signals.

To evaluate this hypothesis, permutations of the pre-processing layers *kalman, normalization, rnn* and *cnn* have been implemented. Table 5.8 gives an overview of the technical implementation of the preprocessing layers. In the following, these models will be referred to as *combined pre-processing models*. The precise parameters of these models are documented in section A.2 in the appendix.



(a) Normalized RNN

(b) Normalized CNN

(c) Kalman filter combined with RNN

(d) Kalman filter combined with CNN

(e) Kalman filter combined with a normalized RNN

(f) Kalman filter combined with a normalized CNN

Table 5.8.: PreProcessing layer implementation of the combined models.

The performance of these combined architectures on the decision events can be compared in figure 5.10. All precise experiment results can be found in section A.4 in the appendix. It can be seen that some combined prediction models outperform the basic models considerably.

The *Normalized CNN* and *Normalized RNN* models perform best in the process prediction tasks without anomalies. When anomalies are introduced in the training data, the *Kalman Normalized CNN* and *Kalman Normalized RNN* show similar performance or even surpass the model counterpart without the Kalman value on the 6000 instance dataset. The *Kalman CNN* and *Kalman RNN* models, on the other hand, have a more limited prediction accuracy similar to their not-combined counterparts.
This indicated that the *Normalization* combined with the *CNN* or *RNN* results in the biggest performance gain. The *Kalman* value makes the model more robust against anomalies without a measurable improvement of the prediction.

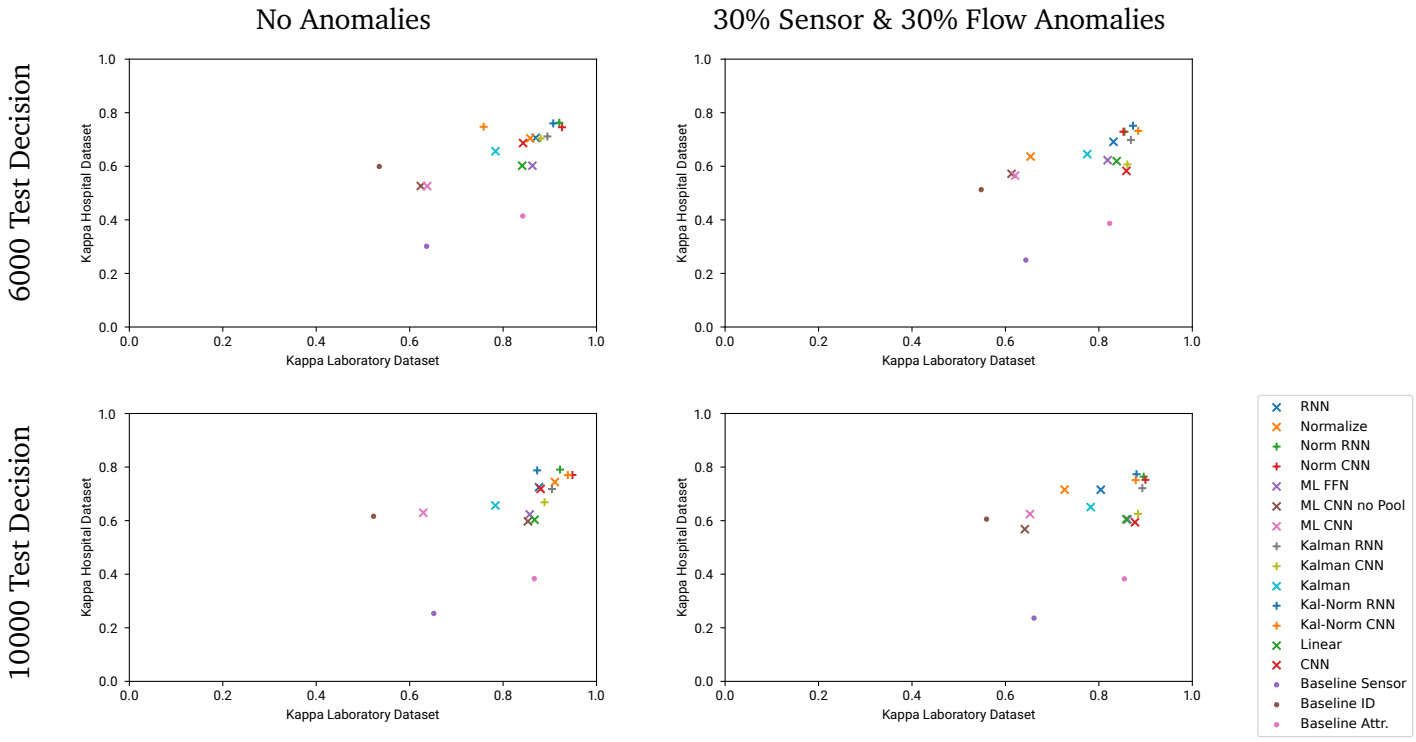


Figure 5.10.: Scatterplot of the Kappa performance of all model configurations on the decision events of the 10000 instances and 6000 instances datasets.

The experiments also confirm the previous hypothesis again that spatial variance of the sensor data is challenging for the *CNN* models. The combined models based on *CNN* layer perform generally slightly worse than the *RNN* models on the hospital dataset decisions

whereas they excel in the laboratory dataset.
In general, it has to be noted that the hospital dataset contains one *independent decision* which can not be predicted perfectly since its result is drawn randomly. Therefore, it is expected that the performances on the hospital dataset are lower than on the laboratory dataset, containing only *conditional decision*.

Overall the combined models manage to predict the laboratory dataset with flow and sensor anomalies with an accuracy of 98% and the hospital dataset with an accuracy of up to 93%. The *baseline id* model predict these datasets with an accuracy of 92.21% and 87.17% respectively. These results show that the process prediction component can successfully improve the *stochastic process model* by incorporating sensor attributes. The model manages to predict the laboratory dataset with an accuracy of up to 99.11% and the hospital dataset by 93.06% if no anomalies are in the training data. Therefore, the introduction of 30% sensor anomalies and 30% flow anomalies decreases the model performance by less than one percentage point.

The robustness of all models towards flow and sensor anomalies is an interesting observation. Unfortunately, none of the process prediction models in the literature presented in section 3.1 measured the isolated influence of anomalies in the training data on the prediction performance. However, the influence of anomalies has been measured in related work by systems for anomaly classification as discussed in section 3.2. It is not possible to compare these results directly with the process prediction system due to the different task objectives. Anomaly classification is based on the resulting probability distribution of the softmax activation for the predicted event. The process prediction task, on the other hand, obscures the distribution information by only using the most probable event from the prediction with an arg-max function. Hence, it is possible that the anomalies change the predicted event distribution but with too little impact that the most probable event changes. In the next chapter, this factor will be evaluated by presenting an anomaly classification system based on the process prediction model.

5.7. Anomaly Detection

To detect anomalies with the proposed system, a trained process prediction model has to be employed. The prediction of this model can then be discriminated against to classify anomalies. Since anomaly detection systems are mostly required in settings where anomalies in the event data exist, only prediction models trained on an event log with 30% flow and 30% sensor anomalies has been used for all experiments.

In the experiments, all baseline models, as well as the best performing process prediction models from the previous section, are evaluated against each other. The experiments are conducted for both datasets with the process prediction model training dataset size of 6000 and 10000 instances. The evaluation dataset has 30% flow anomalies and 30% sensor anomalies. This method aims at detecting all flow anomalies in an evaluation dataset with the size of 2000 process instances. The sensor anomalies only serve as dataset entropy to distract from the process flow anomaly classification task.

As described in subsection 4.2.1, the anomalies to classify can be an *skip* anomaly or an *rework* anomaly. For a skip anomaly, the event after the skipped sequence has to be labelled as anomalous, all events afterwards are considered normal. In an instance with a rework anomaly, all artificially inserted or repeated events have to be labelled as an anomaly.
The rework anomaly poses the challenge that up to three events are repeated after each other. Therefore, in the case of three rework events, two anomalous events have to be detected with anomalies in the input data of the prediction model. As described in the findings of Nolle et al. (2019, p.19-20), these anomalies might be more challenging to classify.

It has to be noted that some annotations of flow anomalies in the hospital dataset might diverge from anomalies in a logical perspective. This is caused by the violation of *R3* for the dataset in certain cases as described in subsection 5.3.2. The *waiting* activity might be repeated or skipped but will always be annotated as an anomaly in these cases. However, the process can still be logically interpreted as normal.

Figure 5.11 showcase the F_1 performance of the system for the anomaly classification task.
It can be seen that among the baselines, the baseline ID model manages to discriminate the anomalies best. The classification can mostly be improved by the sensor attribute models for the laboratory dataset. All combined preprocessing models perform very similar, independent of the variation of the RNN/CNN architecture or the Kalman value. For the hospital dataset, the baseline ID is almost as indicative as the combined preprocessing models. With a precision of less than 30% of all models in all classification tasks, the system performance can not be seen as effective. All exact performance metrics of the experiments with scatterplots of the prediction errors before the discrimination can be seen in the appendix in section A.5.

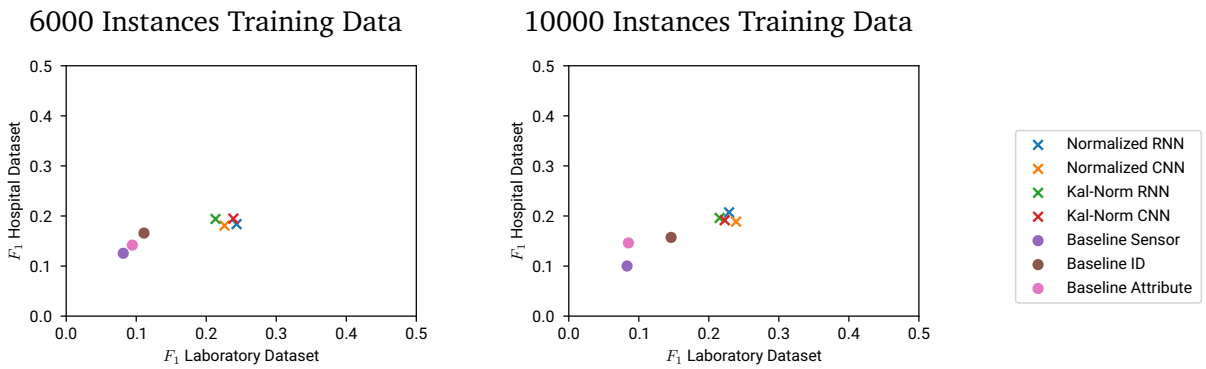


Figure 5.11.: Scatterplots of the F_1 performance of various anomaly classification systems based on the annotated process prediction model configuration. The process prediction models are trained on L10000/H10000 for the right plot and L6000/H6000 for the left plot. The classification is evaluated on an anomaly event classification task based on a dataset of 2000 instances with 30% flow and 30% sensor anomalies.

The analysis of the prediction errors with the computed discrimination threshold heuristic reveals one restriction of this anomaly classification system.
This issue is best described by the scatter plot of the classification system based on a normalized RNN on the hospital dataset in figure 5.12. The red dots indicate truly anomalous events and the green dots represent normal events whereas the red line indicates the computed threshold by the average mean squared error heuristic.
All normal events at the bottom of the plot are very easy to classify as normal since they have almost no prediction error. These events are either trivial to predict since they do not follow a decision, or well-learned and predicted by the prediction model. They also probably follow a *conditional decision*, where the flow outcome can be inferred by previous attribute values as described in section 4.1.1. Events after *independent decisions*, as the decision *HD1* in the hospital dataset, resulting in a more distributed probability distribution by the prediction model. Since the events can never be predicted perfectly, the given event always entail some distance to its prediction. This can be seen by the two green clusters of normal events between an MSE of 0.03-0.04.

The effective prediction model results in low error measures and the underrepresentation of anomalous events in the dataset does not balance the error scores. Therefore, the average error score heuristic always results in an unfitting threshold for the experiments.

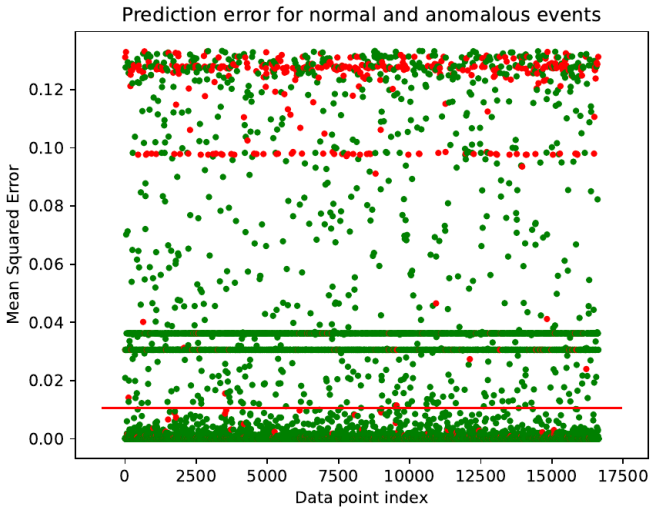


Figure 5.12.: Scatterplots of the prediction error of events for instance event classification with a normalized RNN prediction system trained on the hospital dataset H10000. The green dots are normal events whereas the red dots are actual anomalies.

To disregard the effect of the threshold heuristic, the model performance is evaluated with the threshold set by a grid search strategy. The performance of the models for event anomaly classification can be seen in figure 5.13. All precise performance metrics are documented in the appendix at A.5.3. For both datasets, the sensor models outperform the baseline models on average over the threshold range. Similar to the results for the process prediction task in section 5.5 and 5.6, the baseline ID model performs remarkably well. Like in the process prediction task, this is caused by a high number of trivial process flow sequences without decisions that can be solely inferred and therefore discriminated by the event id.

The RNN-based sensor representations manage to achieve a higher F_1 score for the classification on the hospital dataset than the CNN-based models or the baselines. In the laboratory dataset, the combined pre-processing models performed very similarly, hence no specific performance gain by the Kalman combined models could be detected. The

Kalman models even perform slightly worse than their counterparts without the filter on most threshold discriminations.

Overall, these results indicate that the proposed process prediction models can help identify flow anomalies in process models by incorporating sensor data. It has to be noted that the performance gain by the combined pre-processing models is achieved while having 30% sensor anomalies in the evaluation data. The proposed discrimination strategy on the other hand poses optimization potential.

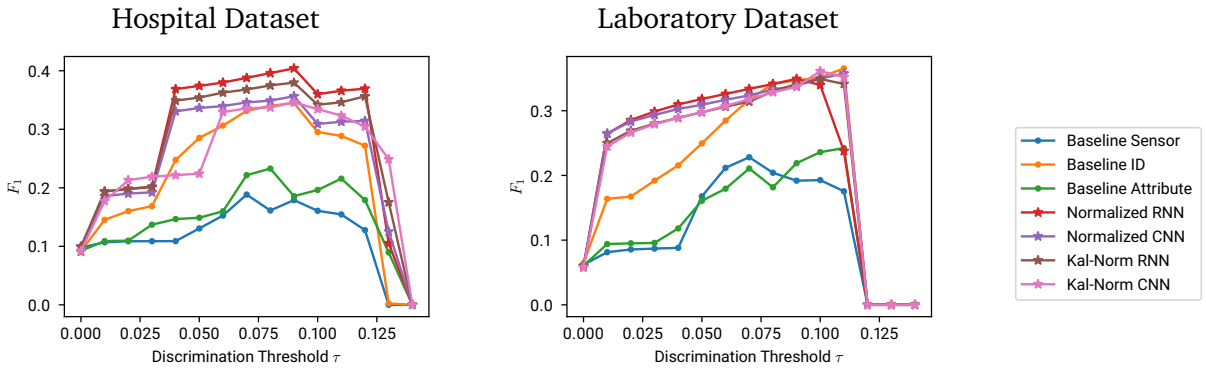


Figure 5.13.: Scatterplots of the F_1 performance of various anomaly classification systems based on the annotated process prediction model configuration. The process prediction models are trained on L10000/H10000 and discriminated with various threshold values as annotated on the x-axis. The classification is evaluated for an instance event classification task based on a dataset of 2000 instances and 30% flow anomalies.

6. Discussion

The results in this thesis are very promising but are so far only evaluated on simulated datasets. The fabrication of these simulated datasets with authentic interdependencies between event attributes and the control flow has been shown to be complicated. The presented datasets represent complex process dynamics but are still limited by the number of simulated decisions in the dataset. The high amount of trivial control flow sequences distract the performance gain of the model while incorporating sensor data for the decision prediction. Hence, the process prediction model has been additionally evaluated in a dataset containing only decision event predictions.
To avoid this filtering of evaluation data and ensure a transferability of the results to the real world, the system should be evaluated on a complete real-world dataset. Since so far none of the said datasets exists, the costly conversion of an existing dataset with the wrong format to an event log might be promising. Foundations of such a dataset could be the MIMIC 3 database hospital dataset with various sensor measurements and flow information given by patient movements (Johnson et al., 2016).

In this thesis, various possible sensor representations in process mining models have been narrowed down to the most effective ones. The presented experiments have shown that sensor data attributes can be represented most beneficially with CNN or RNN networks. Normalizations have also been shown to enhance the interpretability of these networks. On the other hand, the experiment results in this thesis do not indicate the usefulness of the Kalman filter. This thesis does not quantify the significance of the model improvements. To achieve this, the experiments should be repeated multiple times and be statistically evaluated.

For future work with the Kalman filter, it has to be noted that the original Kalman filter, as used in this thesis, is an optimal linear estimator (Kalman, 1960). To improve the training and prediction speed, the extended Kalman filter should be considered (McElhoe, 1966). This filter is a nonlinear Kalman filter, based on a first-order Taylor approximation of the state estimator and therefore serializable.

The evaluation of the anomaly classification experiments has shown that the proposed anomaly detection method is less than optimal. Future research should follow the suggestions of Nolle et al. (2019) and employ more advanced discriminators with advanced heuristics.

7. Conclusion

This thesis has presented several methods to utilize sensor data to effectively perform the tasks of anomaly classification and process prediction. The development of this system required the generation of an appropriate event log with sensor data and the evaluation of various sensor representations ranging from traditional signal processing up to machine learning methodologies. The following sections will answer the research questions posed in the introduction of this thesis and conclude their outcome.

7.1. RQ1: How can process mining tasks with sensor event attributes be evaluated?

The literature research and dataset analysis have shown that no available dataset or dataset generator represents sensor data authentically in an event log. Therefore, this thesis presents a synthetic dataset generator tailored to generate event logs with sensor data event attributes in section 4. The generator can represent two flow anomalies and seven sensor anomalies besides tools to simulate the initial sensor signal. Furthermore, sensor signals can be drawn from real-world datasets. This generator also contributes a novel perspective on the process flow determination of the simulated process by offering a capability of integrating knowledge about the process history into the path determination. Two dataset generators are defined within this thesis as described in section 5.3 based partly on real- and partly on simulated sensor data. The resulting datasets are provided with this thesis to serve for the evaluation of future research on sensor data in process mining.

7.2. RQ2: What is an effective representation of sensor data in a process model?

In this thesis, various possibilities to integrate sensor information as numeric time-series data into a process mining model have been evaluated. The core process mining model is represented by an RNN network. Since this model has statistical nature, it does not depend on a reference model nor needs prior knowledge about the process or the sensor data.
The experiments have shown that CNN and RNN based sensor representations enable the process model best to infer important information for the process flow prediction. This inference can be improved for both architectures by normalizing the pre-processing layer. The final model has shown to be robust against anomalies in the training data. It managed to learn the process from a dataset with 30% sensor anomalies and 30% flow anomalies. Finally, the model can incorporate sensor data with varying sensor data sizes and information characteristics.

7.3. RQ3: Are sensor event attributes helpful for anomaly detection or process prediction tasks?

The thesis found that sensor event attributes can improve the performance of process prediction systems. To what extend the prediction can be improved, depends on the amount of conditional and independent decisions in the dataset. Phrased differently, effective sensor representations enable to infer useful information for the process flow prediction if the process flow depends on the historic attribute information. The prediction model in this thesis managed to improve the process prediction accuracy by 7.68 per cent points from 91,43% to 99.11% by incorporating event attributes with sensor data from a dataset containing only conditional decisions. The system has also shown that this method can be extended to be used for anomaly detection. The sensor information improved the classification of flow anomalies in the test setting.

Bibliography

Aggarwal, Charu C. (2017). “An Introduction to Outlier Analysis”. In: *Outlier Analysis*. Cham: Springer International Publishing, pp. 1–34. ISBN: 978-3-319-47578-3. DOI: 10.1007/978-3-319-47578-3_1. URL: https://doi.org/10.1007/978-3-319-47578-3_1.

Agrawal, Rakesh, Dimitrios Gunopulos, and Frank Leymann (1998). “Mining Process Models from Workflow Logs”. In: *Advances in Database Technology — EDBT’98*. Ed. by Hans-Jörg Schek et al. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 467–483. ISBN: 978-3-540-69709-1. DOI: 10.1007/BFb0101003.

Augusto, Adriano et al. (Apr. 2019). “Automated Discovery of Process Models from Event Logs: Review and Benchmark”. In: *IEEE Transactions on Knowledge and Data Engineering* 31.4, pp. 686–705. ISSN: 1558-2191. DOI: 10.1109/TKDE.2018.2841877.

Bishop, C. M. (2013). *Pattern Recognition and Machine Learning: All ”Just the Facts 101” Material*. Springer (India) Private Limited. ISBN: 978-81-322-0906-5.

Böhmer, Kristof and Stefanie Rinderle-Ma (2016). “Multi-Perspective Anomaly Detection in Business Process Execution Events”. In: *On the Move to Meaningful Internet Systems: OTM 2016 Conferences*. Ed. by Christophe Debruyne et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 80–98. ISBN: 978-3-319-48472-3. DOI: 10.1007/978-3-319-48472-3_5.

Bosch Production Line Performance (2021). URL: https://kaggle.com/c/bosch-production-line-performance (visited on 05/03/2021).

Bote-Curiel, Luis et al. (2019). “Deep Learning and Big Data in Healthcare: A Double Review for Critical Beginners”. In: *Applied Sciences* 9.11. ISSN: 2076-3417. DOI: 10.3390/app9112331. URL: https://www.mdpi.com/2076-3417/9/11/2331.

Burattin, Andrea (2015). “PLG2: Multiperspective Processes Randomization and Simulation for Online and Offline Settings”. In: *CoRR* abs/1506.08415. arXiv: 1506.08415. URL: http://arxiv.org/abs/1506.08415.

Camargo, Manuel, Marlon Dumas, and Oscar González-Rojas (2019). "Learning Accurate LSTM Models of Business Processes". In: *Business Process Management*. Ed. by Thomas Hildebrandt et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 286–302. ISBN: 978-3-030-26619-6. DOI: 10.1007/978-3-030-26619-6_19.

Camargo, Manuel, Marlon Dumas, and Oscar González Rojas (2021). "Learning Accurate Business Process Simulation Models from Event Logs via Automated Process Discovery and Deep Learning". In: *CoRR* abs/2103.11944. arXiv: 2103.11944. URL: https://arxiv.org/abs/2103.11944.

Chalapathy, Raghavendra and Sanjay Chawla (2019). "Deep Learning for Anomaly Detection: A Survey". In: *CoRR* abs/1901.03407. arXiv: 1901.03407. URL: http://arxiv.org/abs/1901.03407.

Chapela-Campa, David, Manuel Mucientes, and Manuel Lama (2019). "Mining frequent patterns in process models". In: *Information Sciences* 472, pp. 235–257. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2018.09.011. URL: https://www.sciencedirect.com/science/article/pii/S0020025517304875.

Cho, Kyunghyun et al. (2014). "Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation". In: *CoRR* abs/1406.1078. arXiv: 1406.1078. URL: http://arxiv.org/abs/1406.1078.

Chung, Junyoung et al. (2014). "Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling". In: *CoRR* abs/1412.3555. arXiv: 1412.3555. URL: http://arxiv.org/abs/1412.3555.

Cohen, Jacob (1960). "A Coefficient of Agreement for Nominal Scales". In: *Educational and Psychological Measurement* 20.1, pp. 37–46. DOI: 10.1177/001316446002000104. eprint: https://doi.org/10.1177/001316446002000104. URL: https://doi.org/10.1177/001316446002000104.

Davenport, T.H., J.E. Short, and S.S.M.C. I (2018). *The New Industrial Engineering: Information Technology and Business Process Redesign*. FRANKLIN CLASSICS TRADE Press, p. 46. ISBN: 9780353294745.

De San Pedro, Javier, Josep Carmona, and Jordi Cortadella (2015). "Log-Based Simplification of Process Models". In: *Business Process Management*. Ed. by Hamid Reza Motahari-Nezhad, Jan Recker, and Matthias Weidlich. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 457–474. ISBN: 978-3-319-23063-4. DOI: 10.1007/978-3-319-23063-4_30.

Dees, M. and Boudewijn van Dongen (Apr. 2016). "BPI Challenge 2016: Clicks NOT Logged In". In: DOI: 10.4121/uuid:9b99a146-51b5-48df-aa70-288a76c82ec4. URL: https://data.4tu.nl/articles/dataset/BPI_Challenge_2016_Clicks_NOT_Logged_In/12708596.

Di Francescomarino, Chiara et al. (2018). "Predictive Process Monitoring Methods: Which One Suits Me Best?" In: *Business Process Management*. Ed. by Mathias Weske et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 462–479. ISBN: 978-3-319-98648-7. DOI: 10.1007/978-3-319-98648-7_27.

Di Mauro, Nicola, Annalisa Appice, and Teresa M. A. Basile (2019). "Activity Prediction of Business Process Instances with Inception CNN Models". In: *AI*IA 2019 – Advances in Artificial Intelligence*. Ed. by Mario Alviano, Gianluigi Greco, and Francesco Scarcello. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 348–361. ISBN: 978-3-030-35166-3. DOI: 10.1007/978-3-030-35166-3_25.

Dong, Yiqun (2019). "Implementing Deep Learning for comprehensive aircraft icing and actuator/sensor fault detection/identification". In: *Engineering Applications of Artificial Intelligence* 83, pp. 28–44. ISSN: 0952-1976. DOI: https://doi.org/10.1016/j.engappai.2019.04.010. URL: https://www.sciencedirect.com/science/article/pii/S0952197619300958.

Dua, Dheeru and Casey Graff (2017). "UCI Machine Learning Repository". In: URL: http://archive.ics.uci.edu/ml.

Evermann, Joerg, Jana-Rebecca Rehse, and Peter Fettke (2017). "Predicting process behaviour using deep learning". In: *Decision Support Systems* 100. Smart Business Process Management, pp. 129–140. ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2017.04.003. URL: https://www.sciencedirect.com/science/article/pii/S0167923617300635.

Gebrauchsanweisungen (2021). pro3dure medical | idea to product. URL: https://www.pro3dure.com/de/downloads/audiologie/gebrauchsanweisungen/ (visited on 11/01/2021).

Gebrauchsinformationen Audio (2021). URL: https://www.detax.de/en/audio/produkte/Gebrauchsinformationen.php#anchor_fd432ecf_Accordion-3D-Printing-Materials (visited on 11/01/2021).

Georgakopoulos, Diimitrios, Mark Hornick, and Amit Sheth (Apr. 1, 1995). "An Overview of Workflow Management: From Process Modeling to Workflow Automation Infrastructure". In: *Distributed and Parallel Databases* 3.2, pp. 119–153. ISSN: 1573-7578. DOI: 10.1007/BF01277643. URL: https://doi.org/10.1007/BF01277643.

Goodfellow, Ian et al. (2016). *Deep Learning*. Vol. 1. 2. http://www.deeplearningbook.org. MIT press Cambridge.

Graves, Alex and Jürgen Schmidhuber (July 1, 2005). "Framewise Phoneme Classification with Bidirectional LSTM and Other Neural Network Architectures". In: *Neural Networks*. IJCNN 2005 18.5, pp. 602–610. ISSN: 0893-6080. DOI: 10.1016/j.neunet.2005.06.042. URL: https://www.sciencedirect.com/science/article/pii/S0893608005001206.

Heinrich, Kai et al. (Apr. 1, 2021). “Process Data Properties Matter: Introducing Gated Convolutional Neural Networks (GCNN) and Key-Value-Predict Attention Networks (KVP) for next Event Prediction with Deep Learning”. In: *Decision Support Systems* 143, p. 113494. ISSN: 0167-9236. DOI: 10.1016/j.dss.2021.113494. URL: https://www.sciencedirect.com/science/article/pii/S016792362100004X.

Hinkka, Markku, Teemu Lehto, and Keijo Heljanko (2020). “Exploiting Event Log Event Attributes in RNN Based Prediction”. In: *Data-Driven Process Discovery and Analysis*. Ed. by Paolo Ceravolo, Maurice van Keulen, and María Teresa Gómez-López. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, pp. 67–85. ISBN: 978-3-030-46633-6. DOI: 10.1007/978-3-030-46633-6_4.

Hinkka, Markku et al. (2019). “Classifying Process Instances Using Recurrent Neural Networks”. In: *Business Process Management Workshops*. Ed. by Florian Daniel, Quan Z. Sheng, and Hamid Motahari. Lecture Notes in Business Information Processing. Cham: Springer International Publishing, pp. 313–324. ISBN: 978-3-030-11641-5. DOI: 10.1007/978-3-030-11641-5_25.

Hochreiter, Sepp (1991). “Untersuchungen Zu Dynamischen Neuronalen Netzen”. In: *Diploma, Technische Universität München* 91.1.

Hochreiter, Sepp and Jürgen Schmidhuber (Nov. 15, 1997). “Long Short-Term Memory”. In: *Neural Computation* 9.8, pp. 1735–1780. ISSN: 0899-7667. DOI: 10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

Hollingsworth, David and U. K. Hampshire (1995). “Workflow Management Coalition: The Workflow Reference Model”. In: *Document Number TC00-1003* 19.16, p. 224.

Ioffe, Sergey and Christian Szegedy (June 1, 2015). “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift”. In: *Proceedings of the 32nd International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 448–456. URL: https://proceedings.mlr.press/v37/ioffe15.html.

Johnson, Alistair E. W. et al. (May 24, 2016). “MIMIC-III, a Freely Accessible Critical Care Database”. In: *Scientific Data* 3.1 (1), p. 160035. ISSN: 2052-4463. DOI: 10.1038/sdata.2016.35. URL: https://www.nature.com/articles/sdata201635 (visited on 11/27/2021).

Jouck, Toon and Benoît Depaire (2016). “PTandLogGenerator: A Generator for Artificial Event Data.” In: *BPM (Demos)* 1789, pp. 23–27.

Jouck, Toon and Benoit Depaire (Dec. 1, 2019). “Generating Artificial Data for Empirical Analysis of Control-Flow Discovery Algorithms”. In: *Business & Information Systems Engineering* 61.6, pp. 695–712. ISSN: 1867-0202. DOI: 10.1007/s12599-018-0541-5. URL: https://doi.org/10.1007/s12599-018-0541-5.

Junior, Sylvio Barbon et al. (Oct. 2020). “Anomaly Detection on Event Logs with a Scarcity of Labels”. In: *2020 2nd International Conference on Process Mining (ICPM)*. 2020 2nd International Conference on Process Mining (ICPM), pp. 161–168. DOI: 10.1109/ICPM49681.2020.00032.

Kalman, R. E. (Mar. 1960). “A New Approach to Linear Filtering and Prediction Problems”. In: *Journal of Basic Engineering* 82.1, pp. 35–45. ISSN: 0021-9223. DOI: 10.1115/1.3662552. eprint: https://asmedigitalcollection.asme.org/fluidsengineering/article-pdf/82/1/35/5518977/35_1.pdf. URL: https://doi.org/10.1115/1.3662552.

Kammerer, Klaus et al. (Jan. 2019). “Anomaly Detections for Manufacturing Systems Based on Sensor Data—Insights into Two Challenging Real-World Production Settings”. In: *Sensors* 19.24 (24), p. 5370. DOI: 10.3390/s19245370. URL: https://www.mdpi.com/1424-8220/19/24/5370 (visited on 09/23/2021).

Karolina Dziugaite, Gintare, Shai Ben-David, and Daniel M. Roy (Oct. 1, 2020). “Enforcing Interpretability and Its Statistical Impacts: Trade-Offs between Accuracy and Interpretability”. In: *arXiv e-prints*, arXiv:2010.13764. URL: https://ui.adsabs.harvard.edu/abs/2020arXiv201013764K.

Kingma, Diederik P. and Jimmy Ba (Jan. 29, 2017). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs]. URL: http://arxiv.org/abs/1412.6980 (visited on 11/18/2021).

Krajsic, P. and B. Franczyk (2020). “Lambda Architecture for Anomaly Detection in Online Process Mining Using Autoencoders”. In: *Advances in Computational Collective Intelligence*. Ed. by Marcin Hernes, Krystian Wojtkiewicz, and Edward Szczerbicki. Communications in Computer and Information Science. Cham: Springer International Publishing, pp. 579–589. ISBN: 978-3-030-63119-2. DOI: 10.1007/978-3-030-63119-2_47.

– (2021). “Semi-Supervised Anomaly Detection in Business Process Event Data using Self-Attention based Classification”. In: *Procedia Computer Science* 192. Knowledge-Based and Intelligent Information & Engineering Systems: Proceedings of the 25th International Conference KES2021, pp. 39–48. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2021.08.005. URL: https://www.sciencedirect.com/science/article/pii/S1877050921014927.

Lawrence, S. et al. (Jan. 1997). “Face Recognition: A Convolutional Neural-Network Approach”. In: *IEEE Transactions on Neural Networks* 8.1, pp. 98–113. ISSN: 1941-0093. DOI: 10.1109/72.554195.

Lee, Ki Bum, Sejune Cheon, and Chang Ouk Kim (May 2017). “A Convolutional Neural Network for Fault Classification and Diagnosis in Semiconductor Manufacturing

Processes". In: *IEEE Transactions on Semiconductor Manufacturing* 30.2, pp. 135–142. ISSN: 1558-2345. DOI: 10.1109/TSM.2017.2676245.

Leigh, Catherine et al. (May 10, 2019). "A Framework for Automated Anomaly Detection in High Frequency Water-Quality Data from in Situ Sensors". In: *Science of The Total Environment* 664, pp. 885–898. ISSN: 0048-9697. DOI: 10.1016/j.scitotenv.2019.02.085. URL: https://www.sciencedirect.com/science/article/pii/S0048969719305662.

Leymann, F. and W. Altenhuber (1994). "Managing Business Processes as an Information Resource". In: *IBM Systems Journal* 33.2, pp. 326–348. ISSN: 0018-8670. DOI: 10.1147/sj.332.0326.

Lin, Li, Lijie Wen, and Jianmin Wang (May 6, 2019). "MM-Pred: A Deep Predictive Model for Multi-Attribute Event Sequence". In: *Proceedings of the 2019 SIAM International Conference on Data Mining (SDM)*. Proceedings. Society for Industrial and Applied Mathematics, pp. 118–126. DOI: 10.1137/1.9781611975673.14. URL: https://epubs.siam.org/doi/10.1137/1.9781611975673.14.

Malhotra, Pankaj et al. (2016). "LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection". In: *CoRR* abs/1607.00148. arXiv: 1607.00148. URL: http://arxiv.org/abs/1607.00148.

Mayer, Hermann et al. (Oct. 2006). "A System for Robotic Heart Surgery That Learns to Tie Knots Using Recurrent Neural Networks". In: *2006 IEEE/RSJ International Conference on Intelligent Robots and Systems*. 2006 IEEE/RSJ International Conference on Intelligent Robots and Systems, pp. 543–548. DOI: 10.1109/IROS.2006.282190.

McElhoe, Bruce A. (1966). "An Assessment of the Navigation and Course Corrections for a Manned Flyby of Mars or Venus". In: *IEEE Transactions on Aerospace and Electronic Systems* 4, pp. 613–623.

Mehdiyev, Nijat, Joerg Evermann, and Peter Fettke (July 2017). "A Multi-Stage Deep Learning Approach for Business Process Event Prediction". In: *2017 IEEE 19th Conference on Business Informatics (CBI)*. 2017 IEEE 19th Conference on Business Informatics (CBI). Vol. 01, pp. 119–128. DOI: 10.1109/CBI.2017.46.

Mitsyuk, Alexey A. et al. (May 1, 2017). "Generating Event Logs for High-Level Process Models". In: *Simulation Modelling Practice and Theory* 74, pp. 1–16. ISSN: 1569-190X. DOI: 10.1016/j.simpat.2017.01.003. URL: https://www.sciencedirect.com/science/article/pii/S1569190X17300047.

Mohammadi, Mehdi et al. (Fourthquarter 2018). "Deep Learning for IoT Big Data and Streaming Analytics: A Survey". In: *IEEE Communications Surveys Tutorials* 20.4, pp. 2923–2960. ISSN: 1553-877X. DOI: 10.1109/COMST.2018.2844341.

Moody, George B and Roger G Mark (2001). *The impact of the MIT-BIH arrhythmia database*. 3. IEEE.

Nolle, Timo, Alexander Seeliger, and Max Mühlhäuser (2016). "Unsupervised Anomaly Detection in Noisy Business Process Event Logs Using Denoising Autoencoders". In: *Discovery Science*. Ed. by Toon Calders, Michelangelo Ceci, and Donato Malerba. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 442–456. ISBN: 978-3-319-46307-0. DOI: 10.1007/978-3-319-46307-0_28.

– (2018a). "BINet: Multivariate Business Process Anomaly Detection Using Deep Learning". In: *Business Process Management*. Ed. by Mathias Weske et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 271–287. ISBN: 978-3-319-98648-7. DOI: 10.1007/978-3-319-98648-7_16.

Nolle, Timo et al. (Nov. 1, 2018b). "Analyzing Business Process Anomalies Using Autoencoders". In: *Machine Learning* 107.11, pp. 1875–1893. ISSN: 1573-0565. DOI: 10.1007/s10994-018-5702-8. URL: https://doi.org/10.1007/s10994-018-5702-8.

– (Oct. 26, 2019). "BINet: Multi-Perspective Business Process Anomaly Classification". In: *Information Systems*, p. 101458. ISSN: 0306-4379. DOI: 10.1016/j.is.2019.101458. URL: https://www.sciencedirect.com/science/article/pii/S0306437919305101.

Pang, Guansong et al. (Mar. 5, 2021). "Deep Learning for Anomaly Detection: A Review". In: *ACM Computing Surveys* 54.2, 38:1–38:38. ISSN: 0360-0300. DOI: 10.1145/3439950. URL: https://doi.org/10.1145/3439950.

Pasquadibisceglie, Vincenzo et al. (June 2019). "Using Convolutional Neural Networks for Predictive Process Analytics". In: *2019 International Conference on Process Mining (ICPM)*. 2019 International Conference on Process Mining (ICPM), pp. 129–136. DOI: 10.1109/ICPM.2019.00028.

Pearson, Karl (Jan. 1, 1895). "Note on Regression and Inheritance in the Case of Two Parents". In: *Proceedings of the Royal Society of London Series I* 58, pp. 240–242. URL: https://ui.adsabs.harvard.edu/abs/1895RSPS...58..240P.

Perlin, Ken (July 1, 1985). "An Image Synthesizer". In: *ACM SIGGRAPH Computer Graphics* 19.3, pp. 287–296. ISSN: 0097-8930. DOI: 10.1145/325165.325247. URL: https://doi.org/10.1145/325165.325247.

Ranjan, Chitta et al. (2019). *Dataset: Rare Event Classification in Multivariate Time Series*. arXiv: 1809.10717 [stat.ML].

Reinwald, Berthold and Hartmut Wedekind (1992). "Automation of Control and Data Flow in Distributed Application Systems". In: *Database and Expert Systems Applications*. Ed. by A Min Tjoa and Isidro Ramos. Vienna: Springer, pp. 475–481. ISBN: 978-3-7091-7557-6. DOI: 10.1007/978-3-7091-7557-6_81.

Rozinat, A. et al. (July 2009). "Process Mining Applied to the Test Process of Wafer Scanners in ASML". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Appli-*

cations and Reviews) 39.4, pp. 474–479. ISSN: 1558-2442. DOI: 10.1109/TSMCC.2009.2014169.

Rumelhart, David E., Geoffrey E. Hinton, and Ronald J. Williams (1986). "Learning Representations by Back-Propagating Errors". In: *nature* 323.6088, pp. 533–536.

Saeed, Mohammed et al. (2002). "MIMIC II: A Massive Temporal ICU Patient Database to Support Research in Intelligent Patient Monitoring". In: *Computers in Cardiology*. IEEE, pp. 641–644. ISBN: 0-7803-7735-4.

Sánchez Ferreres, Josep (June 2018). "Model-Agnostic process modelling". PhD thesis. UPC, Facultat d'Informàtica de Barcelona. URL: http://hdl.handle.net/2117/121614.

Schmidhuber, Jürgen, Daan Wierstra, and Faustino J. Gomez (2005). "Evolino: Hybrid Neuroevolution/Optimal Linear Search for Sequence Prediction". In: *Proceedings of the 19th International Joint Conferenceon Artificial Intelligence (IJCAI)*.

Schönig, Stefan et al. (2018). "Deep Learning Process Prediction with Discrete and Continuous Data Features:" in: *Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering*. 13th International Conference on Evaluation of Novel Approaches to Software Engineering. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, pp. 314–319. ISBN: 978-989-758-300-1. DOI: 10.5220/0006772003140319. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006772003140319.

Schwab, Klaus (2017). *The fourth industrial revolution*. Currency.

Skydanienko, Vasyl et al. (2018). "A Tool for Generating Event Logs from Multi-Perspective Declare Models." In: *BPM (Dissertation/Demos/Industry)*, pp. 111–115.

Solomakhina, Nina et al. (July 2014). "Extending Statistical Data Quality Improvement with Explicit Domain Models". In: *2014 12th IEEE International Conference on Industrial Informatics (INDIN)*. 2014 12th IEEE International Conference on Industrial Informatics (INDIN), pp. 720–725. DOI: 10.1109/INDIN.2014.6945602.

Steeman, Ward (Apr. 12, 2013). *BPI Challenge 2013, Incidents*. In collab. with Volvo IT. Version 1. Ghent University. DOI: 10.4121/UUID:500573E6-ACCC-4B0C-9576-AA5468B10CEE. URL: https://data.4tu.nl/articles/_/12693914/1.

Sutskever, Ilya et al. (May 26, 2013). "On the Importance of Initialization and Momentum in Deep Learning". In: *Proceedings of the 30th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 1139–1147. URL: https://proceedings.mlr.press/v28/sutskever13.html.

Tanuska, Pavol et al. (Jan. 2021). "Smart Anomaly Detection and Prediction for Assembly Process Maintenance in Compliance with Industry 4.0". In: *Sensors* 21.7 (7), p. 2376. DOI: 10.3390/s21072376. URL: https://www.mdpi.com/1424-8220/21/7/2376.

Tax, Niek, Irene Teinemaa, and Sebastiaan J. van Zelst (Nov. 1, 2020). "An Interdisciplinary Comparison of Sequence Modeling Methods for Next-Element Prediction". In: *Software and Systems Modeling* 19.6, pp. 1345–1365. ISSN: 1619-1374. DOI: 10.1007/s10270-020-00789-3. URL: https://doi.org/10.1007/s10270-020-00789-3.

Tax, Niek et al. (2017). "Predictive Business Process Monitoring with LSTM Neural Networks". In: *Advanced Information Systems Engineering*. Ed. by Eric Dubois and Klaus Pohl. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 477–492. ISBN: 978-3-319-59536-8. DOI: 10.1007/978-3-319-59536-8_30.

Teh, Hui Yie, Andreas W. Kempa-Liehr, and Kevin I-Kai Wang (Feb. 11, 2020). "Sensor Data Quality: A Systematic Review". In: *Journal of Big Data* 7.1, p. 11. ISSN: 2196-1115. DOI: 10.1186/s40537-020-0285-1. URL: https://doi.org/10.1186/s40537-020-0285-1.

Van Der Aalst, Wil (Aug. 1, 2012). "Process Mining". In: *Communications of the ACM* 55.8, pp. 76–83. ISSN: 0001-0782. DOI: 10.1145/2240236.2240257. URL: https://doi.org/10.1145/2240236.2240257.

Van der Aalst, Wil et al. (2012). "Process Mining Manifesto". In: *Business Process Management Workshops*. Ed. by Florian Daniel, Kamel Barkaoui, and Schahram Dustdar. Lecture Notes in Business Information Processing. Berlin, Heidelberg: Springer, pp. 169–194. ISBN: 978-3-642-28108-2. DOI: 10.1007/978-3-642-28108-2_19.

Van Dongen, B. F. et al. (2005). "The ProM Framework: A New Era in Process Mining Tool Support". In: *Applications and Theory of Petri Nets 2005*. Ed. by Gianfranco Ciardo and Philippe Darondeau. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, pp. 444–454. ISBN: 978-3-540-31559-9. DOI: 10.1007/11494744_25.

Van Dongen, Boudewijn (Mar. 23, 2011). *Real-life event logs - Hospital log*. In collab. with Eindhoven University Of Technology. Version 1. Eindhoven University of Technology. DOI: 10.4121/UUID:D9769F3D-0AB0-4FB8-803B-0D1120FFCF54. URL: https://data.4tu.nl/articles/_/12716513/1.

– (Apr. 23, 2012). *BPI Challenge 2012*. In collab. with Eindhoven University Of Technology. Version 1. Eindhoven University of Technology. DOI: 10.4121/UUID:3926DB30-F712-4394-AEBC-75976070E91F. URL: https://data.4tu.nl/articles/_/12689204/1.

– (Apr. 23, 2014). *BPI Challenge 2014: Change Details*. In collab. with Rabobank Nederland. Version 1. Rabobank Nederland. DOI: 10.4121/UUID:D5CCB355-CA67-480F-8739-289B9B593AAF. URL: https://data.4tu.nl/articles/_/12716234/1.

– (May 1, 2015). *BPI Challenge 2015 Municipality 1*. In collab. with Eindhoven University Of Technology. Version 1. Eindhoven University of Technology. DOI: 10.4121/UUID:

A0ADDFDA-2044-4541-A450-FDCC9FE16D17. URL: https://data.4tu.nl/articles/_/12709154/1.

Van Dongen, Boudewijn (Feb. 6, 2017). *BPI Challenge 2017*. In collab. with Eindhoven University Of Technology. Version 1. Eindhoven University of Technology. DOI: 10.4121/UUID:5F3067DF-F10B-45DA-B98B-86AE4C7A310B. URL: https://data.4tu.nl/articles/_/12696884/1.

– (Jan. 31, 2019). *BPI Challenge 2019*. In collab. with Department Of Mathematics Eindhoven University Of Technology and Computer Science. Version 1. 4TU.Centre for Research Data. DOI: 10.4121/UUID:D06AFF4B-79F0-45E6-8EC8-E19730C248F1. URL: https://data.4tu.nl/articles/_/12715853/1.

Van Dongen, Boudewijn and F. (Florian) Borchert (Mar. 13, 2018). *BPI Challenge 2018*. In collab. with Eindhoven University Of Technology and Germany Data Experts Neubrandenburg. Version 1. Eindhoven University of Technology. DOI: 10.4121/UUID:3301445F-95E8-4FF0-98A4-901F1F204972. URL: https://data.4tu.nl/articles/_/12688355/1.

Van Wyk, Franco et al. (Mar. 2020). "Real-Time Sensor Anomaly Detection and Identification in Automated Vehicles". In: *IEEE Transactions on Intelligent Transportation Systems* 21.3, pp. 1264–1276. ISSN: 1558-0016. DOI: 10.1109/TITS.2019.2906038.

Verenich, Ilya (Dec. 1, 2016). *Helpdesk*. Mendeley. DOI: 10.17632/39BP3VV62T.1. URL: https://data.mendeley.com/datasets/39bp3vv62t/1.

White, Stephen A. (2004). "Introduction to BPMN". In: *Ibm Cooperation* 2.0.

A. Appendix

A.1. Sensor Anomalies

```
import random
import numpy as np

def drift(x, strength=0.1, positiv=False, negative=False):
    if not positiv or negative:
        positiv = np.random.choice([True, False])
    time = np.arange(len(x))
    if positiv:
        trend = time * ((max(x) * strength) / len(x))
    else:
        trend = time * -((max(x) * strength) / len(x))
    return x + trend

def noise(x, strength=0.1):
    mean_noise = 0
    variance = max(max(x) - min(x), 1)
    noise = np.random.normal(mean_noise, variance * strength, len(x))
    return x + noise

def spike(x):
    random_index = random.randint(0, len(x) - 1)
    amplitude = random.uniform(5.0, 5.5)
    if np.random.choice([False, True], p=[0.5, 0.5]):
        x[random_index] = x[random_index] * amplitude
    else:
```

```
            x[random_index] = x[random_index] / amplitude
        return x

def offset(x, percent=0.1):
    constant = max(x) * percent
    return [value + constant for value in x]

def spike_cluster(x):
    start_index = random.randint(0, int(len(x) * 0.95))
    size = np.random.uniform(low=0.01, high=0.05)
    increase = np.random.choice([True, False], p=[0.50, 0.50])
    end_index = int(len(x) * size) + start_index
    for index in range(start_index, end_index):
        factor = np.random.uniform(low=0.2, high=0.5)
        if increase: factor = (1 - factor) + 1
        x[index] = x[index] * factor
    return x

def stuck_at_zero(x):
    start_index = random.randint(0, int(len(x) / 2))
    end_index = random.randint(int(len(x) / 2), len(x) - 1)
    for index in range(start_index, end_index):
        x[index] = 0
    return x

def stuck_at_constant(x):
    start_index = random.randint(0, int(len(x) / 2))
    end_index = random.randint(int(len(x) / 2), len(x) - 1)
    constant_value = x[start_index]
    for index in range(start_index, end_index):
        x[index] = constant_value
    return x
```

A.2. Model Parameter

A.2.1. CNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	conv1d	Conv1D	(None, None, 10)
8	conv1d_1	Conv1D	(None, None, 10)
9	conv1d_2	Conv1D	(None, None, 10)
10	conv1d_3	Conv1D	(None, None, 10)
11	conv1d_4	Conv1D	(None, None, 10)
12	conv1d_5	Conv1D	(None, None, 10)
13	conv1d_6	Conv1D	(None, None, 10)
14	id	InputLayer	[(None, None, 17)]
15	product_type_0	InputLayer	[(None, None, 1)]
16	tf.cast_20	TFOpLambda	(None, None, 10)
17	tf.cast_21	TFOpLambda	(None, None, 10)
18	skipped_layer_0	InputLayer	[(None, None, 1)]
19	print_finished_0	InputLayer	[(None, None, 1)]
20	tf.cast_22	TFOpLambda	(None, None, 10)
21	tf.cast_23	TFOpLambda	(None, None, 10)
22	tf.cast_24	TFOpLambda	(None, None, 10)
23	tf.cast_25	TFOpLambda	(None, None, 10)
24	tf.cast_26	TFOpLambda	(None, None, 10)
25	concatenate_1	Concatenate	(None, None, 90)
26	encoder	GRU	(None, None, 300)
27	encoder_normalization	BatchNormalization	(None, None, 300)
28	decoder	GRU	(None, None, 200)
29	decoder_normalization	BatchNormalization	(None, None, 200)
30	flatten_1	Flatten	(None, 6000)

31	dense_1	Dense	(None, 17)

A.2.2. ML CNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	conv1d_7	Conv1D	(None, None, 100)
8	conv1d_10	Conv1D	(None, None, 200)
9	conv1d_13	Conv1D	(None, None, 206)
10	conv1d_16	Conv1D	(None, None, 100)
11	conv1d_19	Conv1D	(None, None, 440)
12	conv1d_22	Conv1D	(None, None, 15)
13	conv1d_25	Conv1D	(None, None, 15)
14	max_pooling1d	MaxPooling1D	(None, None, 33)
15	max_pooling1d_3	MaxPooling1D	(None, None, 66)
16	max_pooling1d_6	MaxPooling1D	(None, None, 68)
17	max_pooling1d_9	MaxPooling1D	(None, None, 33)
18	max_pooling1d_12	MaxPooling1D	(None, None, 146)
19	max_pooling1d_15	MaxPooling1D	(None, None, 5)
20	max_pooling1d_18	MaxPooling1D	(None, None, 5)
21	conv1d_8	Conv1D	(None, None, 50)
22	conv1d_11	Conv1D	(None, None, 100)
23	conv1d_14	Conv1D	(None, None, 103)
24	conv1d_17	Conv1D	(None, None, 50)
25	conv1d_20	Conv1D	(None, None, 220)
26	conv1d_23	Conv1D	(None, None, 7)
27	conv1d_26	Conv1D	(None, None, 7)
28	max_pooling1d_1	MaxPooling1D	(None, None, 25)
29	max_pooling1d_4	MaxPooling1D	(None, None, 50)
30	max_pooling1d_7	MaxPooling1D	(None, None, 51)

31	max_pooling1d_10	MaxPooling1D	(None, None, 25)
32	max_pooling1d_13	MaxPooling1D	(None, None, 110)
33	max_pooling1d_16	MaxPooling1D	(None, None, 3)
34	max_pooling1d_19	MaxPooling1D	(None, None, 3)
35	conv1d_9	Conv1D	(None, None, 25)
36	conv1d_12	Conv1D	(None, None, 50)
37	conv1d_15	Conv1D	(None, None, 51)
38	conv1d_18	Conv1D	(None, None, 25)
39	conv1d_21	Conv1D	(None, None, 110)
40	conv1d_24	Conv1D	(None, None, 3)
41	conv1d_27	Conv1D	(None, None, 3)
42	max_pooling1d_2	MaxPooling1D	(None, None, 12)
43	max_pooling1d_5	MaxPooling1D	(None, None, 25)
44	max_pooling1d_8	MaxPooling1D	(None, None, 25)
45	max_pooling1d_11	MaxPooling1D	(None, None, 12)
46	max_pooling1d_14	MaxPooling1D	(None, None, 55)
47	max_pooling1d_17	MaxPooling1D	(None, None, 1)
48	max_pooling1d_20	MaxPooling1D	(None, None, 1)
49	id	InputLayer	[(None, None, 17)]
50	product_type_0	InputLayer	[(None, None, 1)]
51	tf.cast_27	TFOpLambda	(None, None, 12)
52	tf.cast_28	TFOpLambda	(None, None, 25)
53	skipped_layer_0	InputLayer	[(None, None, 1)]
54	print_finished_0	InputLayer	[(None, None, 1)]
55	tf.cast_29	TFOpLambda	(None, None, 25)
56	tf.cast_30	TFOpLambda	(None, None, 12)
57	tf.cast_31	TFOpLambda	(None, None, 55)
58	tf.cast_32	TFOpLambda	(None, None, 1)
59	tf.cast_33	TFOpLambda	(None, None, 1)
60	concatenate_2	Concatenate	(None, None, 151)
61	encoder	GRU	(None, None, 300)
62	encoder_normalization	BatchNormalization	(None, None, 300)
63	decoder	GRU	(None, None, 200)
64	decoder_normalization	BatchNormalization	(None, None, 200)
65	flatten_2	Flatten	(None, 6000)
66	dense_2	Dense	(None, 17)

A.2.3. RNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	gru	GRU	(None, None, 10)
8	gru_1	GRU	(None, None, 10)
9	gru_2	GRU	(None, None, 10)
10	gru_3	GRU	(None, None, 10)
11	gru_4	GRU	(None, None, 10)
12	gru_5	GRU	(None, None, 10)
13	gru_6	GRU	(None, None, 10)
14	id	InputLayer	[(None, None, 17)]
15	product_type_0	InputLayer	[(None, None, 1)]
16	tf.cast_34	TFOpLambda	(None, None, 10)
17	tf.cast_35	TFOpLambda	(None, None, 10)
18	skipped_layer_0	InputLayer	[(None, None, 1)]
19	print_finished_0	InputLayer	[(None, None, 1)]
20	tf.cast_36	TFOpLambda	(None, None, 10)
21	tf.cast_37	TFOpLambda	(None, None, 10)
22	tf.cast_38	TFOpLambda	(None, None, 10)
23	tf.cast_39	TFOpLambda	(None, None, 10)
24	tf.cast_40	TFOpLambda	(None, None, 10)
25	concatenate_3	Concatenate	(None, None, 90)
26	encoder	GRU	(None, None, 300)
27	encoder_normalization	BatchNormalization	(None, None, 300)
28	decoder	GRU	(None, None, 200)
29	decoder_normalization	BatchNormalization	(None, None, 200)
30	flatten_3	Flatten	(None, 6000)
31	dense_3	Dense	(None, 17)

A.2.4. Linear

	Name	Type	Shape
0	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
1	conv1d_28	Conv1D	(None, None, 10)
2	id	InputLayer	[(None, None, 17)]
3	product_type_0	InputLayer	[(None, None, 1)]
4	sensor_print_temp_0	InputLayer	[(None, None, 300)]
5	sensor_load_curve_0	InputLayer	[(None, None, 600)]
6	skipped_layer_0	InputLayer	[(None, None, 1)]
7	print_finished_0	InputLayer	[(None, None, 1)]
8	sensor_washing_0	InputLayer	[(None, None, 620)]
9	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
10	tf.cast_41	TFOpLambda	(None, None, 10)
11	sensor_pressure_0	InputLayer	[(None, None, 45)]
12	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
13	concatenate_4	Concatenate	(None, None, 1940)
14	encoder	GRU	(None, None, 300)
15	encoder_normalization	BatchNormalization	(None, None, 300)
16	decoder	GRU	(None, None, 200)
17	decoder_normalization	BatchNormalization	(None, None, 200)
18	flatten_4	Flatten	(None, 6000)
19	dense_4	Dense	(None, 17)

A.2.5. Normalize

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	id	InputLayer	[(None, None, 17)]

8	product_type_0	InputLayer	[(None, None, 1)]
9	lambda	Lambda	(None, None, 1)
10	layer_normalization	LayerNormalization	(None, None, 300)
11	lambda_1	Lambda	(None, None, 1)
12	layer_normalization_1	LayerNormalization	(None, None, 600)
13	skipped_layer_0	InputLayer	[(None, None, 1)]
14	print_finished_0	InputLayer	[(None, None, 1)]
15	lambda_2	Lambda	(None, None, 1)
16	layer_normalization_2	LayerNormalization	(None, None, 620)
17	lambda_3	Lambda	(None, None, 1)
18	layer_normalization_3	LayerNormalization	(None, None, 300)
19	lambda_4	Lambda	(None, None, 1)
20	layer_normalization_4	LayerNormalization	(None, None, 1320)
21	lambda_5	Lambda	(None, None, 1)
22	layer_normalization_5	LayerNormalization	(None, None, 45)
23	lambda_6	Lambda	(None, None, 1)
24	layer_normalization_6	LayerNormalization	(None, None, 45)
25	concatenate_5	Concatenate	(None, None, 3257)
26	encoder	GRU	(None, None, 300)
27	encoder_normalization	BatchNormalization	(None, None, 300)
28	decoder	GRU	(None, None, 200)
29	decoder_normalization	BatchNormalization	(None, None, 200)
30	flatten_5	Flatten	(None, 6000)
31	dense_5	Dense	(None, 17)

A.2.6. Baseline ID

	Name	Type	Shape
0	id	InputLayer	[(None, None, 17)]
1	encoder	GRU	(None, None, 300)
2	encoder_normalization	BatchNormalization	(None, None, 300)
3	decoder	GRU	(None, None, 200)
4	decoder_normalization	BatchNormalization	(None, None, 200)
5	flatten_1	Flatten	(None, 3800)
6	sensor_ecg_history_0	InputLayer	[(None, None, 186)]
7	sensor_abp_history_0	InputLayer	[(None, None, 1000)]

8	patient_age_0	InputLayer	[(None, None, 1)]
9	sensor_ecg_0	InputLayer	[(None, None, 186)]
10	sensor_abp_0	InputLayer	[(None, None, 1000)]
11	dense_1	Dense	(None, 17)

A.2.7. Baseline Sensor

	Name	Type	Shape
0	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
1	conv1d	Conv1D	(None, None, 10)
2	sensor_print_temp_0	InputLayer	[(None, None, 300)]
3	sensor_load_curve_0	InputLayer	[(None, None, 600)]
4	sensor_washing_0	InputLayer	[(None, None, 620)]
5	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
6	tf.cast	TFOpLambda	(None, None, 10)
7	sensor_pressure_0	InputLayer	[(None, None, 45)]
8	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
9	concatenate	Concatenate	(None, None, 1920)
10	encoder	GRU	(None, None, 300)
11	encoder_normalization	BatchNormalization	(None, None, 300)
12	decoder	GRU	(None, None, 200)
13	decoder_normalization	BatchNormalization	(None, None, 200)
14	flatten	Flatten	(None, 6000)
15	id	InputLayer	[(None, None, 17)]
16	product_type_0	InputLayer	[(None, None, 1)]
17	skipped_layer_0	InputLayer	[(None, None, 1)]
18	print_finished_0	InputLayer	[(None, None, 1)]
19	dense	Dense	(None, 17)

A.2.8. Baseline Attribute

	Name	Type	Shape
0	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
1	conv1d_29	Conv1D	(None, None, 10)

2	product_type_0	InputLayer	[(None, None, 1)]
3	sensor_print_temp_0	InputLayer	[(None, None, 300)]
4	sensor_load_curve_0	InputLayer	[(None, None, 600)]
5	skipped_layer_0	InputLayer	[(None, None, 1)]
6	print_finished_0	InputLayer	[(None, None, 1)]
7	sensor_washing_0	InputLayer	[(None, None, 620)]
8	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
9	tf.cast_42	TFOpLambda	(None, None, 10)
10	sensor_pressure_0	InputLayer	[(None, None, 45)]
11	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
12	concatenate_7	Concatenate	(None, None, 1923)
13	encoder	GRU	(None, None, 300)
14	encoder_normalization	BatchNormalization	(None, None, 300)
15	decoder	GRU	(None, None, 200)
16	decoder_normalization	BatchNormalization	(None, None, 200)
17	flatten_7	Flatten	(None, 6000)
18	id	InputLayer	[(None, None, 17)]
19	dense_7	Dense	(None, 17)

A.2.9. Multi Layer CNN no Pooling

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	conv1d_30	Conv1D	(None, None, 100)
8	conv1d_33	Conv1D	(None, None, 200)
9	conv1d_36	Conv1D	(None, None, 206)
10	conv1d_39	Conv1D	(None, None, 100)
11	conv1d_42	Conv1D	(None, None, 440)
12	conv1d_45	Conv1D	(None, None, 15)
13	conv1d_48	Conv1D	(None, None, 15)

14	conv1d_31	Conv1D	(None, None, 50)
15	conv1d_34	Conv1D	(None, None, 100)
16	conv1d_37	Conv1D	(None, None, 103)
17	conv1d_40	Conv1D	(None, None, 50)
18	conv1d_43	Conv1D	(None, None, 220)
19	conv1d_46	Conv1D	(None, None, 7)
20	conv1d_49	Conv1D	(None, None, 7)
21	conv1d_32	Conv1D	(None, None, 25)
22	conv1d_35	Conv1D	(None, None, 50)
23	conv1d_38	Conv1D	(None, None, 51)
24	conv1d_41	Conv1D	(None, None, 25)
25	conv1d_44	Conv1D	(None, None, 110)
26	conv1d_47	Conv1D	(None, None, 3)
27	conv1d_50	Conv1D	(None, None, 3)
28	id	InputLayer	[(None, None, 17)]
29	product_type_0	InputLayer	[(None, None, 1)]
30	tf.cast_43	TFOpLambda	(None, None, 25)
31	tf.cast_44	TFOpLambda	(None, None, 50)
32	skipped_layer_0	InputLayer	[(None, None, 1)]
33	print_finished_0	InputLayer	[(None, None, 1)]
34	tf.cast_45	TFOpLambda	(None, None, 51)
35	tf.cast_46	TFOpLambda	(None, None, 25)
36	tf.cast_47	TFOpLambda	(None, None, 110)
37	tf.cast_48	TFOpLambda	(None, None, 3)
38	tf.cast_49	TFOpLambda	(None, None, 3)
39	concatenate_8	Concatenate	(None, None, 287)
40	encoder	GRU	(None, None, 300)
41	encoder_normalization	BatchNormalization	(None, None, 300)
42	decoder	GRU	(None, None, 200)
43	decoder_normalization	BatchNormalization	(None, None, 200)
44	flatten_8	Flatten	(None, 6000)
45	dense_8	Dense	(None, 17)

A.2.10. Multi Layer FFN

	Name	Type	Shape

0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	dense_9	Dense	(None, None, 50)
8	dense_11	Dense	(None, None, 50)
9	dense_13	Dense	(None, None, 50)
10	dense_15	Dense	(None, None, 50)
11	dense_17	Dense	(None, None, 50)
12	dense_19	Dense	(None, None, 50)
13	dense_21	Dense	(None, None, 50)
14	dense_10	Dense	(None, None, 50)
15	dense_12	Dense	(None, None, 50)
16	dense_14	Dense	(None, None, 50)
17	dense_16	Dense	(None, None, 50)
18	dense_18	Dense	(None, None, 50)
19	dense_20	Dense	(None, None, 50)
20	dense_22	Dense	(None, None, 50)
21	id	InputLayer	[(None, None, 17)]
22	product_type_0	InputLayer	[(None, None, 1)]
23	tf.cast_50	TFOpLambda	(None, None, 50)
24	tf.cast_51	TFOpLambda	(None, None, 50)
25	skipped_layer_0	InputLayer	[(None, None, 1)]
26	print_finished_0	InputLayer	[(None, None, 1)]
27	tf.cast_52	TFOpLambda	(None, None, 50)
28	tf.cast_53	TFOpLambda	(None, None, 50)
29	tf.cast_54	TFOpLambda	(None, None, 50)
30	tf.cast_55	TFOpLambda	(None, None, 50)
31	tf.cast_56	TFOpLambda	(None, None, 50)
32	concatenate_9	Concatenate	(None, None, 370)
33	encoder	GRU	(None, None, 300)
34	encoder_normalization	BatchNormalization	(None, None, 300)
35	decoder	GRU	(None, None, 200)
36	decoder_normalization	BatchNormalization	(None, None, 200)
37	flatten_9	Flatten	(None, 6000)

38	dense_23	Dense	(None, 17)

A.2.11. Normalized RNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	layer_normalization_8	LayerNormalization	(None, None, 300)
8	layer_normalization_9	LayerNormalization	(None, None, 600)
9	layer_normalization_10	LayerNormalization	(None, None, 620)
10	layer_normalization_11	LayerNormalization	(None, None, 300)
11	layer_normalization_12	LayerNormalization	(None, None, 1320)
12	layer_normalization_13	LayerNormalization	(None, None, 45)
13	layer_normalization_14	LayerNormalization	(None, None, 45)
14	gru_4	GRU	(None, None, 10)
15	gru_5	GRU	(None, None, 10)
16	gru_6	GRU	(None, None, 10)
17	gru_7	GRU	(None, None, 10)
18	gru_8	GRU	(None, None, 10)
19	gru_9	GRU	(None, None, 10)
20	gru_10	GRU	(None, None, 10)
21	id	InputLayer	[(None, None, 17)]
22	product_type_0	InputLayer	[(None, None, 1)]
23	lambda_8	Lambda	(None, None, 1)
24	tf.cast_8	TFOpLambda	(None, None, 10)
25	lambda_9	Lambda	(None, None, 1)
26	tf.cast_9	TFOpLambda	(None, None, 10)
27	skipped_layer_0	InputLayer	[(None, None, 1)]
28	print_finished_0	InputLayer	[(None, None, 1)]
29	lambda_10	Lambda	(None, None, 1)
30	tf.cast_10	TFOpLambda	(None, None, 10)

31	lambda_11	Lambda	(None, None, 1)
32	tf.cast_11	TFOpLambda	(None, None, 10)
33	lambda_12	Lambda	(None, None, 1)
34	tf.cast_12	TFOpLambda	(None, None, 10)
35	lambda_13	Lambda	(None, None, 1)
36	tf.cast_13	TFOpLambda	(None, None, 10)
37	lambda_14	Lambda	(None, None, 1)
38	tf.cast_14	TFOpLambda	(None, None, 10)
39	concatenate_2	Concatenate	(None, None, 97)
40	encoder	GRU	(None, None, 300)
41	encoder_normalization	BatchNormalization	(None, None, 300)
42	decoder	GRU	(None, None, 200)
43	decoder_normalization	BatchNormalization	(None, None, 200)
44	flatten_2	Flatten	(None, 6000)
45	dense_2	Dense	(None, 17)

A.2.12. Normalized CNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	layer_normalization_15	LayerNormalization	(None, None, 300)
8	layer_normalization_16	LayerNormalization	(None, None, 600)
9	layer_normalization_17	LayerNormalization	(None, None, 620)
10	layer_normalization_18	LayerNormalization	(None, None, 300)
11	layer_normalization_19	LayerNormalization	(None, None, 1320)
12	layer_normalization_20	LayerNormalization	(None, None, 45)
13	layer_normalization_21	LayerNormalization	(None, None, 45)
14	conv1d_4	Conv1D	(None, None, 10)
15	conv1d_5	Conv1D	(None, None, 10)
16	conv1d_6	Conv1D	(None, None, 10)

17	conv1d_7	Conv1D	(None, None, 10)
18	conv1d_8	Conv1D	(None, None, 10)
19	conv1d_9	Conv1D	(None, None, 10)
20	conv1d_10	Conv1D	(None, None, 10)
21	id	InputLayer	[(None, None, 17)]
22	product_type_0	InputLayer	[(None, None, 1)]
23	lambda_15	Lambda	(None, None, 1)
24	tf.cast_15	TFOpLambda	(None, None, 10)
25	lambda_16	Lambda	(None, None, 1)
26	tf.cast_16	TFOpLambda	(None, None, 10)
27	skipped_layer_0	InputLayer	[(None, None, 1)]
28	print_finished_0	InputLayer	[(None, None, 1)]
29	lambda_17	Lambda	(None, None, 1)
30	tf.cast_17	TFOpLambda	(None, None, 10)
31	lambda_18	Lambda	(None, None, 1)
32	tf.cast_18	TFOpLambda	(None, None, 10)
33	lambda_19	Lambda	(None, None, 1)
34	tf.cast_19	TFOpLambda	(None, None, 10)
35	lambda_20	Lambda	(None, None, 1)
36	tf.cast_20	TFOpLambda	(None, None, 10)
37	lambda_21	Lambda	(None, None, 1)
38	tf.cast_21	TFOpLambda	(None, None, 10)
39	concatenate_3	Concatenate	(None, None, 97)
40	encoder	GRU	(None, None, 300)
41	encoder_normalization	BatchNormalization	(None, None, 300)
42	decoder	GRU	(None, None, 200)
43	decoder_normalization	BatchNormalization	(None, None, 200)
44	flatten_3	Flatten	(None, 6000)
45	dense_3	Dense	(None, 17)

A.2.13. Kalman

	Name	Type	Shape
0	id	InputLayer	[(None, None, 17)]
1	product_type_0	InputLayer	[(None, None, 1)]
2	sensor_print_temp_0	InputLayer	[(None, None, 1)]

3	sensor_load_curve_0	InputLayer	[(None, None, 1)]
4	skipped_layer_0	InputLayer	[(None, None, 1)]
5	print_finished_0	InputLayer	[(None, None, 1)]
6	sensor_washing_0	InputLayer	[(None, None, 1)]
7	sensor_cure_energy_0	InputLayer	[(None, None, 1)]
8	sensor_dry_temp_0	InputLayer	[(None, None, 1)]
9	sensor_pressure_0	InputLayer	[(None, None, 1)]
10	sensor_poly_temp_0	InputLayer	[(None, None, 1)]
11	concatenate	Concatenate	(None, None, 27)
12	encoder	GRU	(None, None, 300)
13	encoder_normalization	BatchNormalization	(None, None, 300)
14	decoder	GRU	(None, None, 200)
15	decoder_normalization	BatchNormalization	(None, None, 200)
16	flatten	Flatten	(None, 6000)
17	dense	Dense	(None, 17)

A.2.14. Kalman RNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	gru_4	GRU	(None, None, 10)
8	gru_5	GRU	(None, None, 10)
9	gru_6	GRU	(None, None, 10)
10	gru_7	GRU	(None, None, 10)
11	gru_8	GRU	(None, None, 10)
12	gru_9	GRU	(None, None, 10)
13	gru_10	GRU	(None, None, 10)
14	id	InputLayer	[(None, None, 17)]
15	product_type_0	InputLayer	[(None, None, 1)]
16	kalman_print_temp_0	InputLayer	[(None, None, 1)]

17	tf.cast_16	TFOpLambda	(None, None, 10)
18	kalman_load_curve_0	InputLayer	[(None, None, 1)]
19	tf.cast_17	TFOpLambda	(None, None, 10)
20	skipped_layer_0	InputLayer	[(None, None, 1)]
21	print_finished_0	InputLayer	[(None, None, 1)]
22	kalman_washing_0	InputLayer	[(None, None, 1)]
23	tf.cast_18	TFOpLambda	(None, None, 10)
24	kalman_cure_energy_0	InputLayer	[(None, None, 1)]
25	tf.cast_19	TFOpLambda	(None, None, 10)
26	kalman_dry_temp_0	InputLayer	[(None, None, 1)]
27	tf.cast_20	TFOpLambda	(None, None, 10)
28	kalman_pressure_0	InputLayer	[(None, None, 1)]
29	tf.cast_21	TFOpLambda	(None, None, 10)
30	kalman_poly_temp_0	InputLayer	[(None, None, 1)]
31	tf.cast_22	TFOpLambda	(None, None, 10)
32	concatenate_3	Concatenate	(None, None, 97)
33	encoder	GRU	(None, None, 300)
34	encoder_normalization	BatchNormalization	(None, None, 300)
35	decoder	GRU	(None, None, 200)
36	decoder_normalization	BatchNormalization	(None, None, 200)
37	flatten_3	Flatten	(None, 6000)
38	dense_3	Dense	(None, 17)

A.2.15. Kalman CNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	conv1d_8	Conv1D	(None, None, 10)
8	conv1d_9	Conv1D	(None, None, 10)
9	conv1d_10	Conv1D	(None, None, 10)

10	conv1d_11	Conv1D	(None, None, 10)
11	conv1d_12	Conv1D	(None, None, 10)
12	conv1d_13	Conv1D	(None, None, 10)
13	conv1d_14	Conv1D	(None, None, 10)
14	id	InputLayer	[(None, None, 17)]
15	product_type_0	InputLayer	[(None, None, 1)]
16	kalman_print_temp_0	InputLayer	[(None, None, 1)]
17	tf.cast_23	TFOpLambda	(None, None, 10)
18	kalman_load_curve_0	InputLayer	[(None, None, 1)]
19	tf.cast_24	TFOpLambda	(None, None, 10)
20	skipped_layer_0	InputLayer	[(None, None, 1)]
21	print_finished_0	InputLayer	[(None, None, 1)]
22	kalman_washing_0	InputLayer	[(None, None, 1)]
23	tf.cast_25	TFOpLambda	(None, None, 10)
24	kalman_cure_energy_0	InputLayer	[(None, None, 1)]
25	tf.cast_26	TFOpLambda	(None, None, 10)
26	kalman_dry_temp_0	InputLayer	[(None, None, 1)]
27	tf.cast_27	TFOpLambda	(None, None, 10)
28	kalman_pressure_0	InputLayer	[(None, None, 1)]
29	tf.cast_28	TFOpLambda	(None, None, 10)
30	kalman_poly_temp_0	InputLayer	[(None, None, 1)]
31	tf.cast_29	TFOpLambda	(None, None, 10)
32	concatenate_4	Concatenate	(None, None, 97)
33	encoder	GRU	(None, None, 300)
34	encoder_normalization	BatchNormalization	(None, None, 300)
35	decoder	GRU	(None, None, 200)
36	decoder_normalization	BatchNormalization	(None, None, 200)
37	flatten_4	Flatten	(None, 6000)
38	dense_4	Dense	(None, 17)

A.2.16. Kalman Normalized RNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]

3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	layer_normalization_8	LayerNormalization	(None, None, 300)
8	layer_normalization_9	LayerNormalization	(None, None, 600)
9	layer_normalization_10	LayerNormalization	(None, None, 620)
10	layer_normalization_11	LayerNormalization	(None, None, 300)
11	layer_normalization_12	LayerNormalization	(None, None, 1320)
12	layer_normalization_13	LayerNormalization	(None, None, 45)
13	layer_normalization_14	LayerNormalization	(None, None, 45)
14	gru_11	GRU	(None, None, 10)
15	gru_12	GRU	(None, None, 10)
16	gru_13	GRU	(None, None, 10)
17	gru_14	GRU	(None, None, 10)
18	gru_15	GRU	(None, None, 10)
19	gru_16	GRU	(None, None, 10)
20	gru_17	GRU	(None, None, 10)
21	id	InputLayer	[(None, None, 17)]
22	product_type_0	InputLayer	[(None, None, 1)]
23	kalman_print_temp_0	InputLayer	[(None, None, 1)]
24	lambda_8	Lambda	(None, None, 1)
25	tf.cast_30	TFOpLambda	(None, None, 10)
26	kalman_load_curve_0	InputLayer	[(None, None, 1)]
27	lambda_9	Lambda	(None, None, 1)
28	tf.cast_31	TFOpLambda	(None, None, 10)
29	skipped_layer_0	InputLayer	[(None, None, 1)]
30	print_finished_0	InputLayer	[(None, None, 1)]
31	kalman_washing_0	InputLayer	[(None, None, 1)]
32	lambda_10	Lambda	(None, None, 1)
33	tf.cast_32	TFOpLambda	(None, None, 10)
34	kalman_cure_energy_0	InputLayer	[(None, None, 1)]
35	lambda_11	Lambda	(None, None, 1)
36	tf.cast_33	TFOpLambda	(None, None, 10)
37	kalman_dry_temp_0	InputLayer	[(None, None, 1)]
38	lambda_12	Lambda	(None, None, 1)
39	tf.cast_34	TFOpLambda	(None, None, 10)
40	kalman_pressure_0	InputLayer	[(None, None, 1)]

41	lambda_13	Lambda	(None, None, 1)
42	tf.cast_35	TFOpLambda	(None, None, 10)
43	kalman_poly_temp_0	InputLayer	[(None, None, 1)]
44	lambda_14	Lambda	(None, None, 1)
45	tf.cast_36	TFOpLambda	(None, None, 10)
46	concatenate_5	Concatenate	(None, None, 104)
47	encoder	GRU	(None, None, 300)
48	encoder_normalization	BatchNormalization	(None, None, 300)
49	decoder	GRU	(None, None, 200)
50	decoder_normalization	BatchNormalization	(None, None, 200)
51	flatten_5	Flatten	(None, 6000)
52	dense_5	Dense	(None, 17)

A.2.17. Kalman Normalized CNN

	Name	Type	Shape
0	sensor_print_temp_0	InputLayer	[(None, None, 300)]
1	sensor_load_curve_0	InputLayer	[(None, None, 600)]
2	sensor_washing_0	InputLayer	[(None, None, 620)]
3	sensor_cure_energy_0	InputLayer	[(None, None, 300)]
4	sensor_dry_temp_0	InputLayer	[(None, None, 1320)]
5	sensor_pressure_0	InputLayer	[(None, None, 45)]
6	sensor_poly_temp_0	InputLayer	[(None, None, 45)]
7	layer_normalization	LayerNormalization	(None, None, 300)
8	layer_normalization_1	LayerNormalization	(None, None, 600)
9	layer_normalization_2	LayerNormalization	(None, None, 620)
10	layer_normalization_3	LayerNormalization	(None, None, 300)
11	layer_normalization_4	LayerNormalization	(None, None, 1320)
12	layer_normalization_5	LayerNormalization	(None, None, 45)
13	layer_normalization_6	LayerNormalization	(None, None, 45)
14	conv1d	Conv1D	(None, None, 10)
15	conv1d_1	Conv1D	(None, None, 10)
16	conv1d_2	Conv1D	(None, None, 10)
17	conv1d_3	Conv1D	(None, None, 10)
18	conv1d_4	Conv1D	(None, None, 10)
19	conv1d_5	Conv1D	(None, None, 10)

20	conv1d_6	Conv1D	(None, None, 10)
21	id	InputLayer	[(None, None, 17)]
22	product_type_0	InputLayer	[(None, None, 1)]
23	kalman_print_temp_0	InputLayer	[(None, None, 1)]
24	lambda	Lambda	(None, None, 1)
25	tf.cast	TFOpLambda	(None, None, 10)
26	kalman_load_curve_0	InputLayer	[(None, None, 1)]
27	lambda_1	Lambda	(None, None, 1)
28	tf.cast_1	TFOpLambda	(None, None, 10)
29	skipped_layer_0	InputLayer	[(None, None, 1)]
30	print_finished_0	InputLayer	[(None, None, 1)]
31	kalman_washing_0	InputLayer	[(None, None, 1)]
32	lambda_2	Lambda	(None, None, 1)
33	tf.cast_2	TFOpLambda	(None, None, 10)
34	kalman_cure_energy_0	InputLayer	[(None, None, 1)]
35	lambda_3	Lambda	(None, None, 1)
36	tf.cast_3	TFOpLambda	(None, None, 10)
37	kalman_dry_temp_0	InputLayer	[(None, None, 1)]
38	lambda_4	Lambda	(None, None, 1)
39	tf.cast_4	TFOpLambda	(None, None, 10)
40	kalman_pressure_0	InputLayer	[(None, None, 1)]
41	lambda_5	Lambda	(None, None, 1)
42	tf.cast_5	TFOpLambda	(None, None, 10)
43	kalman_poly_temp_0	InputLayer	[(None, None, 1)]
44	lambda_6	Lambda	(None, None, 1)
45	tf.cast_6	TFOpLambda	(None, None, 10)
46	concatenate	Concatenate	(None, None, 104)
47	encoder	GRU	(None, None, 300)
48	encoder_normalization	BatchNormalization	(None, None, 300)
49	decoder	GRU	(None, None, 200)
50	decoder_normalization	BatchNormalization	(None, None, 200)
51	flatten	Flatten	(None, 6000)
52	dense	Dense	(None, 17)

A.3. Process Prediction

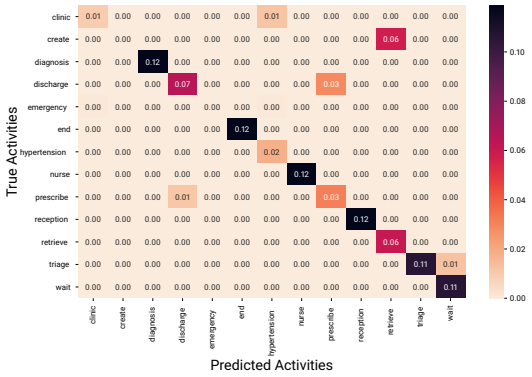
A.3.1. Experiment Results on the Train Datasets with 6000 Instances and no Anomalies

Table A.1.: Performance of the model with different model configurations evaluated on L6000 and H6000 without anomalies.

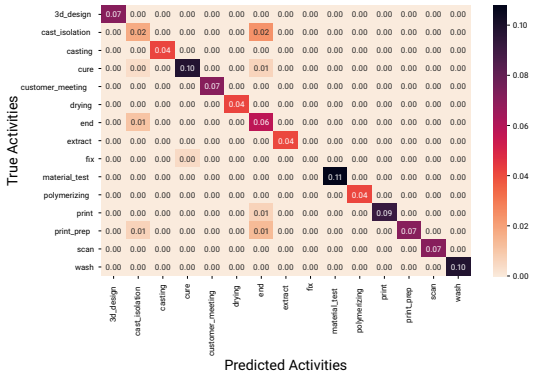
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000	rnn	0.9775	0.9755	0.9758	0.9775	0.9754
	normalize	0.9756	0.9735	0.9727	0.9756	0.9735
	ml_ffn	0.9764	0.9744	0.9745	0.9764	0.9743
	ml_cnn_no_pool	0.9369	0.9314	0.9409	0.9369	0.9361
	ml_cnn	0.9388	0.9335	0.9419	0.9388	0.9380
	kalman	0.9629	0.9597	0.9663	0.9629	0.9615
	default	0.9727	0.9703	0.9739	0.9727	0.9706
	convolution	0.9727	0.9703	0.9707	0.9727	0.9709
	baseline_sensor	0.6395	0.6057	0.6328	0.6395	0.5807
	baseline_id	0.9216	0.9148	0.9350	0.9216	0.9239
	baseline_attr	0.7527	0.7320	0.7703	0.7527	0.7282
H6000	rnn	0.9031	0.8922	0.8729	0.9031	0.8819
	normalize	0.9021	0.8912	0.8721	0.9021	0.8812
	ml_ffn	0.8702	0.8554	0.8174	0.8702	0.8320
	ml_cnn_no_pool	0.8295	0.8097	0.7717	0.8295	0.7935
	ml_cnn	0.8294	0.8096	0.7715	0.8294	0.7933
	kalman	0.8876	0.8748	0.8589	0.8876	0.8635
	default	0.8703	0.8555	0.8175	0.8703	0.8320
	convolution	0.8972	0.8856	0.8691	0.8972	0.8754
	baseline_sensor	0.5866	0.5380	0.6753	0.5866	0.5719
	baseline_id	0.8693	0.8544	0.8134	0.8693	0.8303
	baseline_attr	0.7541	0.7252	0.7069	0.7541	0.7160

Table A.2.: Performance of the model with different model configurations evaluated on **the decisions** of L6000 and H6000 without anomalies.

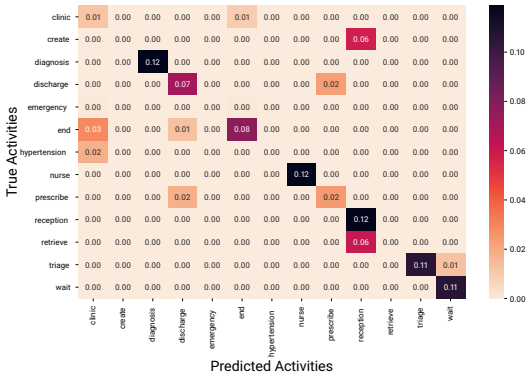
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Decisions	rnn	0.9087	0.8699	0.9001	0.9087	0.8956
	normalize	0.9012	0.8581	0.8815	0.9012	0.8839
	ml_ffn	0.9045	0.8627	0.8879	0.9045	0.8839
	ml_cnn_no_pool	0.7443	0.6239	0.7594	0.7443	0.7180
	ml_cnn	0.7520	0.6375	0.7612	0.7520	0.7329
	kalman	0.8497	0.7836	0.8310	0.8497	0.8164
	default	0.8894	0.8409	0.8896	0.8894	0.8658
	convolution	0.8900	0.8428	0.8719	0.8900	0.8751
	baseline_sensor	0.7509	0.6364	0.7633	0.7509	0.7243
	baseline_id	0.6823	0.5349	0.6452	0.6823	0.6426
	baseline_attr	0.8901	0.8421	0.8869	0.8901	0.8678
H6000 Decisions	rnn	0.7575	0.7063	0.6826	0.7575	0.7040
	normalize	0.7553	0.7043	0.6781	0.7553	0.7008
	ml_ffn	0.6753	0.6020	0.5168	0.6753	0.5666
	ml_cnn_no_pool	0.6064	0.5263	0.5068	0.6064	0.5427
	ml_cnn	0.6063	0.5262	0.5066	0.6063	0.5426
	kalman	0.7187	0.6562	0.6180	0.7187	0.6445
	default	0.6754	0.6022	0.5172	0.6754	0.5668
	convolution	0.7427	0.6868	0.6423	0.7427	0.6738
	baseline_sensor	0.3628	0.3013	0.4466	0.3628	0.3855
	baseline_id	0.6731	0.5992	0.5069	0.6731	0.5624
	baseline_attr	0.4904	0.4142	0.4563	0.4904	0.4714



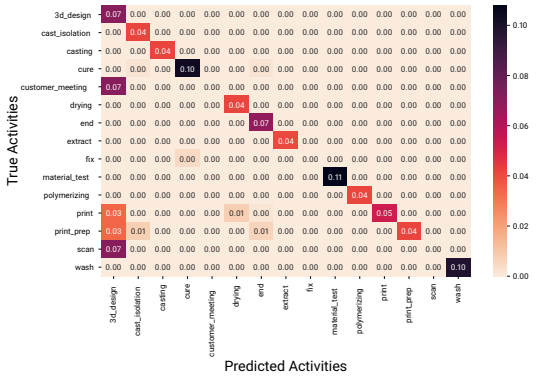
(a) Baseline ID Hospital



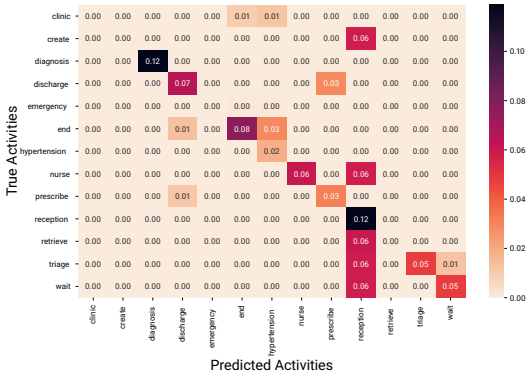
(b) Baseline ID Laboratory



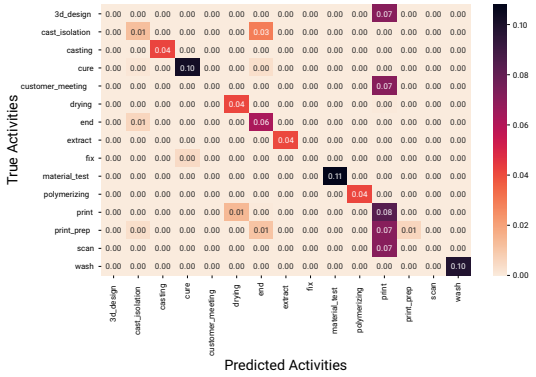
(c) Baseline Attribute Hospital



(d) Baseline Attribute Laboratory

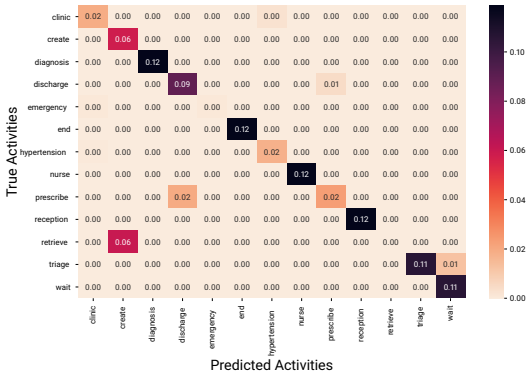


(e) Baseline Sensor Hospital

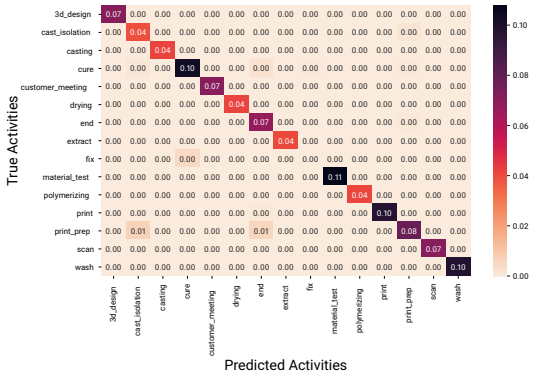


(f) Baseline Sensor Laboratory

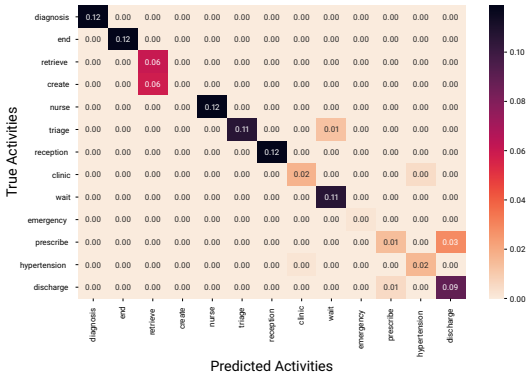
Figure A.1.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances and without Anomalies.



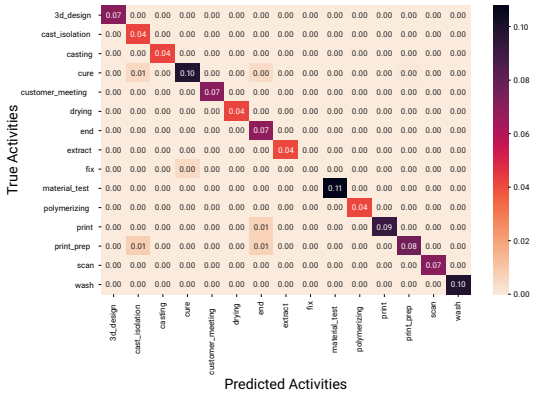
(g) Convolution Hospital



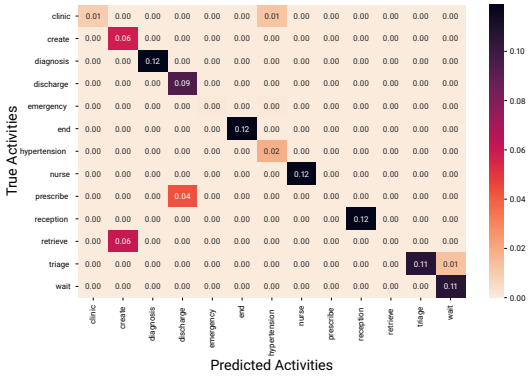
(h) Convolution Laboratory



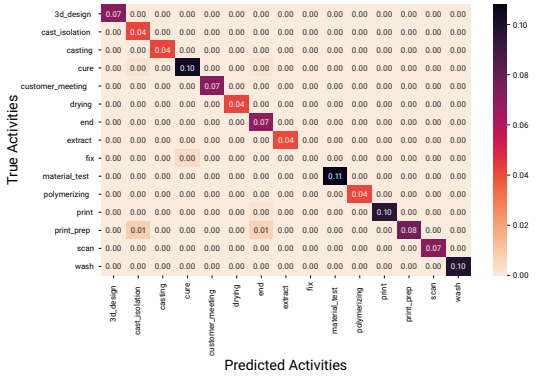
(i) Kalman Hospital



(j) Kalman Laboratory

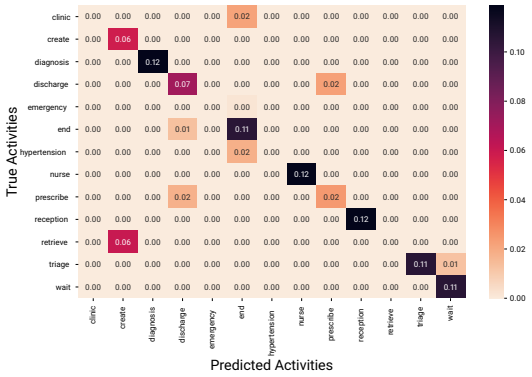


(k) Default Hospital

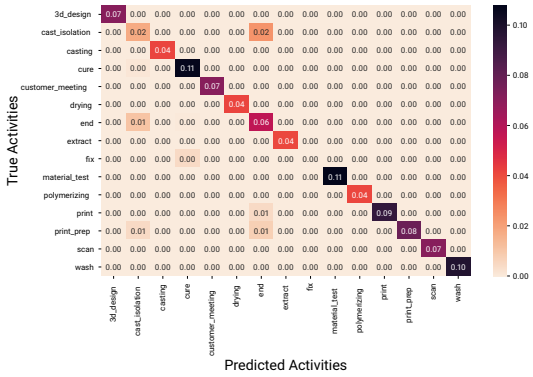


(l) Default Laboratory

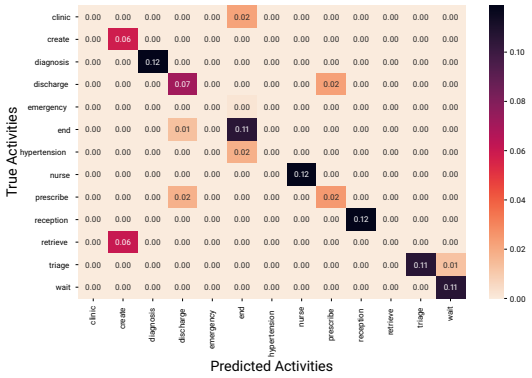
Figure A.1.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances and without Anomalies (cont.).



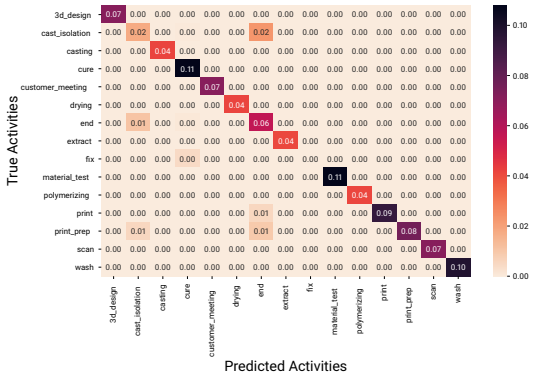
(m) ML CNN Hospital



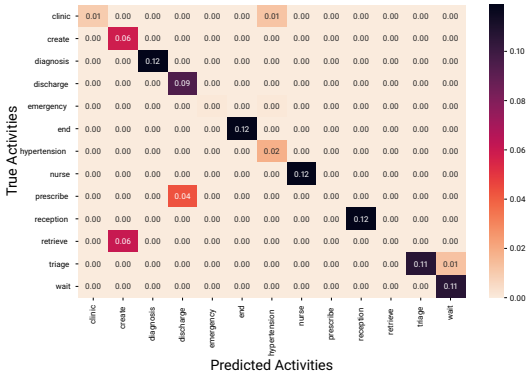
(n) ML CNN Laboratory



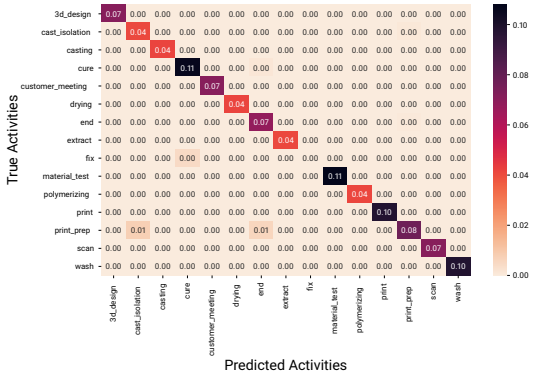
(o) ML CNN no Pooling Hospital



(p) ML CNN no Pooling Laboratory

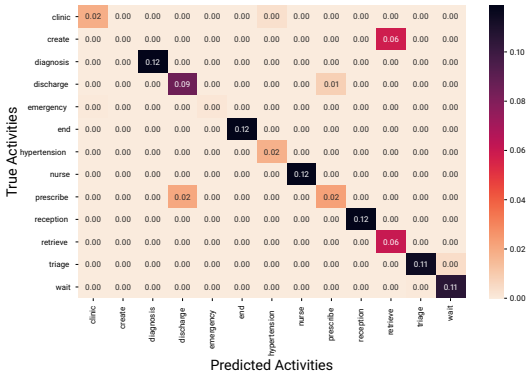


(q) ML FFN Hospital

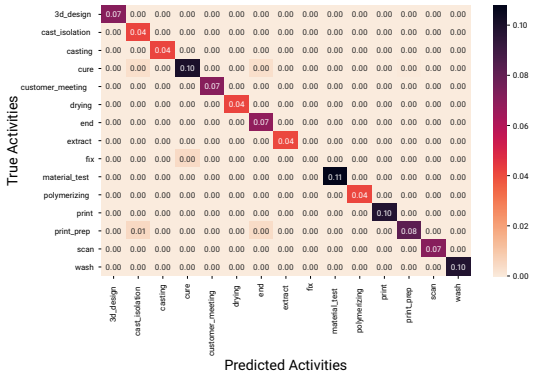


(r) ML FFN Laboratory

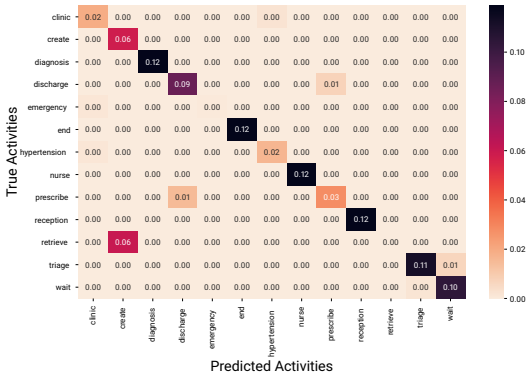
Figure A.1.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances and without Anomalies (cont.).



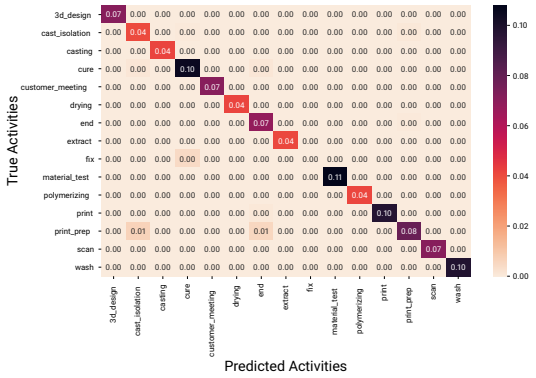
(s) RNN Hospital



(t) RNN Laboratory



(u) Normalize Hospital



(v) Normalize Laboratory

Figure A.1.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances and without Anomalies (cont.).

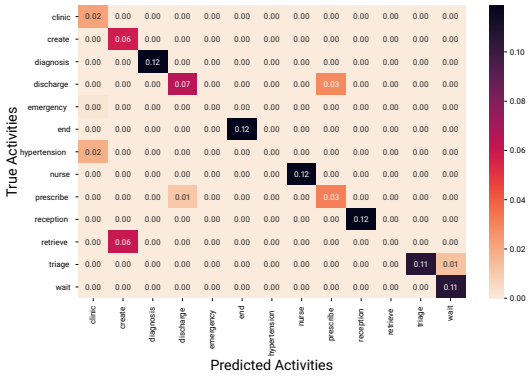
A.3.2. Experiment Results on the Train Datasets with 6000 Instances and 30% Sensor Anomalies

Table A.3.: Performance of the model with different model configurations evaluated on L6000 and H6000 with 30% sensor anomalies.

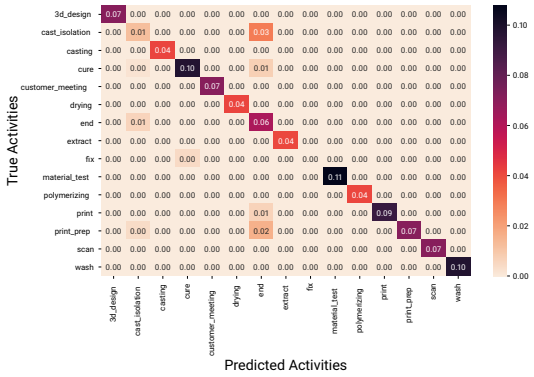
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000	rnn	0.9749	0.9727	0.9730	0.9749	0.9727
	normalize	0.9666	0.9637	0.9646	0.9666	0.9652
	ml_ffn	0.9758	0.9737	0.9742	0.9758	0.9735
	ml_cnn_no_pool	0.9370	0.9315	0.9441	0.9370	0.9313
	ml_cnn	0.9390	0.9337	0.9436	0.9390	0.9335
	kalman	0.9630	0.9598	0.9696	0.9630	0.9616
	default	0.9754	0.9732	0.9739	0.9754	0.9731
	convolution	0.9791	0.9772	0.9758	0.9791	0.9769
	baseline_sensor	0.6410	0.6114	0.7198	0.6410	0.5904
	baseline_id	0.9209	0.9141	0.9336	0.9209	0.9211
	baseline_attr	0.7555	0.7349	0.7724	0.7555	0.7306
H6000	rnn	0.9028	0.8919	0.8723	0.9028	0.8819
	normalize	0.9055	0.8949	0.8754	0.9055	0.8854
	ml_ffn	0.8664	0.8517	0.8346	0.8664	0.8414
	ml_cnn_no_pool	0.8307	0.8106	0.7693	0.8307	0.7863
	ml_cnn	0.8572	0.8403	0.8180	0.8572	0.8187
	kalman	0.8810	0.8676	0.8557	0.8810	0.8590
	default	0.8649	0.8496	0.8218	0.8649	0.8300
	convolution	0.9022	0.8911	0.8737	0.9022	0.8806
	baseline_sensor	0.5953	0.5433	0.6429	0.5953	0.5581
	baseline_id	0.8656	0.8508	0.8324	0.8656	0.8400
	baseline_attr	0.7715	0.7431	0.6864	0.7715	0.7164

Table A.4.: Performance of the model with different model configurations evaluated on **the decisions** of L6000 and H6000 with 30% sensor anomalies.

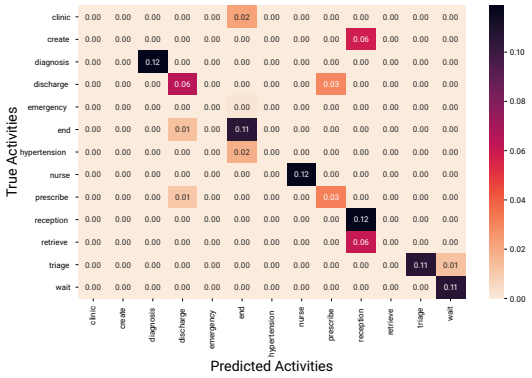
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Decisions	rnn	0.8982	0.8519	0.8894	0.8982	0.8824
	normalize	0.8648	0.8084	0.8478	0.8648	0.8540
	ml_ffn	0.9021	0.8589	0.8851	0.9021	0.8779
	ml_cnn_no_pool	0.7447	0.6196	0.7711	0.7447	0.7000
	ml_cnn	0.7531	0.6339	0.7664	0.7531	0.7155
	kalman	0.8501	0.7842	0.8447	0.8501	0.8168
	default	0.9002	0.8560	0.8838	0.9002	0.8775
	convolution	0.9151	0.8779	0.8972	0.9151	0.9013
	baseline_sensor	0.7567	0.6427	0.7621	0.7567	0.7289
	baseline_id	0.6797	0.5269	0.6268	0.6797	0.6312
	baseline_attr	0.9028	0.8593	0.8991	0.9028	0.8782
H6000 Decisions	rnn	0.7568	0.7066	0.6810	0.7568	0.7045
	normalize	0.7636	0.7152	0.6894	0.7636	0.7139
	ml_ffn	0.6658	0.5962	0.5526	0.6658	0.5869
	ml_cnn_no_pool	0.6094	0.5247	0.4970	0.6094	0.5252
	ml_cnn	0.6580	0.5856	0.6042	0.6580	0.5900
	kalman	0.7022	0.6372	0.6088	0.7022	0.6330
	default	0.6621	0.5868	0.5264	0.6621	0.5612
	convolution	0.7574	0.7058	0.6799	0.7574	0.7010
	baseline_sensor	0.3119	0.2403	0.3925	0.3119	0.3247
	baseline_id	0.6638	0.5938	0.5470	0.6638	0.5833
	baseline_attr	0.4614	0.3858	0.4461	0.4614	0.4485



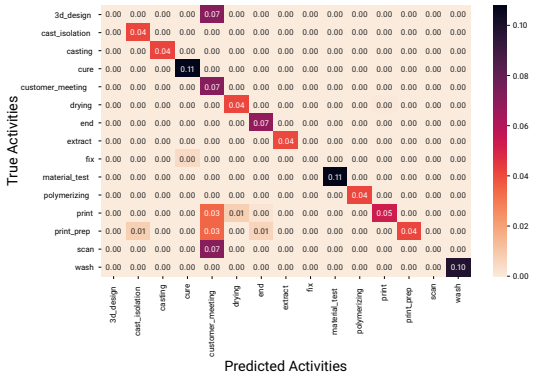
(a) Baseline ID Hospital



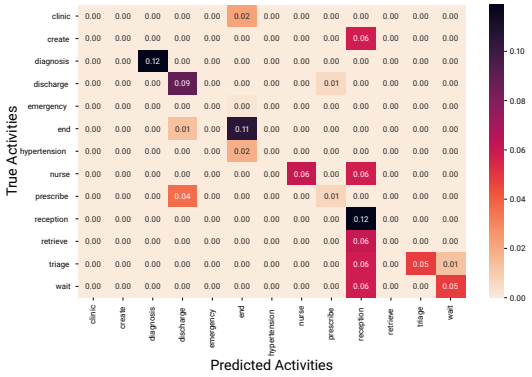
(b) Baseline ID Laboratory



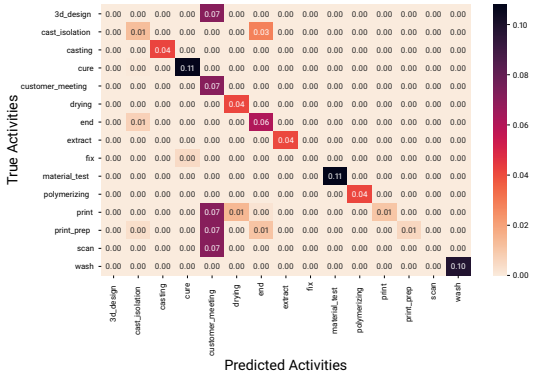
(c) Baseline Attribute Hospital



(d) Baseline Attribute Laboratory

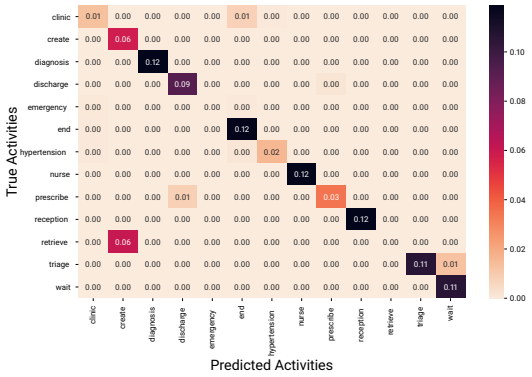


(e) Baseline Sensor Hospital

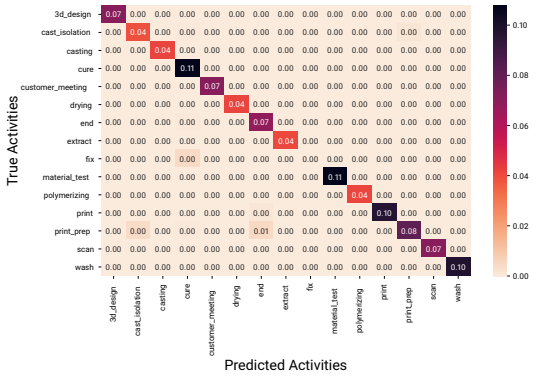


(f) Baseline Sensor Laboratory

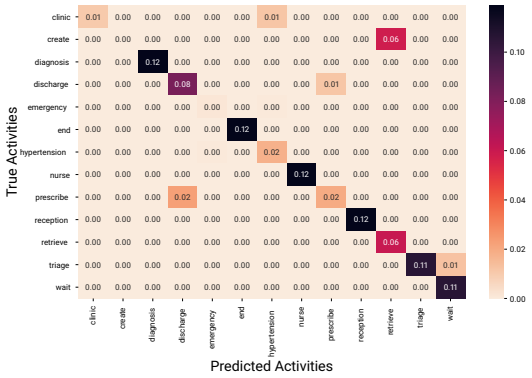
Figure A.2.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies.



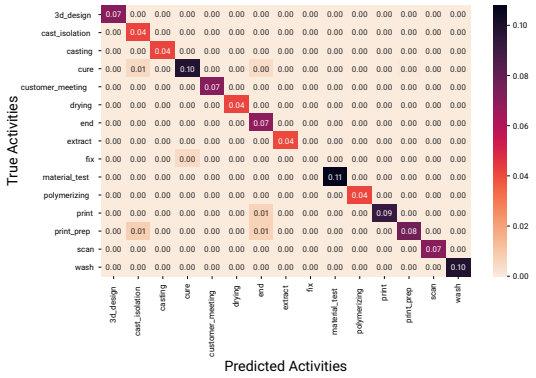
(g) Convolution Hospital



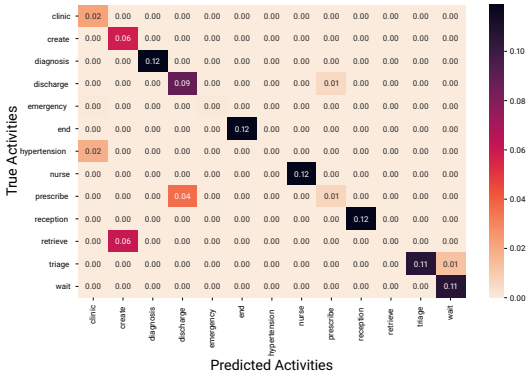
(h) Convolution Laboratory



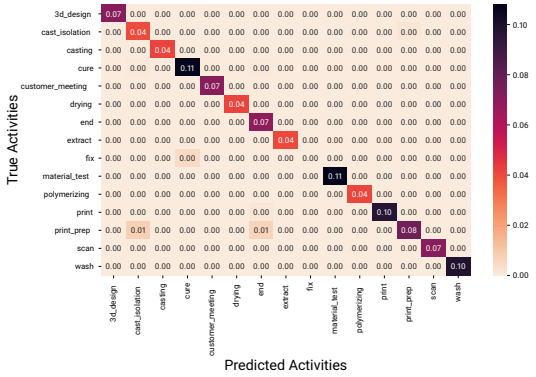
(i) Kalman Hospital



(j) Kalman Laboratory

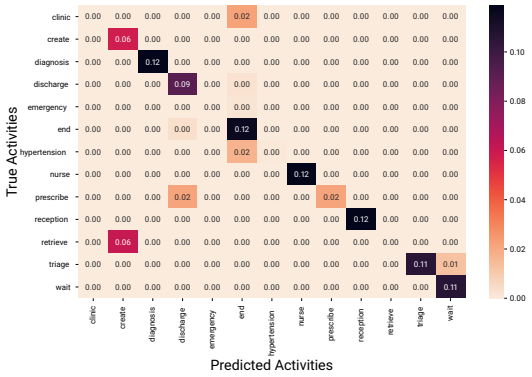


(k) Default Hospital

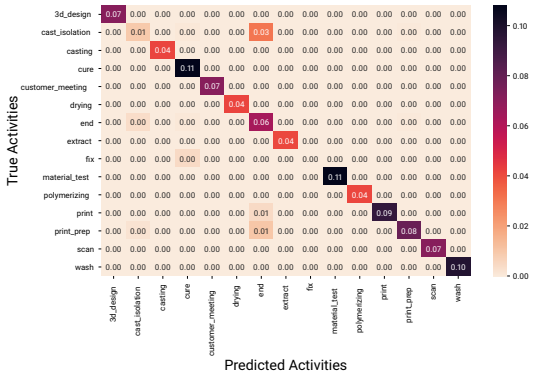


(l) Default Laboratory

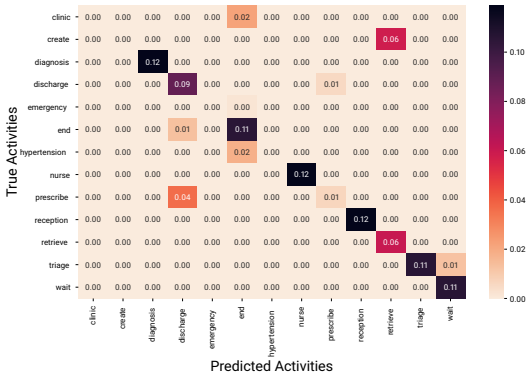
Figure A.2.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies (cont.).



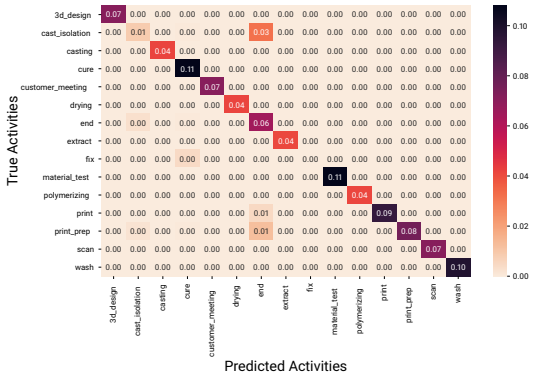
(m) ML CNN Hospital



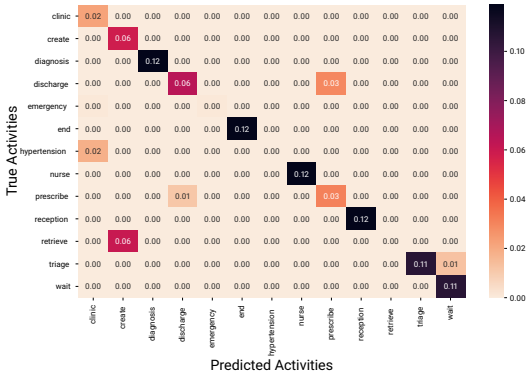
(n) ML CNN Laboratory



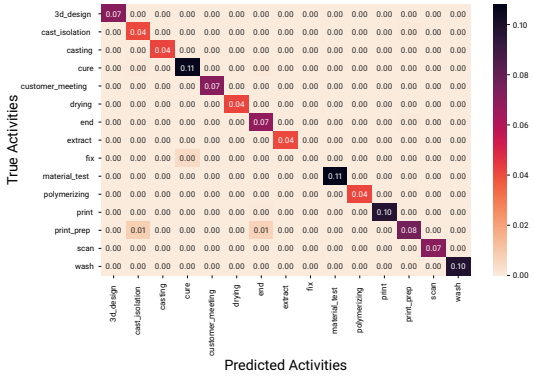
(o) ML CNN no Pooling Hospital



(p) ML CNN no Pooling Laboratory

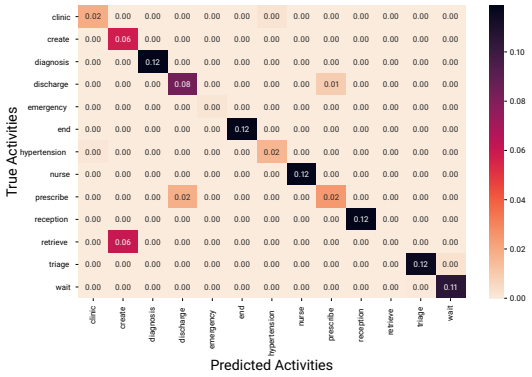


(q) ML FFN Hospital

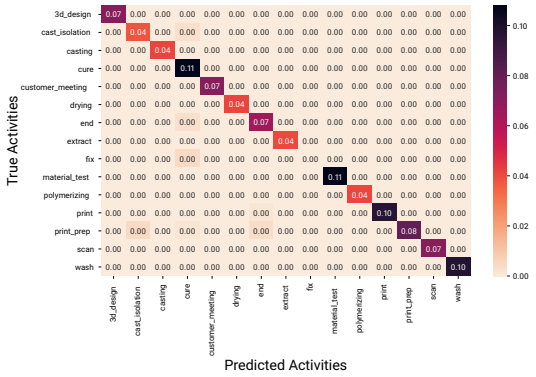


(r) ML FFN Laboratory

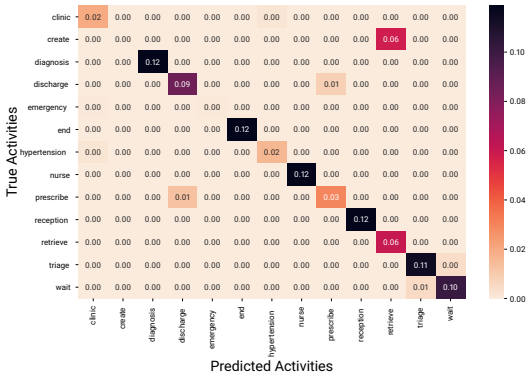
Figure A.2.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies (cont.).



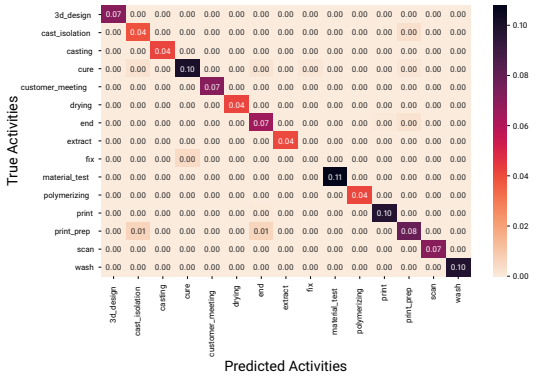
(s) RNN Hospital



(t) RNN Laboratory



(u) Normalize Hospital



(v) Normalize Laboratory

Figure A.2.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies (cont.).

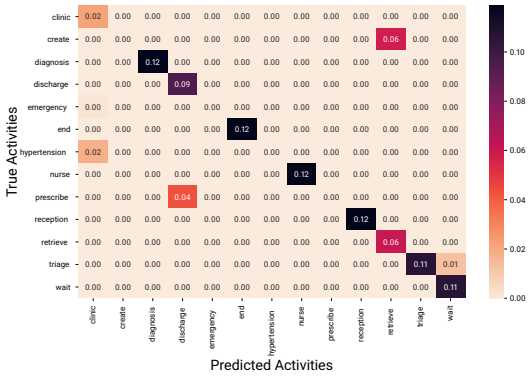
A.3.3. Experiment Results on the Train Datasets with 6000 Instances and 30% Flow Anomalies

Table A.5.: Performance of the model with different model configurations evaluated on L6000 and H6000 with 30% flow anomalies.

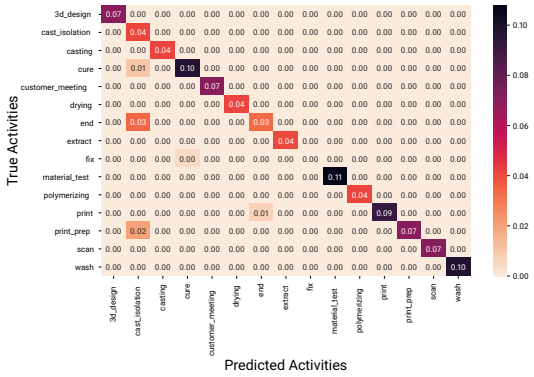
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000	rnn	0.9774	0.9754	0.9739	0.9774	0.9752
	normalize	0.9607	0.9573	0.9589	0.9607	0.9590
	ml_ffn	0.9757	0.9736	0.9739	0.9757	0.9735
	ml_cnn_no_pool	0.9369	0.9315	0.9561	0.9369	0.9378
	ml_cnn	0.9381	0.9328	0.9498	0.9381	0.9378
	kalman	0.9596	0.9562	0.9621	0.9596	0.9583
	default	0.9770	0.9750	0.9759	0.9770	0.9747
	convolution	0.9797	0.9779	0.9790	0.9797	0.9774
	baseline_sensor	0.6421	0.6124	0.7052	0.6421	0.5938
	baseline_id	0.9216	0.9150	0.9518	0.9216	0.9264
	baseline_attr	0.7563	0.7357	0.7709	0.7563	0.7323
H6000	rnn	0.9008	0.8897	0.8713	0.9008	0.8811
	normalize	0.8898	0.8774	0.8612	0.8898	0.8699
	ml_ffn	0.8665	0.8512	0.7992	0.8665	0.8229
	ml_cnn_no_pool	0.8764	0.8621	0.8306	0.8764	0.8441
	ml_cnn	0.8287	0.8097	0.8015	0.8287	0.7977
	kalman	0.8825	0.8694	0.8558	0.8825	0.8629
	default	0.8719	0.8572	0.8153	0.8719	0.8346
	convolution	0.8756	0.8614	0.8296	0.8756	0.8455
	baseline_sensor	0.5919	0.5404	0.6519	0.5919	0.5607
	baseline_id	0.8654	0.8500	0.7968	0.8654	0.8212
	baseline_attr	0.7659	0.7366	0.6879	0.7659	0.7081

Table A.6.: Performance of the model with different model configurations evaluated on **the decisions** of L6000 and H6000 with 30% flow anomalies.

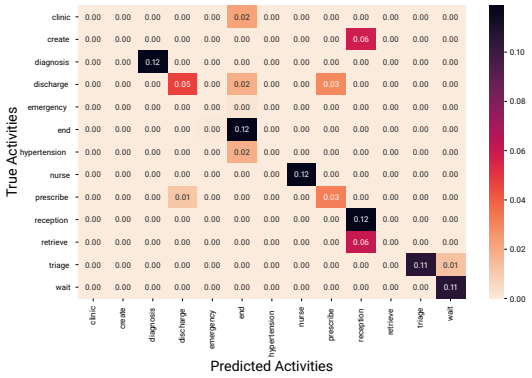
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Decisions	rnn	0.9090	0.8702	0.8912	0.9090	0.8961
	normalize	0.8431	0.7773	0.8278	0.8431	0.8318
	ml_ffn	0.9022	0.8589	0.8849	0.9022	0.8800
	ml_cnn_no_pool	0.7443	0.6397	0.8076	0.7443	0.7302
	ml_cnn	0.7492	0.6454	0.7816	0.7492	0.7346
	kalman	0.8368	0.7665	0.8327	0.8368	0.8123
	default	0.9069	0.8658	0.8977	0.9069	0.8837
	convolution	0.9177	0.8820	0.9115	0.9177	0.9006
	baseline_sensor	0.7614	0.6485	0.7475	0.7614	0.7290
	baseline_id	0.6823	0.5550	0.7166	0.6823	0.6557
	baseline_attr	0.9064	0.8648	0.8935	0.9064	0.8849
H6000 Decisions	rnn	0.7528	0.7018	0.6790	0.7528	0.7033
	normalize	0.7263	0.6726	0.6689	0.7263	0.6826
	ml_ffn	0.6661	0.5895	0.4715	0.6661	0.5440
	ml_cnn_no_pool	0.7052	0.6413	0.6035	0.7052	0.6333
	ml_cnn	0.5976	0.5190	0.5393	0.5976	0.5181
	kalman	0.7070	0.6442	0.6083	0.7070	0.6428
	default	0.6795	0.6062	0.5117	0.6795	0.5733
	convolution	0.6982	0.6334	0.5931	0.6982	0.6290
	baseline_sensor	0.3167	0.2537	0.4197	0.3167	0.3515
	baseline_id	0.6632	0.5859	0.4652	0.6632	0.5395
	baseline_attr	0.4626	0.3871	0.4526	0.4626	0.4498



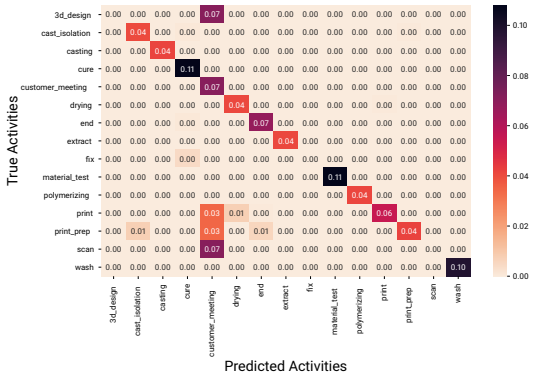
(a) Baseline ID Hospital



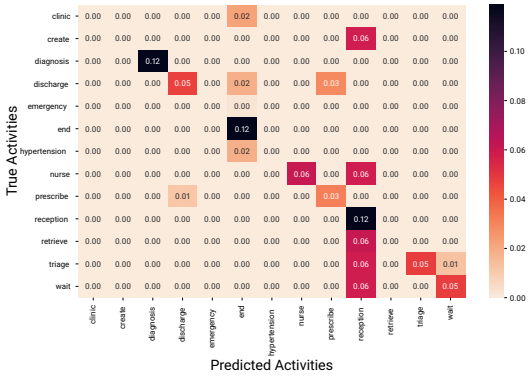
(b) Baseline ID Laboratory



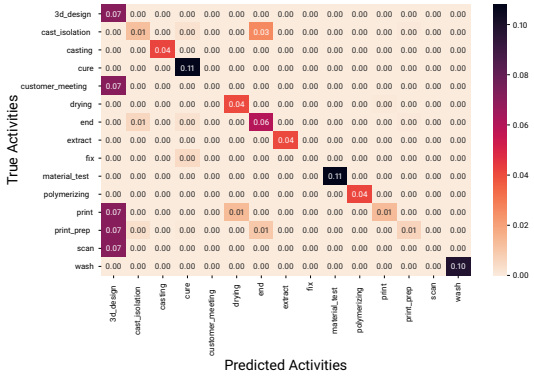
(c) Baseline Attribute Hospital



(d) Baseline Attribute Laboratory

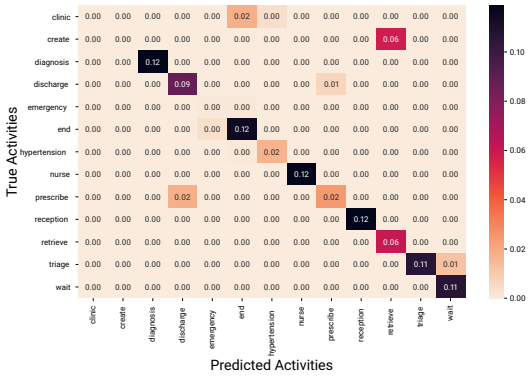


(e) Baseline Sensor Hospital

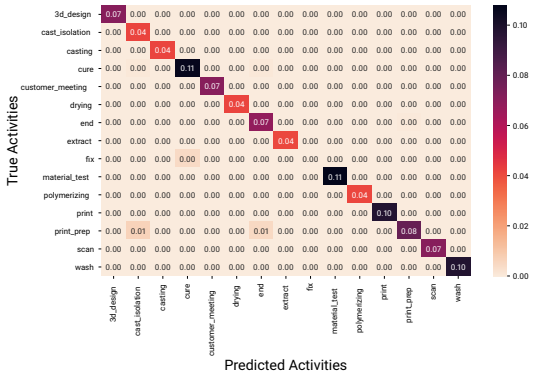


(f) Baseline Sensor Laboratory

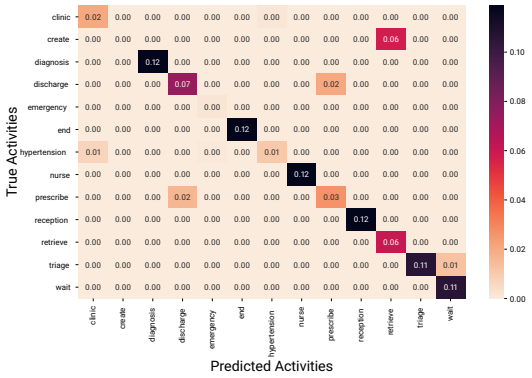
Figure A.3.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Flow Anomalies



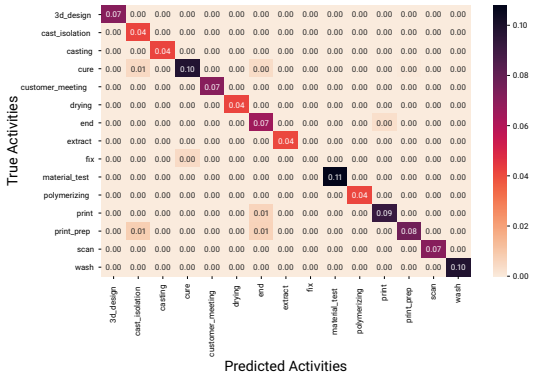
(g) Convolution Hospital



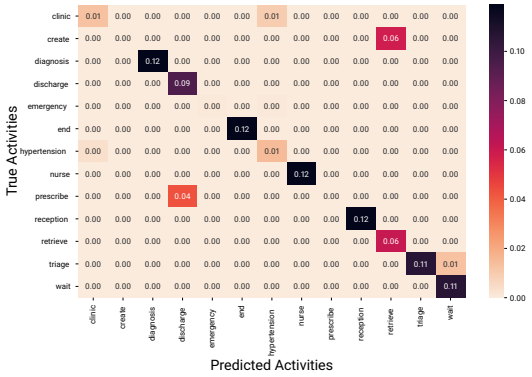
(h) Convolution Laboratory



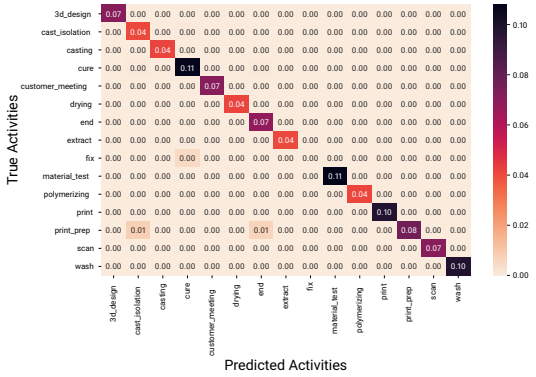
(i) Kalman Hospital



(j) Kalman Laboratory

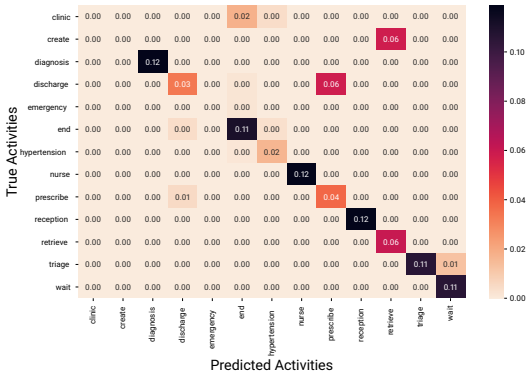


(k) Default Hospital

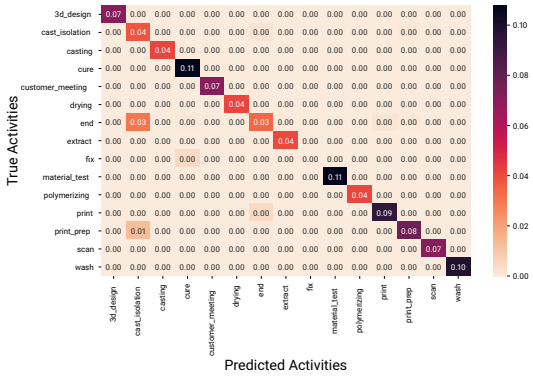


(l) Default Laboratory

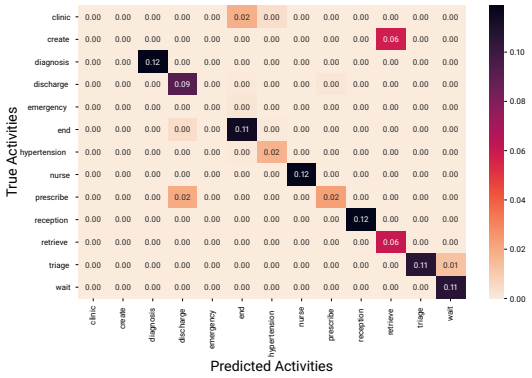
Figure A.3.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances and with 30% Flow Anomalies (cont.).



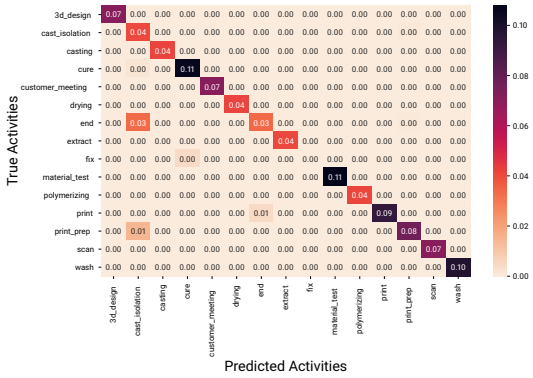
(m) ML CNN Hospital



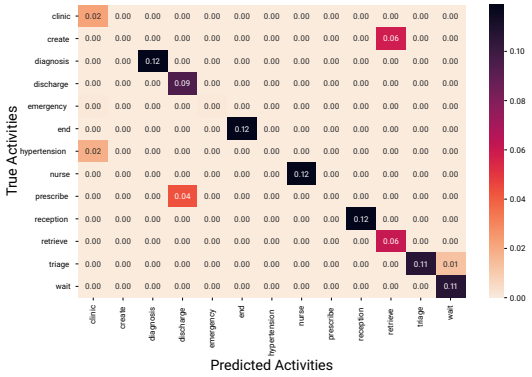
(n) ML CNN Laboratory



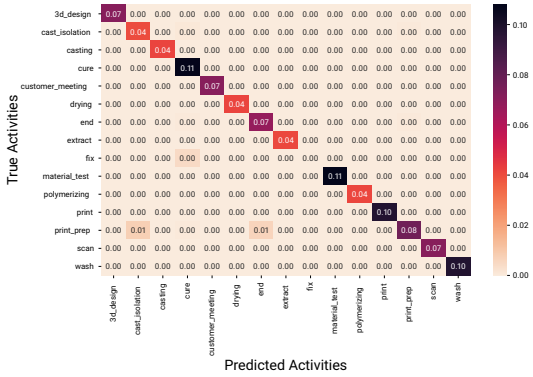
(o) ML CNN no Pooling Hospital



(p) ML CNN no Pooling Laboratory

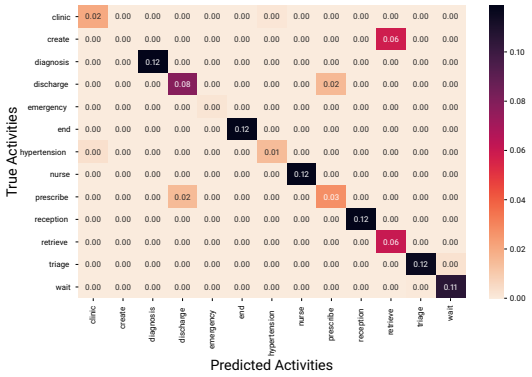


(q) ML FFN Hospital

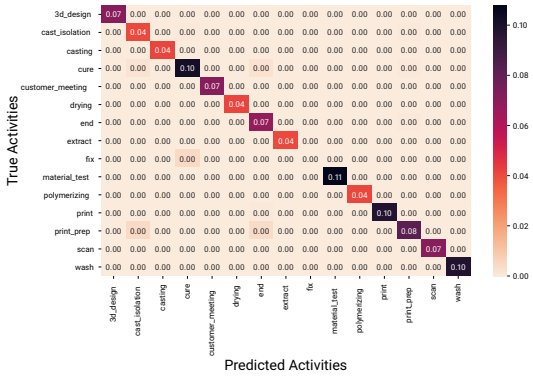


(r) ML FFN Laboratory

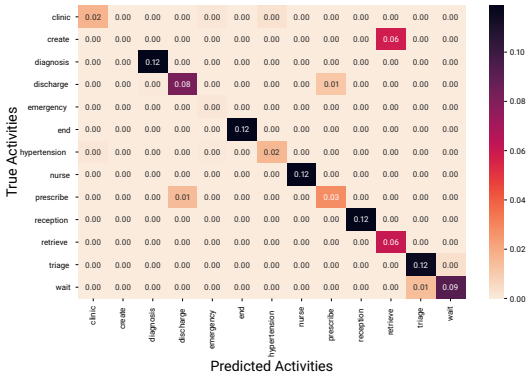
Figure A.3.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Flow Anomalies (cont.).



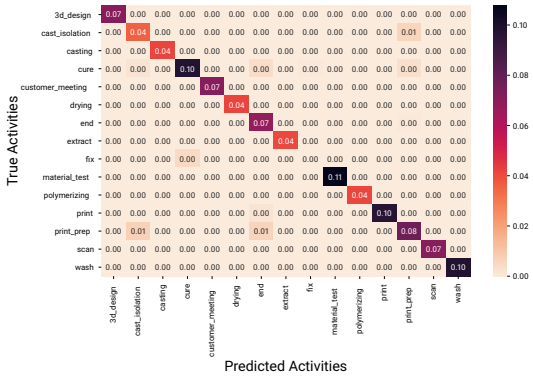
(s) RNN Hospital



(t) RNN Laboratory



(u) Normalize Hospital



(v) Normalize Laboratory

Figure A.3.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Flow Anomalies (cont.).

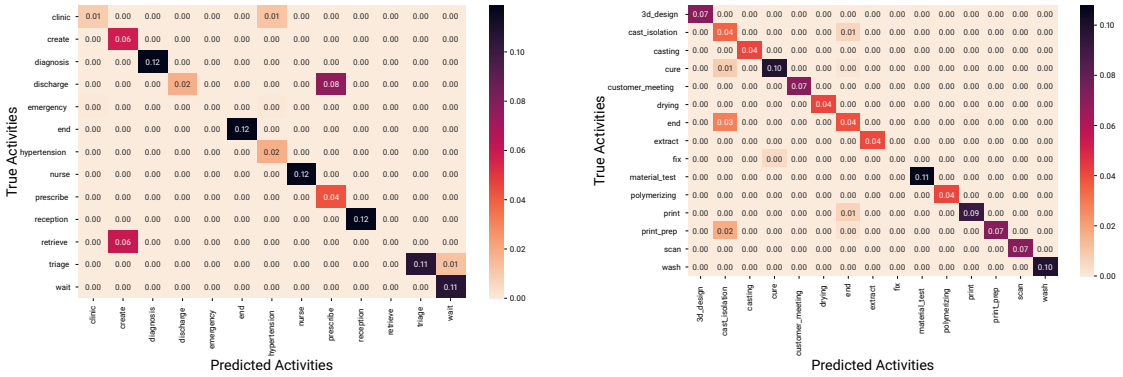
A.3.4. Experiment Results on the Train Datasets with 6000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies

Table A.7.: Performance of the model with different model configurations evaluated on L6000 and H6000 with 30% sensor anomalies and 30% flow anomalies.

	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000	rnn	0.9703	0.9678	0.9682	0.9703	0.9683
	normalize	0.9375	0.9319	0.9376	0.9375	0.9240
	ml_ffn	0.9682	0.9655	0.9661	0.9682	0.9661
	ml_cnn_no_pool	0.9327	0.9269	0.9390	0.9327	0.9328
	ml_cnn	0.9340	0.9283	0.9416	0.9340	0.9348
	kalman	0.9613	0.9580	0.9642	0.9613	0.9600
	default	0.9720	0.9696	0.9698	0.9720	0.9698
	convolution	0.9757	0.9736	0.9734	0.9757	0.9733
	baseline_sensor	0.6410	0.6072	0.6288	0.6410	0.5807
	baseline_id	0.9212	0.9145	0.9414	0.9212	0.9261
	baseline_attr	0.7488	0.7278	0.7621	0.7488	0.7244
H6000	rnn	0.8967	0.8853	0.8777	0.8967	0.8782
	normalize	0.8752	0.8612	0.8484	0.8752	0.8539
	ml_ffn	0.8753	0.8615	0.8580	0.8753	0.8563
	ml_cnn_no_pool	0.8409	0.8228	0.8023	0.8409	0.8119
	ml_cnn	0.8450	0.8271	0.8014	0.8450	0.8138
	kalman	0.8837	0.8706	0.8536	0.8837	0.8595
	default	0.8740	0.8601	0.8527	0.8740	0.8556
	convolution	0.8617	0.8457	0.8155	0.8617	0.8307
	baseline_sensor	0.5958	0.5448	0.6473	0.5958	0.5669
	baseline_id	0.8364	0.8189	0.8585	0.8364	0.8071
	baseline_attr	0.7712	0.7427	0.6866	0.7712	0.7156

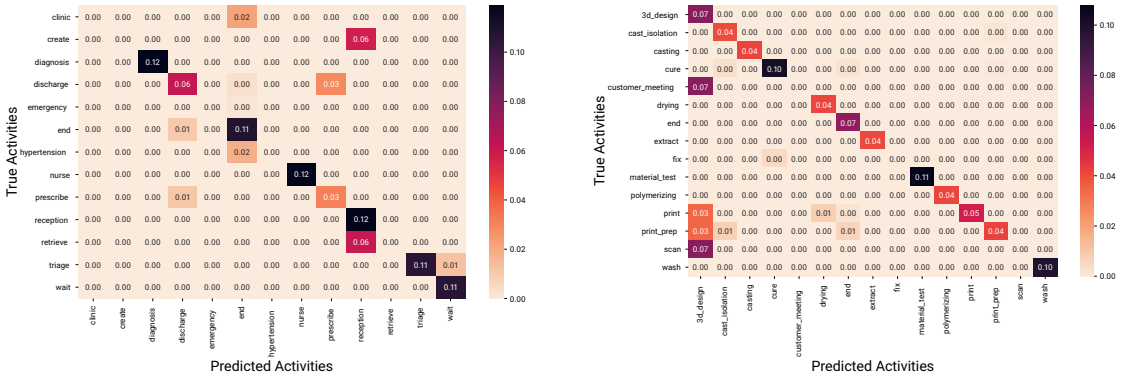
Table A.8.: Performance of the model with different model configurations evaluated on **the decisions** of L6000 and H6000 with 30% sensor anomalies and 30% flow anomalies.

	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Decisions	rnn	0.8813	0.8313	0.8698	0.8813	0.8683
	normalize	0.7535	0.6540	0.8130	0.7535	0.7278
	ml_ffn	0.8731	0.8188	0.8491	0.8731	0.8524
	ml_cnn_no_pool	0.7285	0.6131	0.7217	0.7285	0.7105
	ml_cnn	0.7335	0.6207	0.7499	0.7335	0.7211
	kalman	0.8434	0.7751	0.8254	0.8434	0.8134
	default	0.8874	0.8381	0.8664	0.8874	0.8648
	convolution	0.9022	0.8590	0.8883	0.9022	0.8782
	baseline_sensor	0.7579	0.6437	0.7591	0.7579	0.7241
	baseline_id	0.6807	0.5481	0.6846	0.6807	0.6548
	baseline_attr	0.8766	0.8231	0.8566	0.8766	0.8561
H6000 Decisions	rnn	0.7422	0.6912	0.6928	0.7422	0.6936
	normalize	0.6942	0.6363	0.6647	0.6942	0.6611
	ml_ffn	0.6882	0.6229	0.6116	0.6882	0.6243
	ml_cnn_no_pool	0.6433	0.5718	0.6002	0.6433	0.5884
	ml_cnn	0.6396	0.5654	0.5668	0.6396	0.5835
	kalman	0.7091	0.6453	0.6048	0.7091	0.6346
	default	0.6853	0.6192	0.5987	0.6853	0.6227
	convolution	0.6546	0.5828	0.6014	0.6546	0.5973
	baseline_sensor	0.3150	0.2498	0.4083	0.3150	0.3490
	baseline_id	0.5907	0.5129	0.4244	0.5907	0.4715
	baseline_attr	0.4623	0.3868	0.4466	0.4623	0.4494



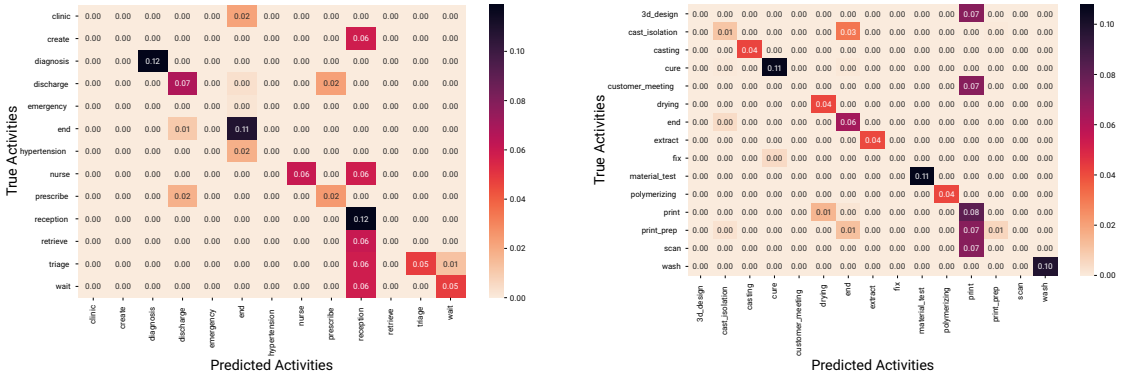
(a) Baseline ID Hospital

(b) Baseline ID Laboratory



(c) Baseline Attribute Hospital

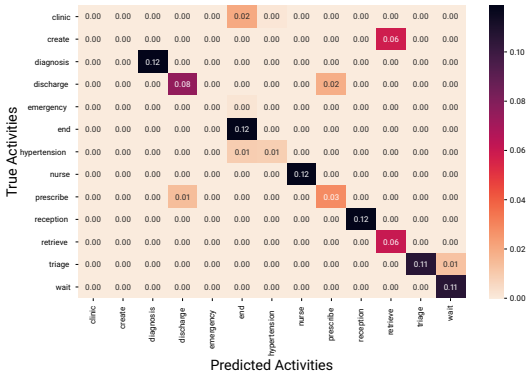
(d) Baseline Attribute Laboratory



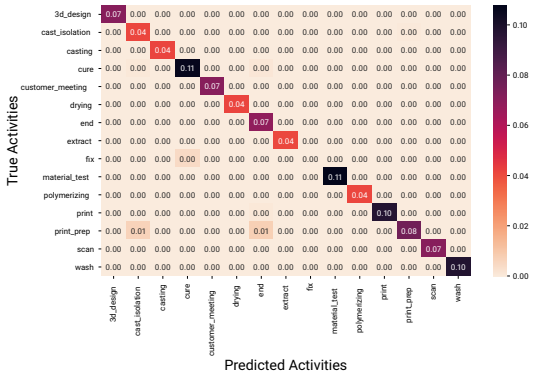
(e) Baseline Sensor Hospital

(f) Baseline Sensor Laboratory

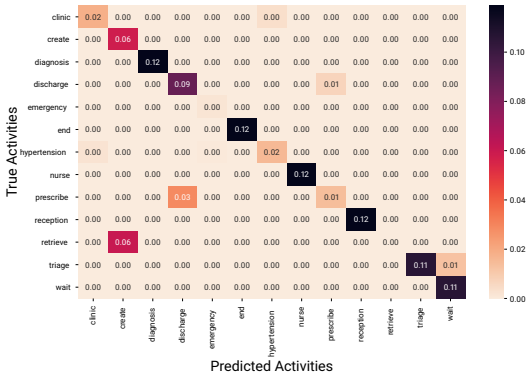
Figure A.4.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies.



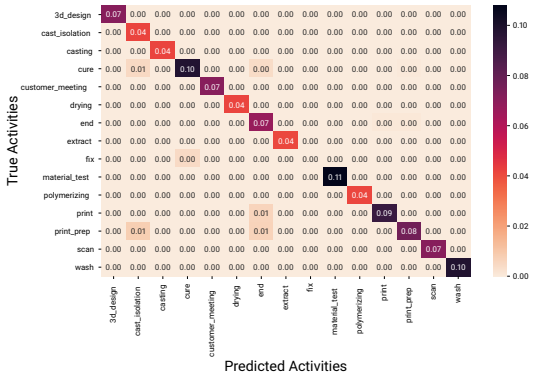
(g) Convolution Hospital



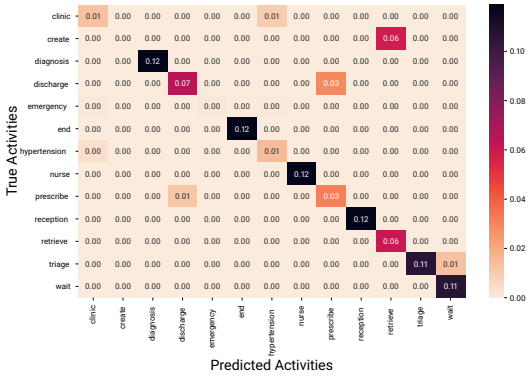
(h) Convolution Laboratory



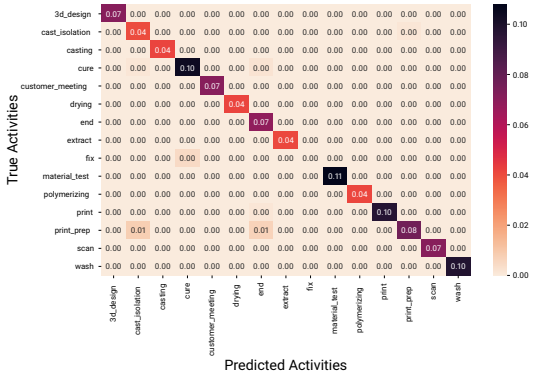
(i) Kalman Hospital



(j) Kalman Laboratory

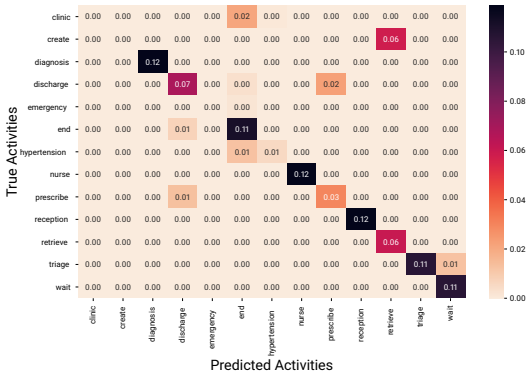


(k) Default Hospital

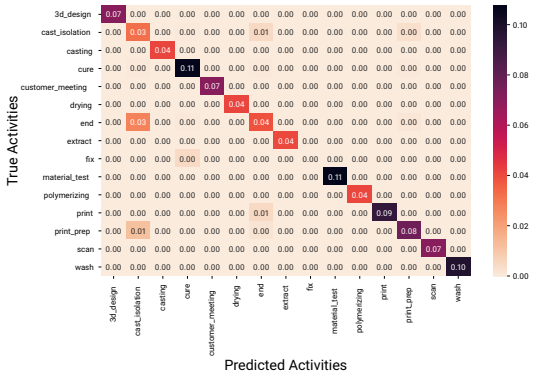


(l) Default Laboratory

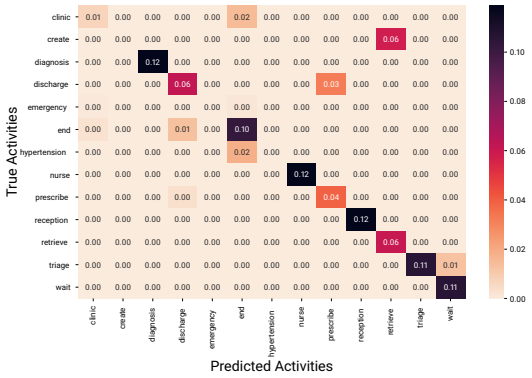
Figure A.4.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies (cont.).



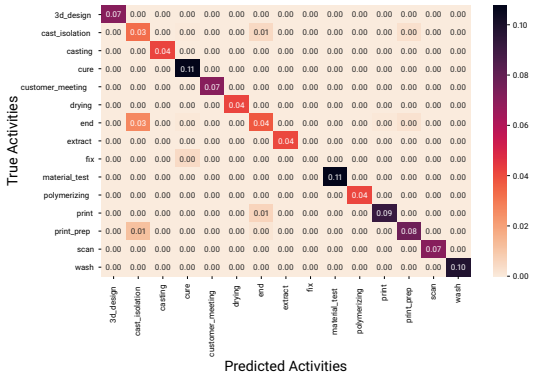
(m) ML CNN Hospital



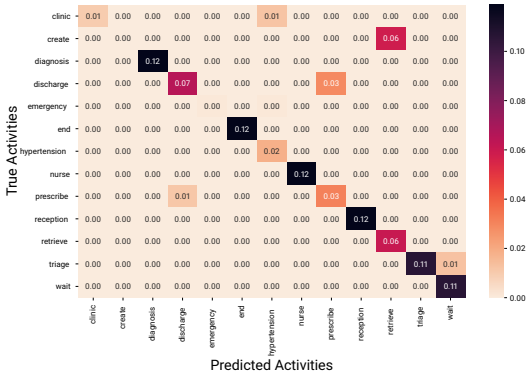
(n) ML CNN Laboratory



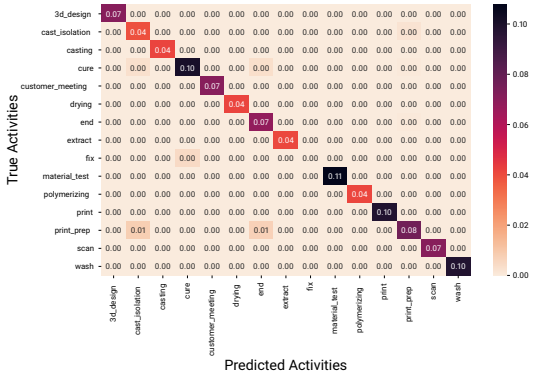
(o) ML CNN no Pooling Hospital



(p) ML CNN no Pooling Laboratory

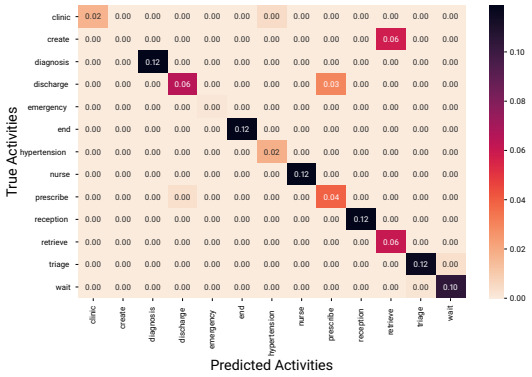


(q) ML FFN Hospital

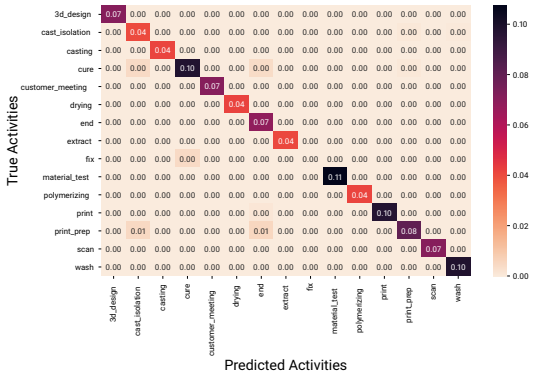


(r) ML FFN Laboratory

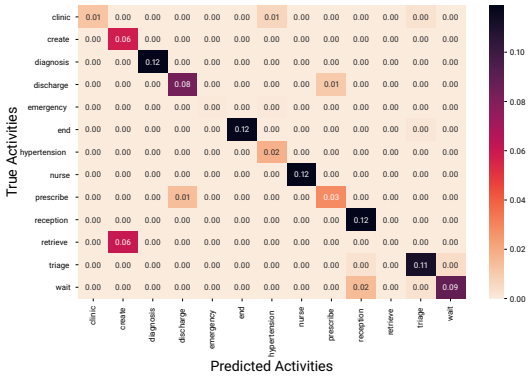
Figure A.4.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies (cont.).



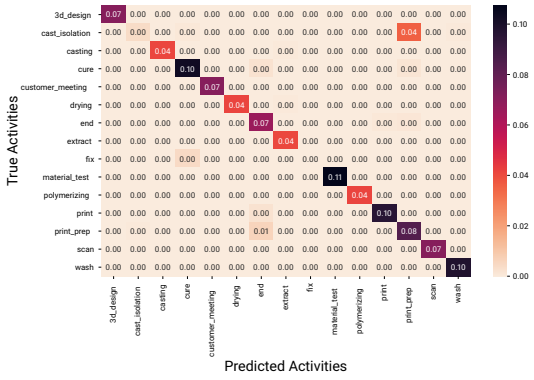
(s) RNN Hospital



(t) RNN Laboratory



(u) Normalize Hospital



(v) Normalize Laboratory

Figure A.4.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies (cont.).

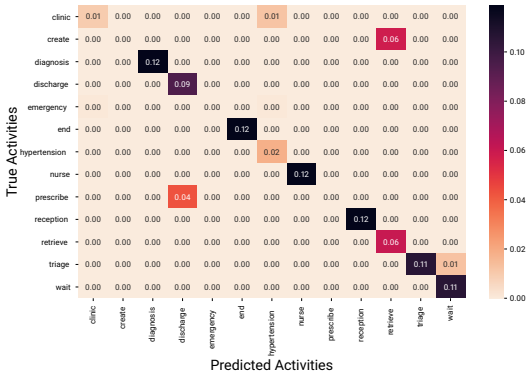
A.3.5. Experiment Results on the Train Datasets with 10000 Instances and no Anomalies

Table A.9.: Performance of the model with different model configurations evaluated on L10000 and H10000 without anomalies.

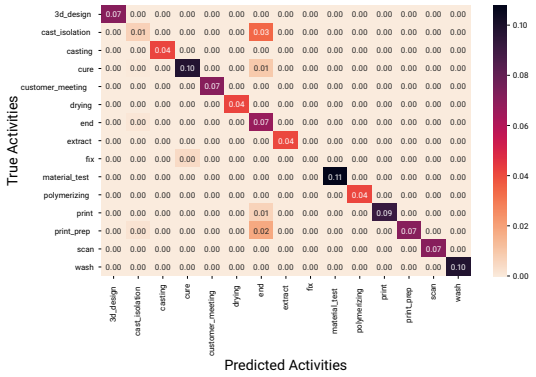
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000	rnn	0.9787	0.9769	0.9747	0.9787	0.9764
	normalize	0.9845	0.9831	0.9811	0.9845	0.9824
	ml_ffn	0.9756	0.9735	0.9724	0.9756	0.9731
	ml_cnn_no_pool	0.9750	0.9728	0.9728	0.9750	0.9728
	ml_cnn	0.9384	0.9330	0.9418	0.9384	0.9337
	kalman	0.9629	0.9597	0.9645	0.9629	0.9614
	default	0.9774	0.9754	0.9751	0.9774	0.9750
	convolution	0.9794	0.9777	0.9763	0.9794	0.9771
	baseline_sensor	0.6438	0.6102	0.6491	0.6438	0.5801
	baseline_id	0.9212	0.9143	0.9422	0.9212	0.9166
	baseline_attr	0.7573	0.7369	0.7725	0.7573	0.7323
H10000	rnn	0.9089	0.8987	0.8802	0.9089	0.8870
	normalize	0.9153	0.9059	0.8861	0.9153	0.8957
	ml_ffn	0.8755	0.8617	0.8584	0.8755	0.8565
	ml_cnn_no_pool	0.8310	0.8119	0.8010	0.8310	0.7993
	ml_cnn	0.8723	0.8575	0.8287	0.8723	0.8391
	kalman	0.8872	0.8746	0.8581	0.8872	0.8670
	default	0.8706	0.8559	0.8612	0.8706	0.8326
	convolution	0.9055	0.8949	0.8768	0.9055	0.8849
	baseline_sensor	0.5972	0.5467	0.6507	0.5972	0.5691
	baseline_id	0.8736	0.8595	0.8486	0.8736	0.8529
	baseline_attr	0.7714	0.7428	0.7022	0.7714	0.7158

Table A.10.: Performance of the model with different model configurations evaluated on **the decisions** of L10000 and H10000 without anomalies.

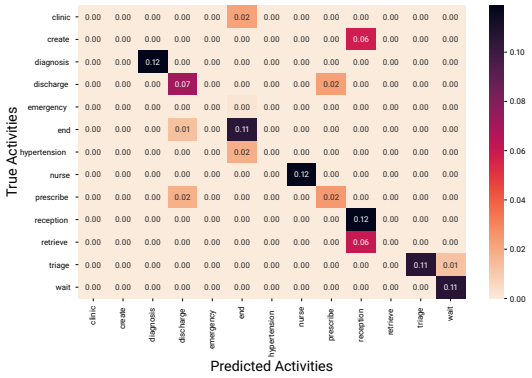
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000 Decisions	rnn	0.9139	0.8769	0.8927	0.9139	0.9010
	normalize	0.9374	0.9109	0.9238	0.9374	0.9285
	ml_ffn	0.9012	0.8570	0.8777	0.9012	0.8777
	ml_cnn_no_pool	0.8987	0.8532	0.8866	0.8987	0.8744
	ml_cnn	0.7505	0.6292	0.7642	0.7505	0.7089
	kalman	0.8496	0.7834	0.8509	0.8496	0.8163
	default	0.9084	0.8674	0.8986	0.9084	0.8831
	convolution	0.9167	0.8804	0.9009	0.9167	0.8979
	baseline_sensor	0.7653	0.6517	0.7955	0.7653	0.7186
	baseline_id	0.6806	0.5227	0.6548	0.6806	0.6122
	baseline_attr	0.9078	0.8667	0.8967	0.9078	0.8833
H10000 Decisions	rnn	0.7721	0.7248	0.7017	0.7721	0.7177
	normalize	0.7882	0.7441	0.7140	0.7882	0.7375
	ml_ffn	0.6885	0.6232	0.6436	0.6885	0.6248
	ml_cnn_no_pool	0.6698	0.5974	0.5657	0.6698	0.5782
	ml_cnn	0.6951	0.6295	0.6005	0.6951	0.6217
	kalman	0.7178	0.6567	0.6139	0.7178	0.6525
	default	0.6763	0.6033	0.6264	0.6763	0.5681
	convolution	0.7678	0.7184	0.7157	0.7678	0.7109
	baseline_sensor	0.3167	0.2537	0.4197	0.3167	0.3515
	baseline_id	0.6838	0.6162	0.5891	0.6838	0.6168
	baseline_attr	0.4610	0.3835	0.5131	0.4610	0.4475



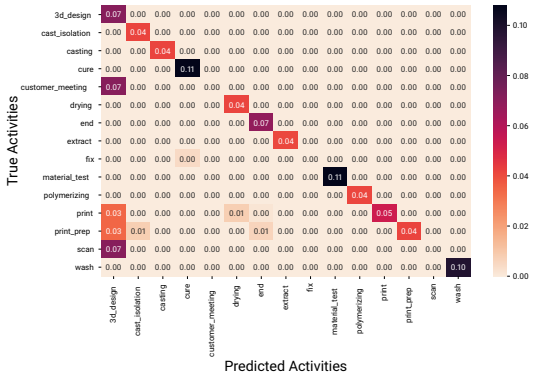
(a) Baseline ID Hospital



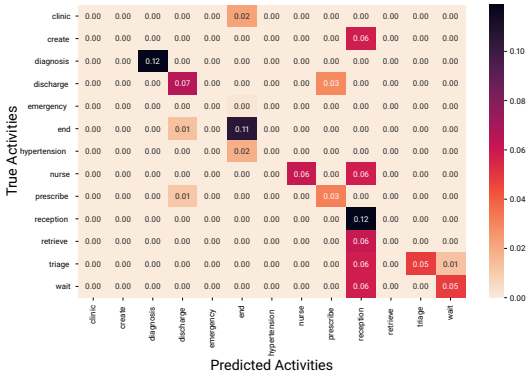
(b) Baseline ID Laboratory



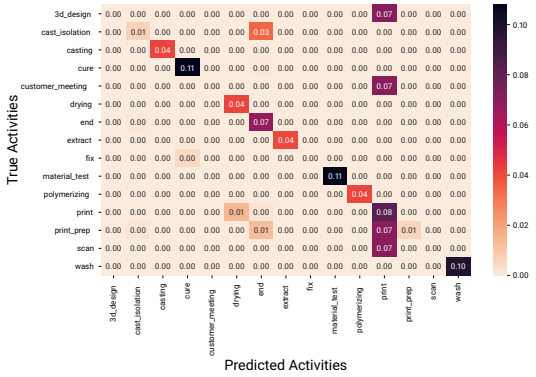
(c) Baseline Attribute Hospital



(d) Baseline Attribute Laboratory

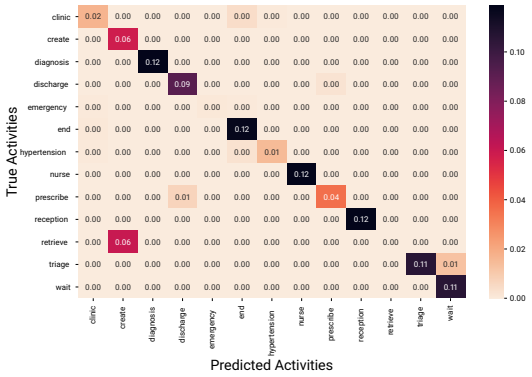


(e) Baseline Sensor Hospital

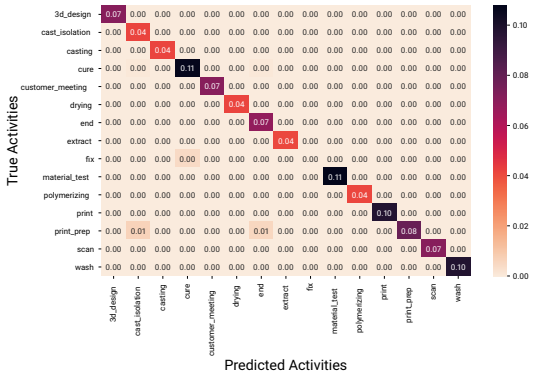


(f) Baseline Sensor Laboratory

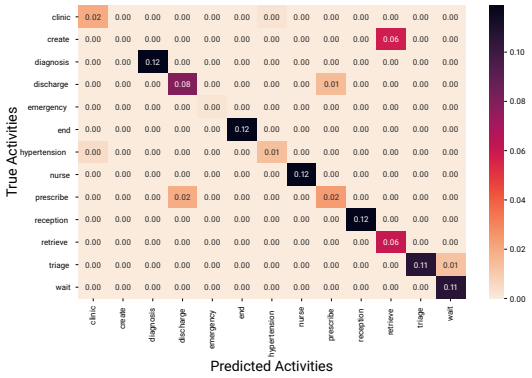
Figure A.5.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances and without Anomalies.



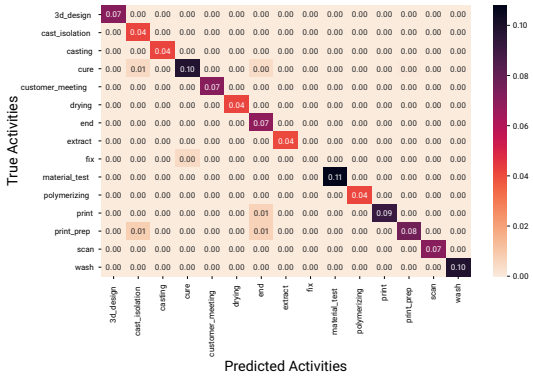
(g) Convolution Hospital



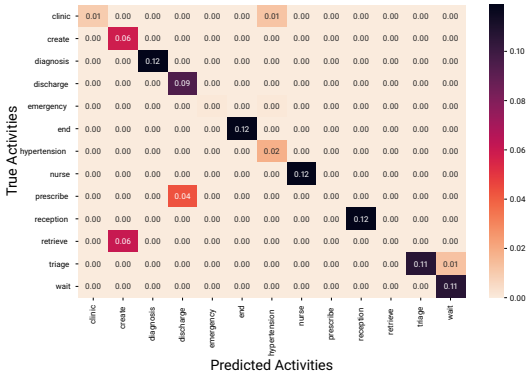
(h) Convolution Laboratory



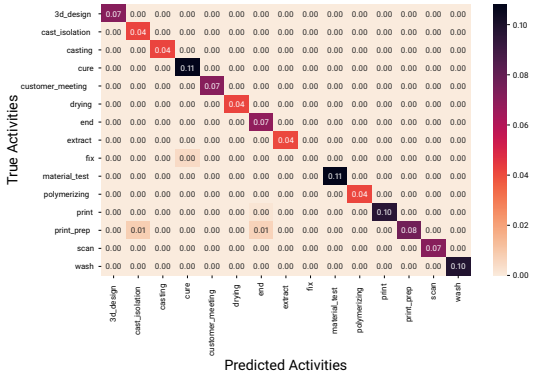
(i) Kalman Hospital



(j) Kalman Laboratory

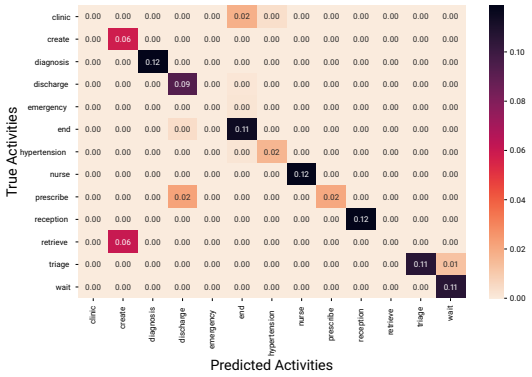


(k) Default Hospital

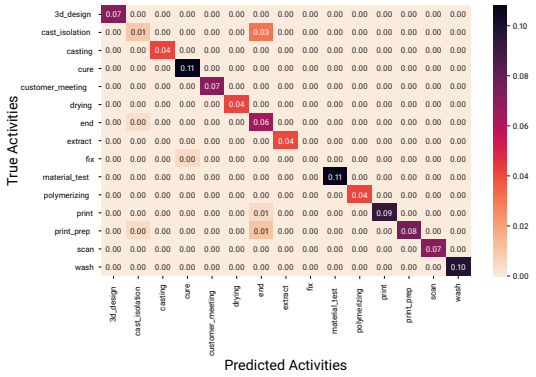


(l) Default Laboratory

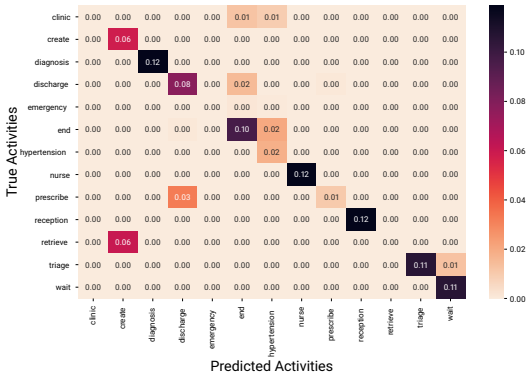
Figure A.5.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances and without Anomalies (cont.).



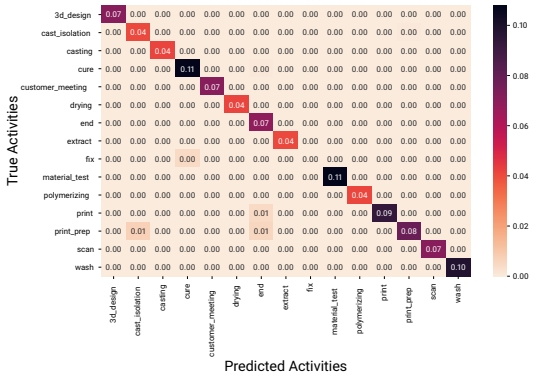
(m) ML CNN Hospital



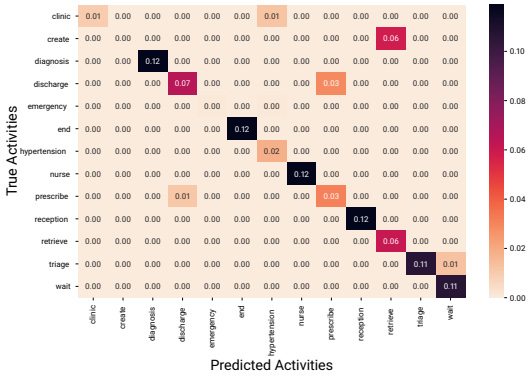
(n) ML CNN Laboratory



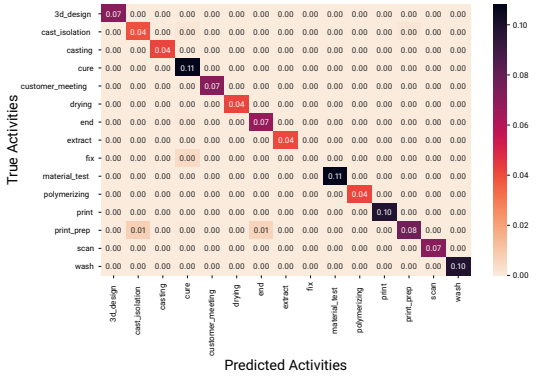
(o) ML CNN no Pooling Hospital



(p) ML CNN no Pooling Laboratory

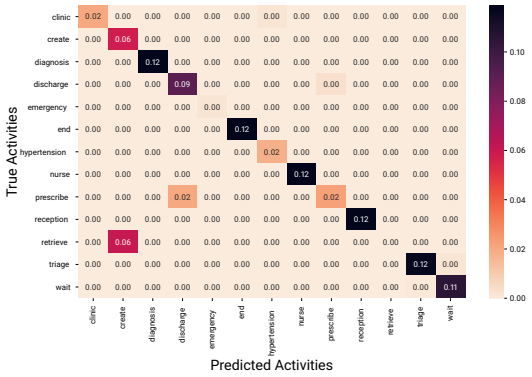


(q) ML FFN Hospital

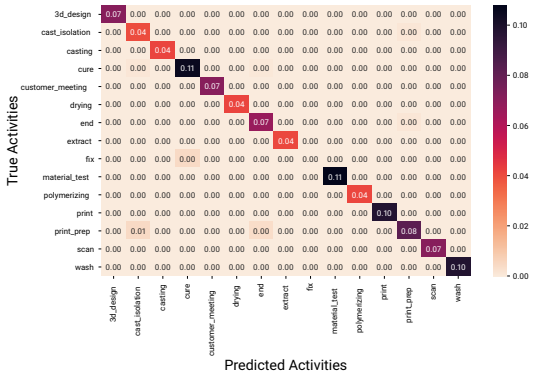


(r) ML FFN Laboratory

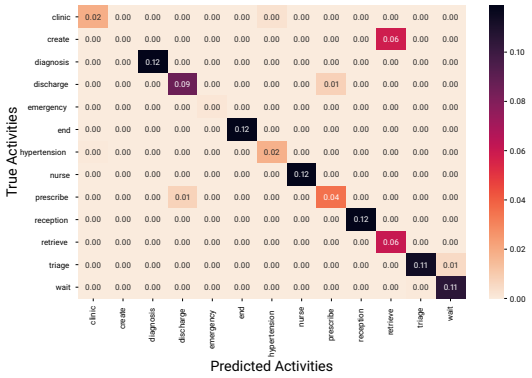
Figure A.5.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances and without Anomalies (cont.).



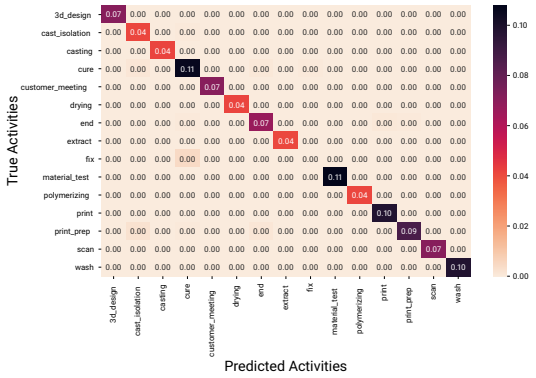
(s) RNN Hospital



(t) RNN Laboratory



(u) Normalize Hospital



(v) Normalize Laboratory

Figure A.5.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances and without Anomalies (cont.).

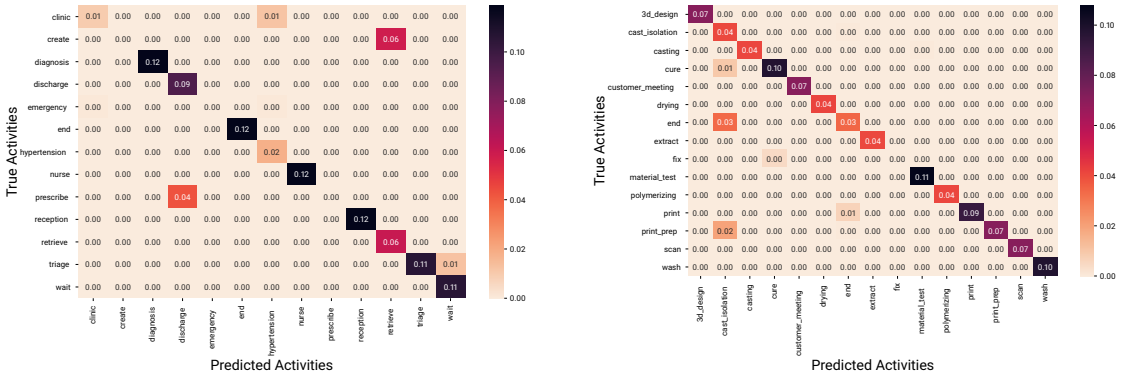
A.3.6. Experiment Results on the Train Datasets with 10000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies

Table A.11.: Performance of the model with different model configurations evaluated on L10000 and H10000 with 30% sensor anomalies and 30% flow anomalies.

	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000	rnn	0.9657	0.9628	0.9665	0.9657	0.9641
	normalize	0.9091	0.9017	0.9432	0.9091	0.9140
	ml_ffn	0.9762	0.9742	0.9738	0.9762	0.9739
	ml_cnn_no_pool	0.9372	0.9319	0.9543	0.9372	0.9385
	ml_cnn	0.9396	0.9345	0.9533	0.9396	0.9407
	kalman	0.9627	0.9595	0.9643	0.9627	0.9612
	default	0.9759	0.9738	0.9739	0.9759	0.9735
	convolution	0.9789	0.9771	0.9761	0.9789	0.9766
	baseline_sensor	0.6417	0.6088	0.6577	0.6417	0.5857
	baseline_id	0.9221	0.9156	0.9500	0.9221	0.9270
	baseline_attr	0.7545	0.7339	0.7696	0.7545	0.7296
H10000	0.9058	0.8952	0.8783	0.9058	0.8835	
	normalize	0.9049	0.8943	0.8758	0.9049	0.8850
	ml_ffn	0.8713	0.8565	0.8577	0.8713	0.8339
	ml_cnn_no_pool	0.8452	0.8271	0.8119	0.8452	0.8052
	ml_cnn	0.8634	0.8476	0.8182	0.8634	0.8321
	kalman	0.8853	0.8724	0.8561	0.8853	0.8633
	default	0.8716	0.8569	0.8151	0.8716	0.8344
	convolution	0.8641	0.8484	0.8172	0.8641	0.8324
	baseline_sensor	0.5946	0.5422	0.6188	0.5946	0.5486
	baseline_id	0.8717	0.8570	0.8158	0.8717	0.8329
	baseline_attr	0.7709	0.7423	0.6836	0.7709	0.7153

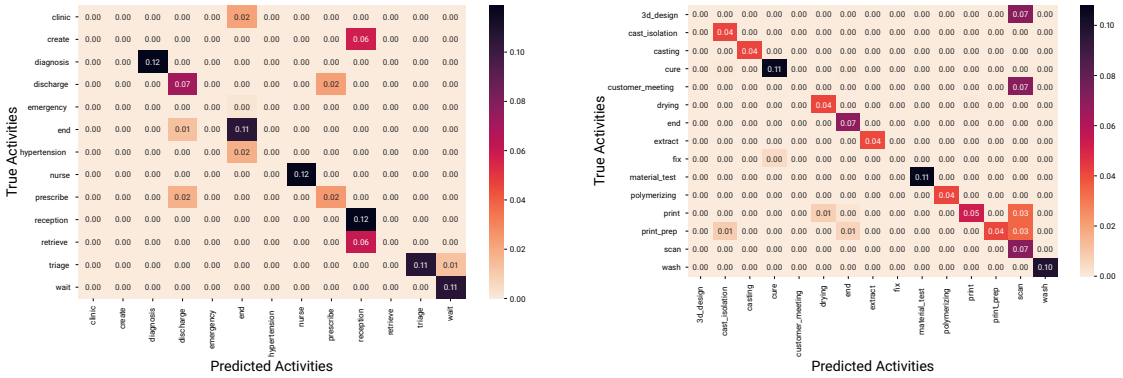
Table A.12.: Performance of the model with different model configurations evaluated on **the decisions** of L10000 and H10000 with 30% sensor anomalies and 30% flow anomalies.

	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000 Decisions	rnn	0.8632	0.8041	0.8643	0.8632	0.8396
	normalize	0.8038	0.7268	0.8532	0.8038	0.7859
	ml_ffn	0.9039	0.8615	0.8882	0.9039	0.8801
	ml_cnn_no_pool	0.7456	0.6415	0.8068	0.7456	0.7339
	ml_cnn	0.7554	0.6526	0.8089	0.7554	0.7413
	kalman	0.8488	0.7824	0.8285	0.8488	0.8159
	default	0.9022	0.8587	0.8925	0.9022	0.8769
	convolution	0.9147	0.8773	0.9006	0.9147	0.8946
	baseline_sensor	0.7602	0.6612	0.8367	0.7602	0.7457
	baseline_id	0.6865	0.5595	0.7343	0.6865	0.6582
	baseline_attr	0.8992	0.8545	0.8868	0.8992	0.8744
H10000 Decisions	rnn	0.7650	0.7153	0.6973	0.7650	0.7095
	normalize	0.7650	0.7158	0.6932	0.7650	0.7143
	ml_ffn	0.6780	0.6044	0.6177	0.6780	0.5713
	ml_cnn_no_pool	0.6455	0.5681	0.5815	0.6455	0.5596
	ml_cnn	0.6911	0.6246	0.5954	0.6911	0.6251
	kalman	0.7129	0.6507	0.6100	0.7129	0.6437
	default	0.6789	0.6055	0.5110	0.6789	0.5726
	convolution	0.6640	0.5939	0.5809	0.6640	0.6037
	baseline_sensor	0.3101	0.2361	0.3320	0.3101	0.3011
	baseline_id	0.6790	0.6057	0.5128	0.6790	0.5690
	baseline_attr	0.4604	0.3828	0.4352	0.4604	0.4467



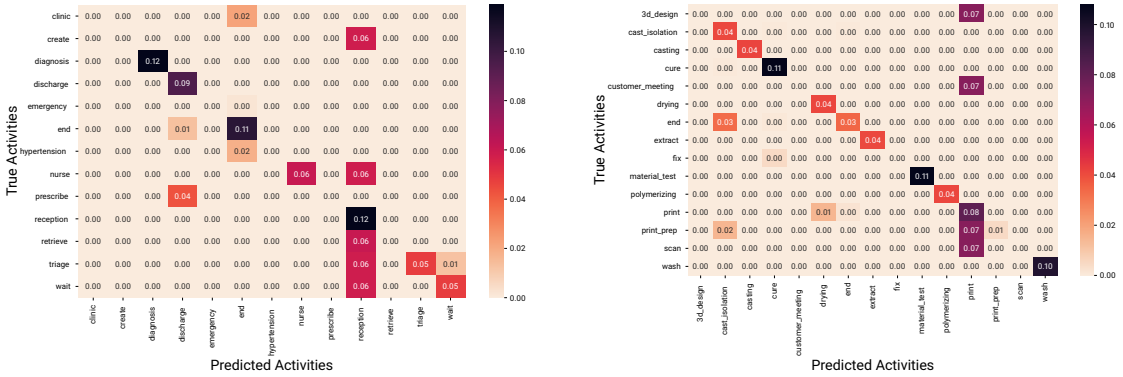
(a) Baseline ID Hospital

(b) Baseline ID Laboratory



(c) Baseline Attribute Hospital

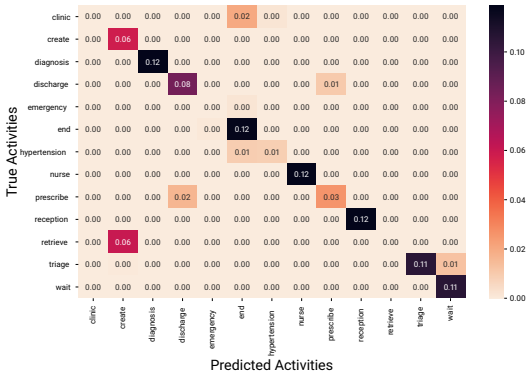
(d) Baseline Attribute Laboratory



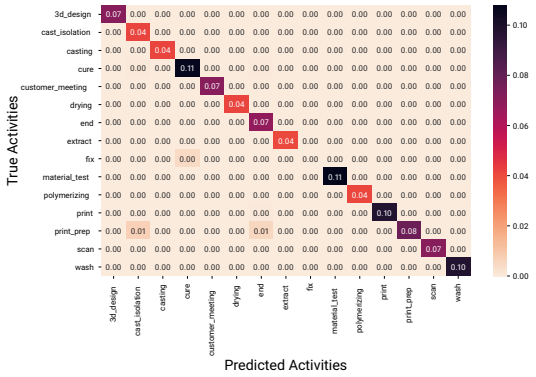
(e) Baseline Sensor Hospital

(f) Baseline Sensor Laboratory

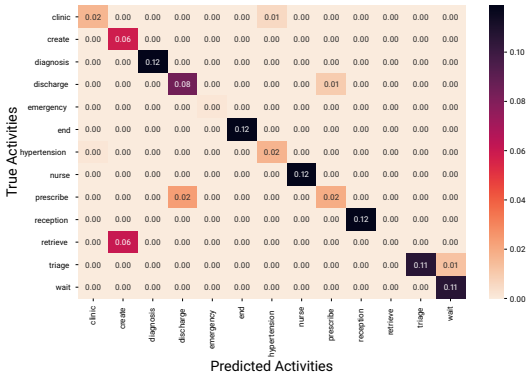
Figure A.6.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies.



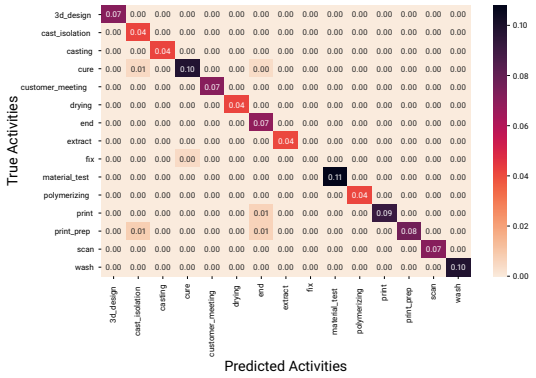
(g) Convolution Hospital



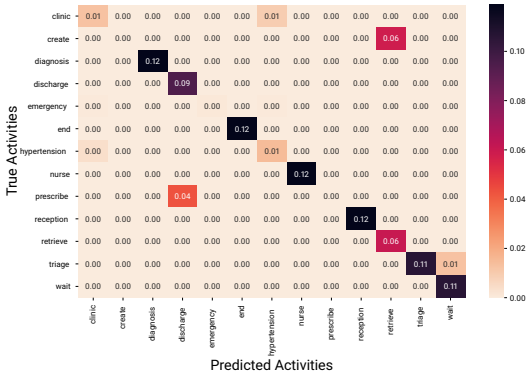
(h) Convolution Laboratory



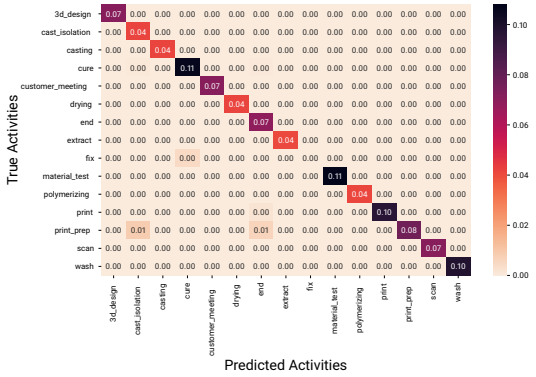
(i) Kalman Hospital



(j) Kalman Laboratory

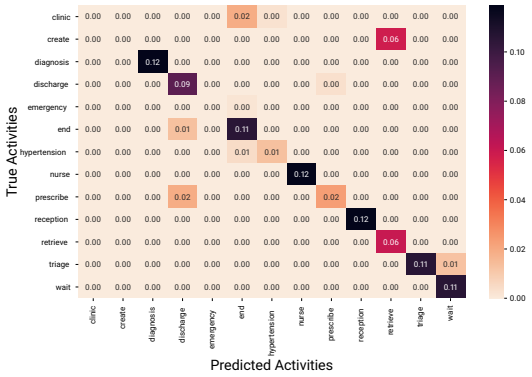


(k) Default Hospital

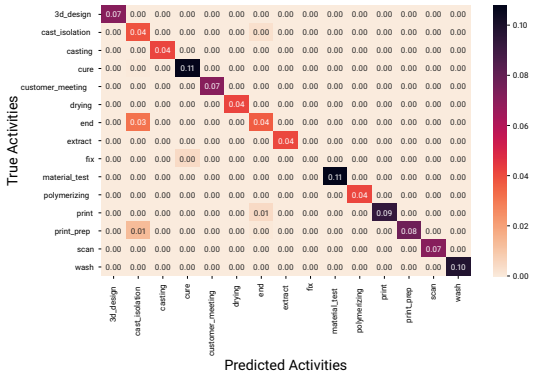


(l) Default Laboratory

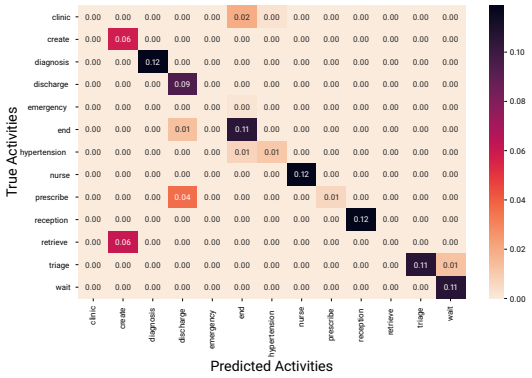
Figure A.6.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies (cont.).



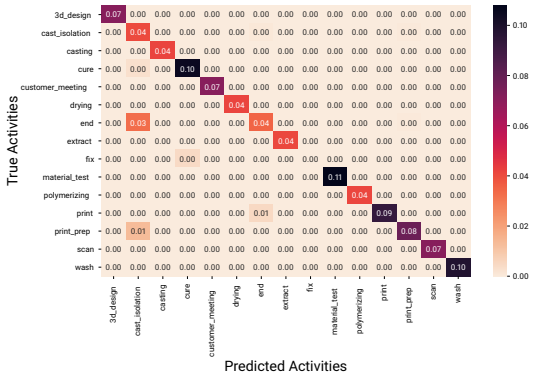
(m) ML CNN Hospital



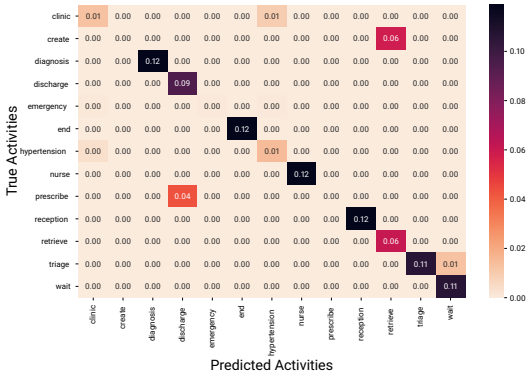
(n) ML CNN Laboratory



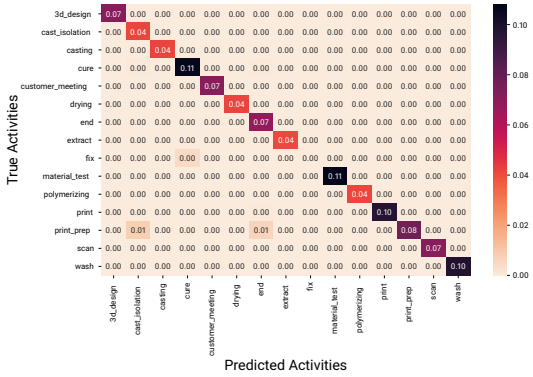
(o) ML CNN no Pooling Hospital



(p) ML CNN no Pooling Laboratory

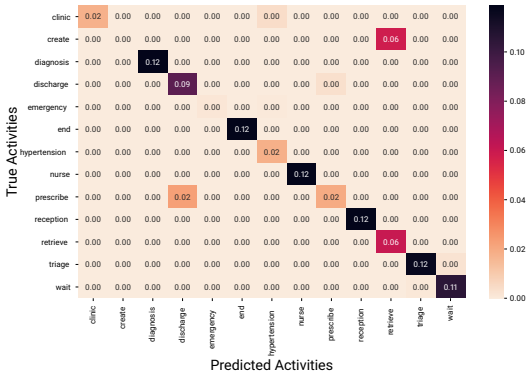


(q) ML FFN Hospital

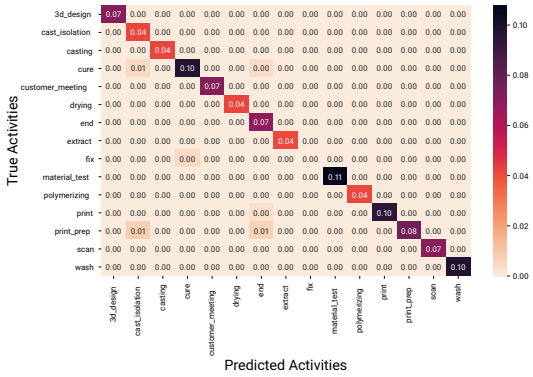


(r) ML FFN Laboratory

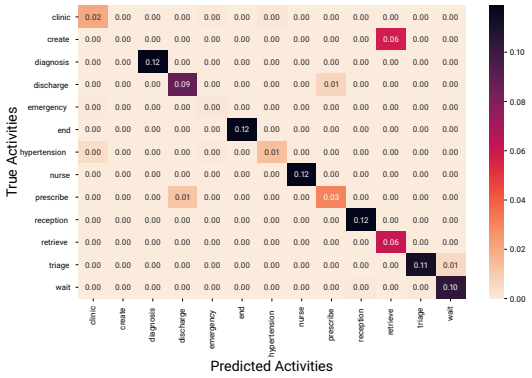
Figure A.6.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies (cont.).



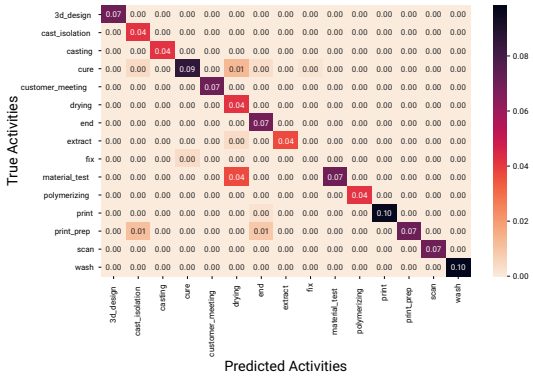
(s) RNN Hospital



(t) RNN Laboratory



(u) Normalize Hospital



(v) Normalize Laboratory

Figure A.6.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies (cont.).

A.4. Combined PreProcessing for Process Prediction

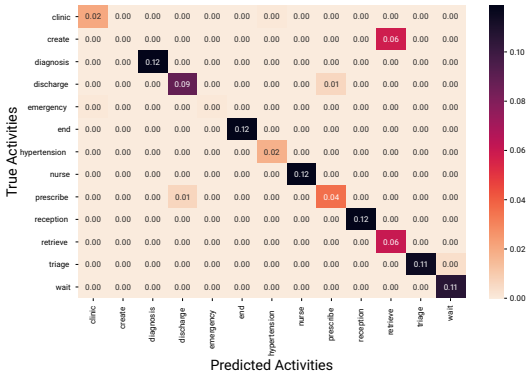
A.4.1. Experiment Results on the Train Datasets with 6000 Instances and no Anomalies

Table A.13.: Performance of the model with combined model configurations evaluated on L6000 and H6000 without anomalies.

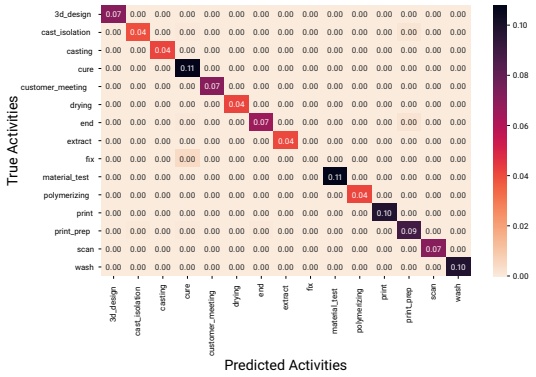
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000	norm_rnn	0.9862	0.9850	0.9826	0.9862	0.9839
	norm_cnn	0.9871	0.9860	0.9837	0.9871	0.9852
	kalman_rnn	0.9819	0.9804	0.9806	0.9819	0.9796
	kalman_cnn	0.9794	0.9776	0.9775	0.9794	0.9772
	kal_norm_rnn	0.9841	0.9827	0.9848	0.9841	0.9818
	kal_norm_cnn	0.9572	0.9534	0.9615	0.9572	0.9536
H6000	norm_rnn	0.9215	0.9127	0.8918	0.9215	0.9016
	norm_cnn	0.9159	0.9065	0.8866	0.9159	0.8961
	kalman_rnn	0.9046	0.8939	0.8748	0.9046	0.8832
	kalman_cnn	0.9023	0.8913	0.8756	0.9023	0.8782
	kal_norm_rnn	0.9203	0.9114	0.8908	0.9203	0.9003
	kal_norm_cnn	0.8264	0.7584	0.8923	0.8264	0.8329

Table A.14.: Performance of the model with different model configurations evaluated on **the decisions** of L6000 and H6000 without anomalies.

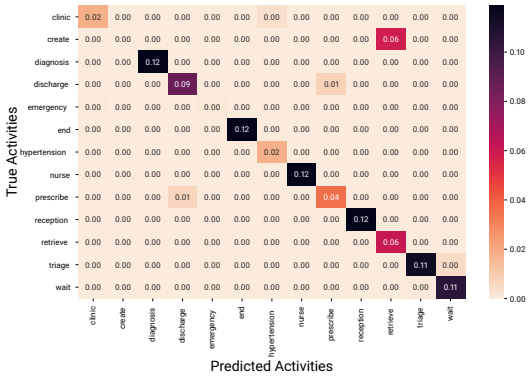
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Decisions	norm_rnn	0.9441	0.9203	0.9311	0.9441	0.9356
	norm_cnn	0.9481	0.9262	0.9338	0.9481	0.9396
	kalman_rnn	0.9268	0.8948	0.9197	0.9268	0.9128
	kalman_cnn	0.9166	0.8802	0.9024	0.9166	0.9004
	kal_norm_rnn	0.9356	0.9074	0.9377	0.9356	0.9239
	kal_norm_cnn	0.7357	0.6413	0.8913	0.7357	0.7531
H6000 Decisions	norm_rnn	0.8035	0.7626	0.7291	0.8035	0.7529
	norm_cnn	0.7895	0.7458	0.7153	0.7895	0.7387
	kalman_rnn	0.7613	0.7115	0.6878	0.7613	0.7075
	kalman_cnn	0.7556	0.7036	0.6907	0.7556	0.6952
	kal_norm_rnn	0.8005	0.7601	0.7264	0.8005	0.7505
	kal_norm_cnn	0.7898	0.7475	0.7164	0.7898	0.7402



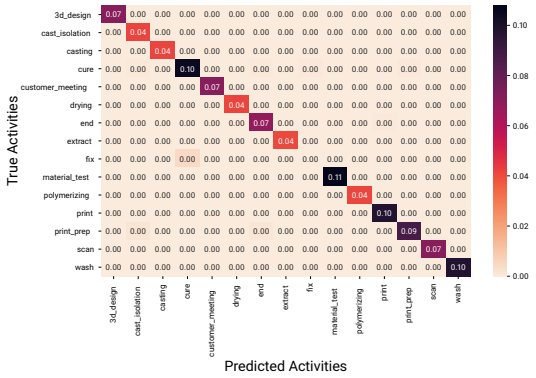
(a) Normalized RNN Hospital



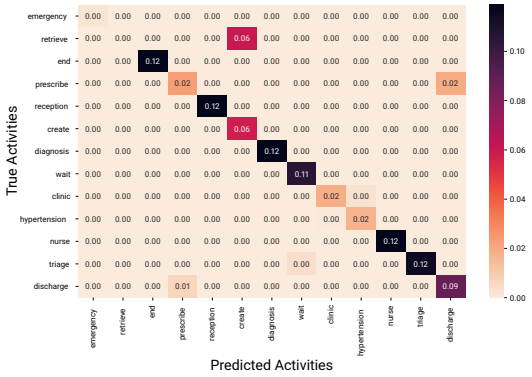
(b) Normalized RNN Laboratory



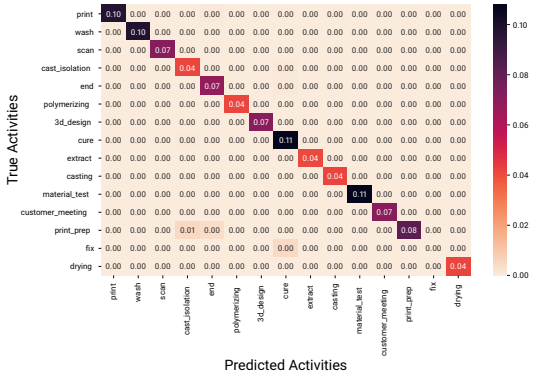
(c) Normalized CNN Hospital



(d) Normalized CNN Laboratory

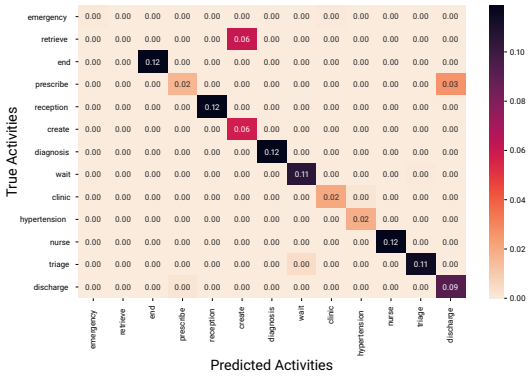


(e) Kalman RNN Hospital

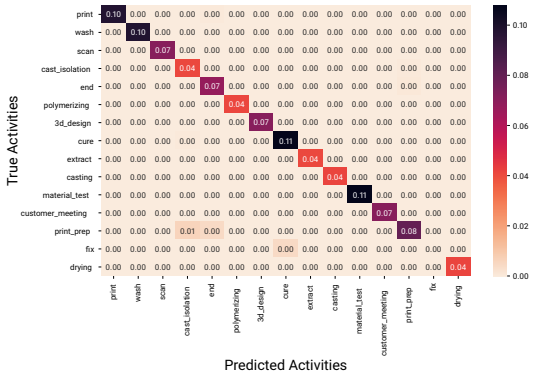


(f) Kalman RNN Laboratory

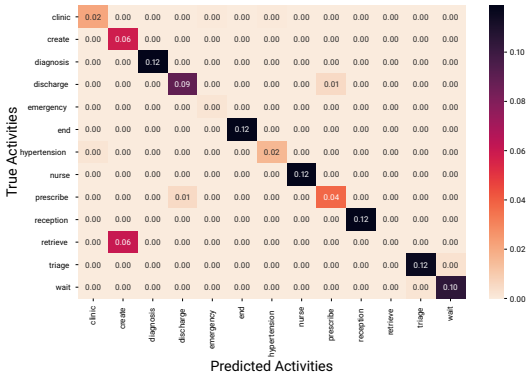
Figure A.7.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances and without Anomalies.



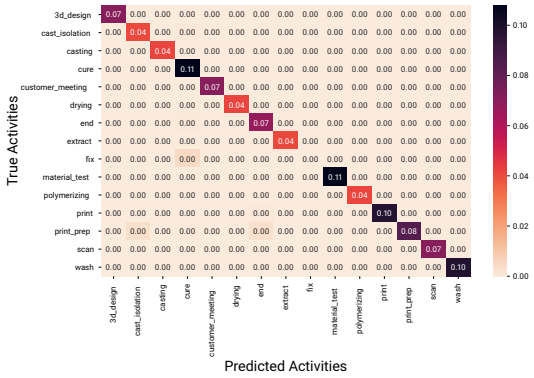
(g) Kalman CNN Hospital



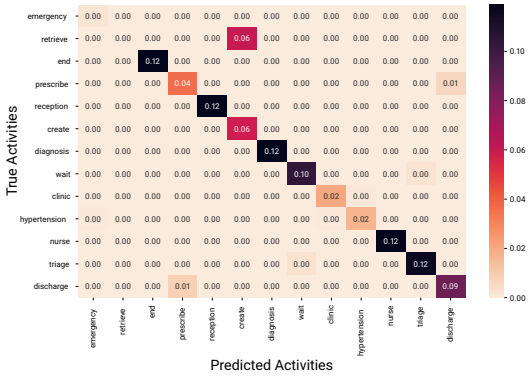
(h) Kalman CNN Laboratory



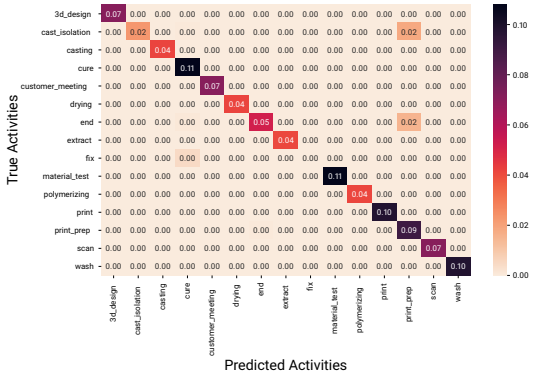
(i) Kalman Normalized RNN Hospital



(j) Kalman Normalized RNN Laboratory



(k) Kalman Normalized CNN Hospital



(l) Kalman Normalized CNN Laboratory

Figure A.7.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances and without Anomalies (cont.).

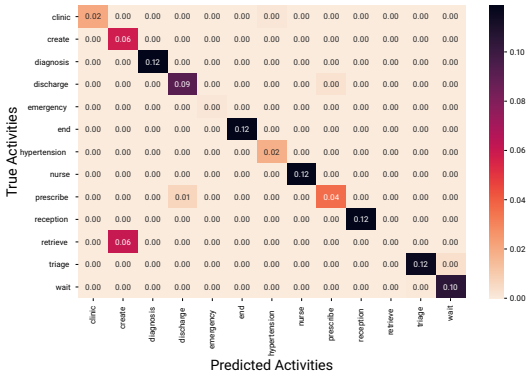
A.4.2. Experiment Results on the Train Datasets with 6000 Instances and 30% Sensor Anomalies

Table A.15.: Performance of the model with the combined model configurations evaluated on L6000 and H6000 with 30% sensor anomalies.

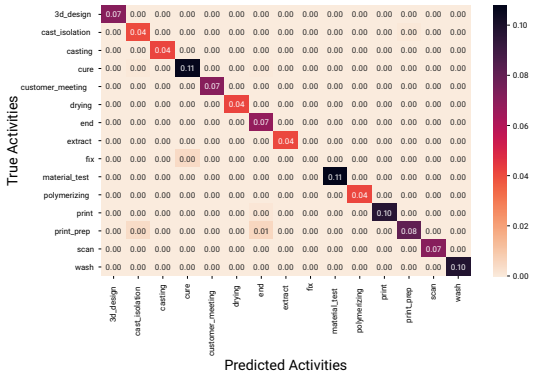
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000	norm_rnn	0.9793	0.9775	0.9759	0.9793	0.9771
	norm_cnn	0.9823	0.9807	0.9778	0.9823	0.9799
	kalman_rnn	0.9720	0.9696	0.9707	0.9720	0.9699
	kalman_cnn	0.9821	0.9805	0.9800	0.9821	0.9798
	kal_norm_rnn	0.9767	0.9747	0.9740	0.9767	0.9749
	kal_norm_cnn	0.9815	0.9799	0.9771	0.9815	0.9792
H6000	norm_rnn	0.9228	0.9142	0.8932	0.9228	0.9027
	norm_cnn	0.9183	0.9092	0.8889	0.9183	0.8979
	kalman_rnn	0.9005	0.8894	0.8730	0.9005	0.8811
	kalman_cnn	0.8972	0.8857	0.8685	0.8972	0.8760
	kal_norm_rnn	0.9218	0.9131	0.8922	0.9218	0.9021
	kal_norm_cnn	0.9156	0.9062	0.8859	0.9156	0.8956

Table A.16.: Performance of the model with the combined model configurations evaluated on **the decisions** of L6000 and H6000 with 30% sensor anomalies.

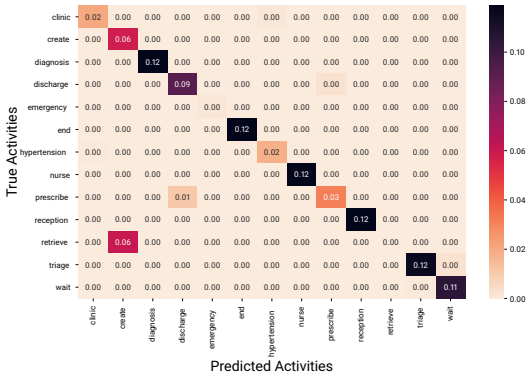
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Decisions	norm_rnn	0.9162	0.8794	0.8989	0.9162	0.9015
	norm_cnn	0.9285	0.8975	0.9080	0.9285	0.9169
	kalman_rnn	0.8867	0.8372	0.8757	0.8867	0.8639
	kalman_cnn	0.9274	0.8956	0.9173	0.9274	0.9131
	kal_norm_rnn	0.9057	0.8647	0.8897	0.9057	0.8944
	kal_norm_cnn	0.9251	0.8925	0.9049	0.9251	0.9135
H6000 Decisions	norm_rnn	0.8069	0.7674	0.7324	0.8069	0.7563
	norm_cnn	0.7956	0.7536	0.7220	0.7956	0.7444
	kalman_rnn	0.7510	0.7008	0.6811	0.7510	0.7017
	kalman_cnn	0.7428	0.6873	0.6404	0.7428	0.6752
	kal_norm_rnn	0.8044	0.7640	0.7298	0.8044	0.7546
	kal_norm_cnn	0.7889	0.7449	0.7141	0.7889	0.7382



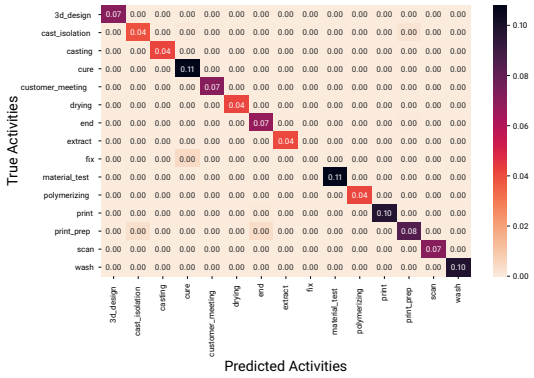
(a) Normalized RNN Hospital



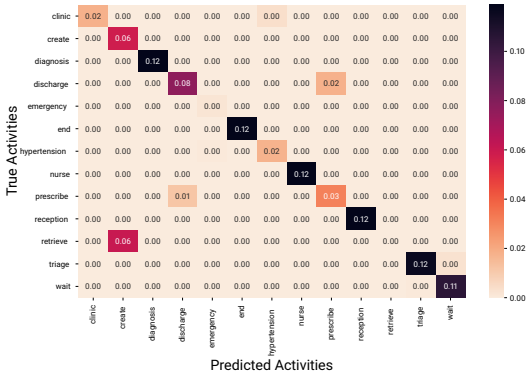
(b) Normalized RNN Laboratory



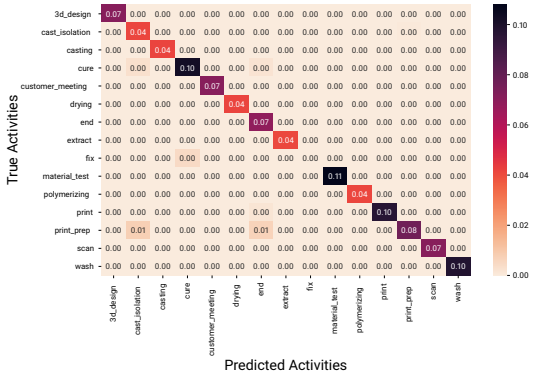
(c) Normalized CNN Hospital



(d) Normalized CNN Laboratory

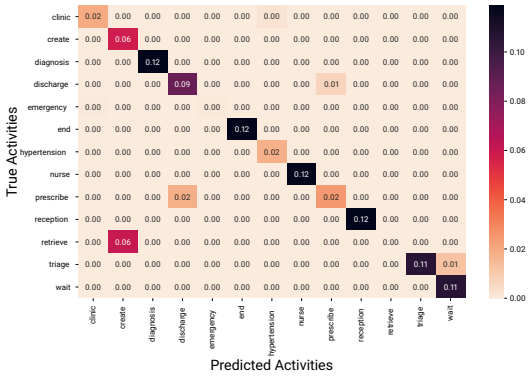


(e) Kalman RNN Hospital

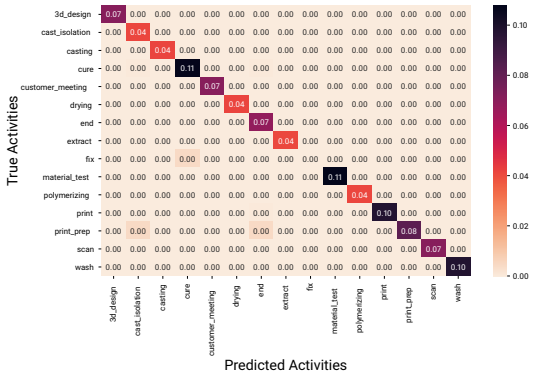


(f) Kalman RNN Laboratory

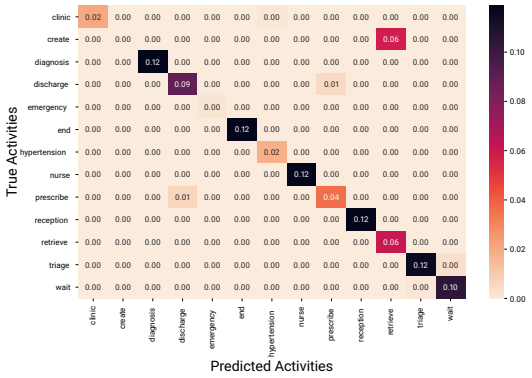
Figure A.8.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Flow Anomalies.



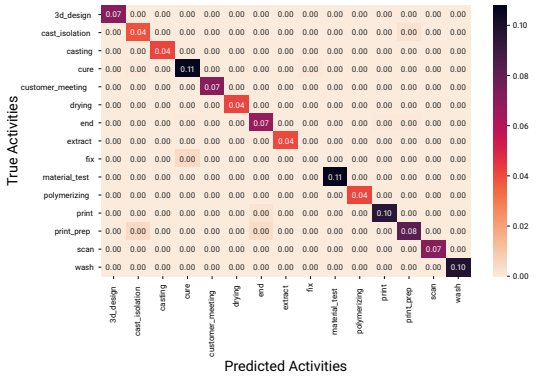
(g) Kalman CNN Hospital



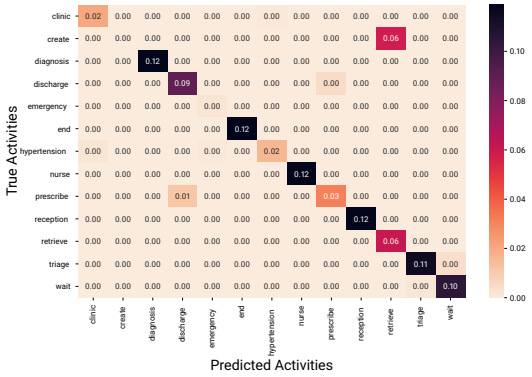
(h) Kalman CNN Laboratory



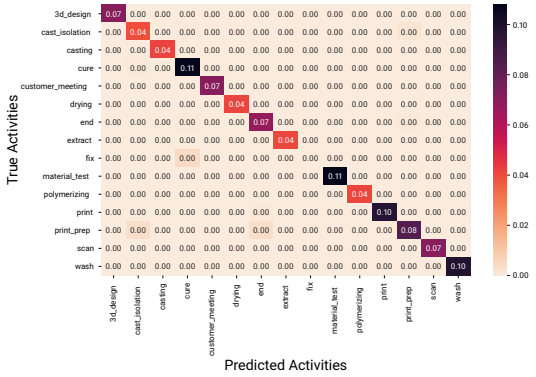
(i) Kalman Normalized RNN Hospital



(j) Kalman Normalized RNN Laboratory



(k) Kalman Normalized CNN Hospital



(l) Kalman Normalized CNN Laboratory

Figure A.8.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor Anomalies(cont.).

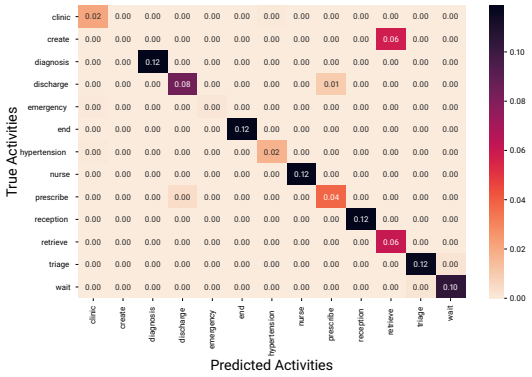
A.4.3. Experiment Results on the Train Datasets with 6000 Instances and 30% Flow Anomalies

Table A.17.: Performance of the model with the combined model configurations evaluated on L6000 and H6000 with 30% flow anomalies.

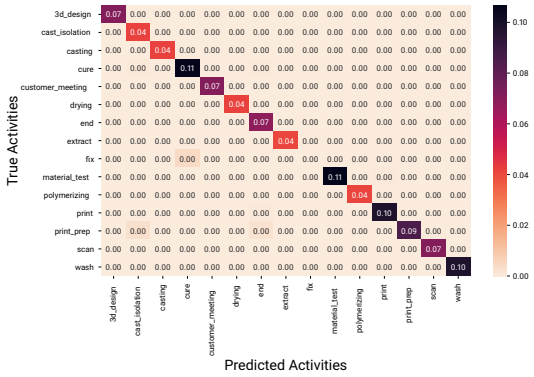
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000	norm_rnn	0.9839	0.9825	0.9798	0.9839	0.9816
	norm_cnn	0.9804	0.9787	0.9766	0.9804	0.9783
	kalman_rnn	0.9827	0.9812	0.9807	0.9827	0.9805
	kalman_cnn	0.9800	0.9783	0.9800	0.9800	0.9777
	kal_norm_rnn	0.9839	0.9825	0.9811	0.9839	0.9817
	kal_norm_cnn	0.9827	0.9812	0.9790	0.9827	0.9804
H6000	norm_rnn	0.9187	0.9097	0.8899	0.9187	0.8992
	norm_cnn	0.9119	0.9020	0.8824	0.9119	0.8917
	kalman_rnn	0.9035	0.8926	0.8731	0.9035	0.8827
	kalman_cnn	0.9042	0.8934	0.8755	0.9042	0.8837
	kal_norm_rnn	0.9202	0.9112	0.8909	0.9202	0.8999
	kal_norm_cnn	0.9133	0.9035	0.8851	0.9133	0.8921

Table A.18.: Performance of the model with the combined model configurations evaluated on **the decisions** of L6000 and H6000 with 30% flow anomalies.

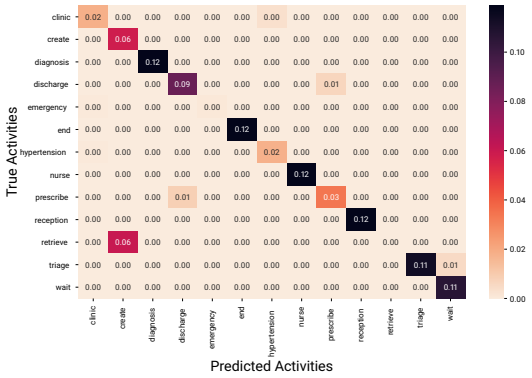
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Decisions	norm_rnn	0.9425	0.9178	0.9251	0.9425	0.9319
	norm_cnn	0.9251	0.8931	0.9096	0.9251	0.9162
	kalman_rnn	0.9311	0.9009	0.9231	0.9311	0.9168
	kalman_cnn	0.9199	0.8849	0.9169	0.9199	0.9020
	kal_norm_rnn	0.9364	0.9087	0.9246	0.9364	0.9245
	kal_norm_cnn	0.9334	0.9047	0.9180	0.9334	0.9213
H6000 Decisions	norm_rnn	0.7973	0.7561	0.7250	0.7973	0.7484
	norm_cnn	0.7808	0.7354	0.7071	0.7808	0.7288
	kalman_rnn	0.7589	0.7081	0.6833	0.7589	0.7063
	kalman_cnn	0.7720	0.7229	0.6853	0.7720	0.7134
	kal_norm_rnn	0.8004	0.7584	0.7269	0.8004	0.7489
	kal_norm_cnn	0.7830	0.7368	0.7133	0.7830	0.7290



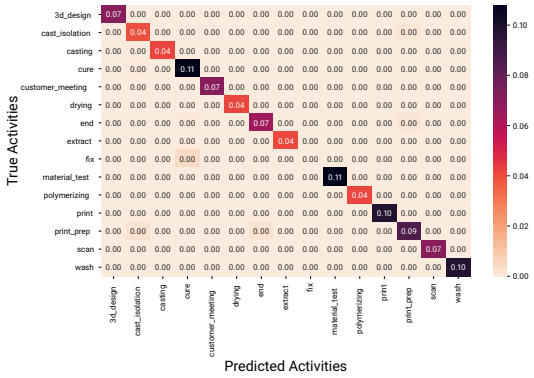
(a) Normalized RNN Hospital



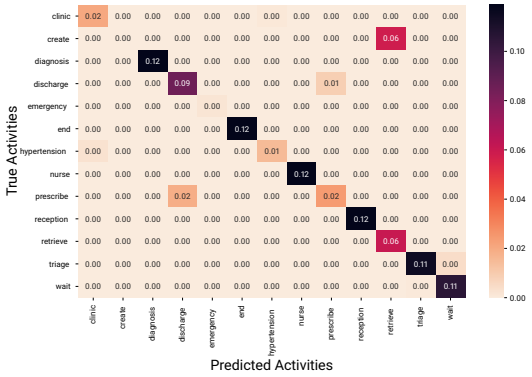
(b) Normalized RNN Laboratory



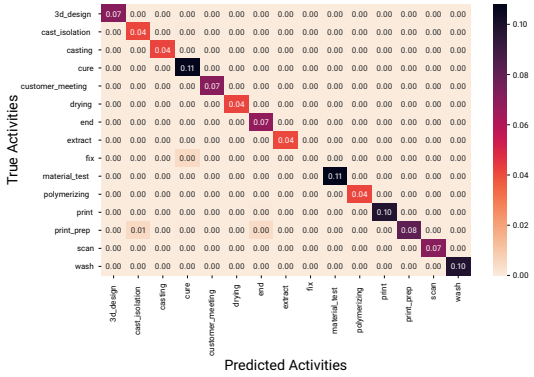
(c) Normalized CNN Hospital



(d) Normalized CNN Laboratory

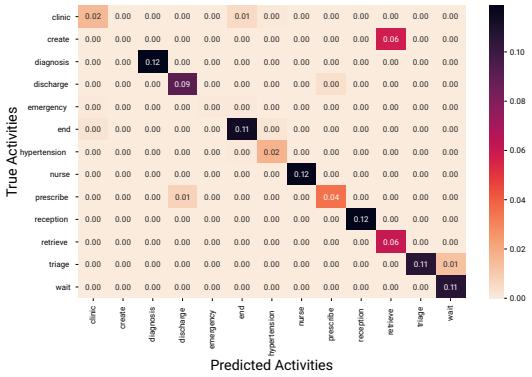


(e) Kalman RNN Hospital

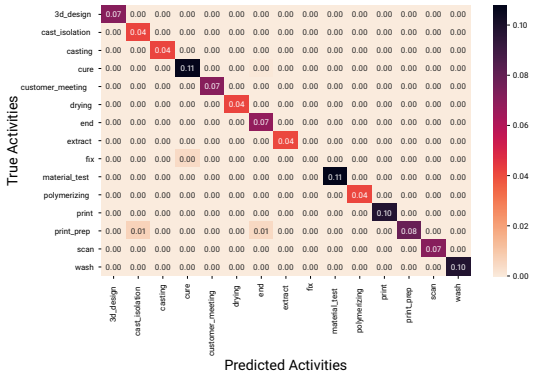


(f) Kalman RNN Laboratory

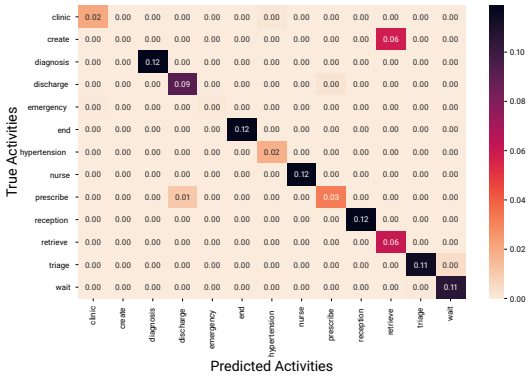
Figure A.9.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Flow Anomalies.



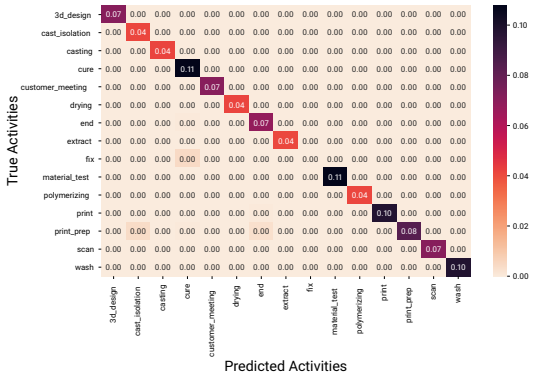
(g) Kalman CNN Hospital



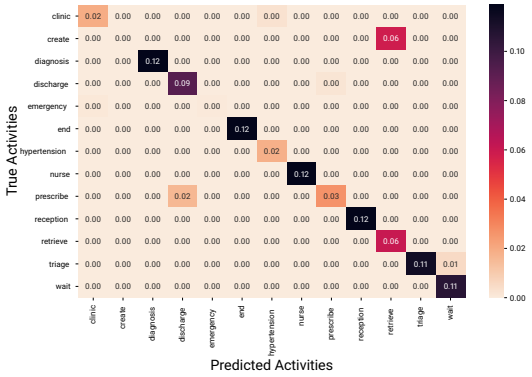
(h) Kalman CNN Laboratory



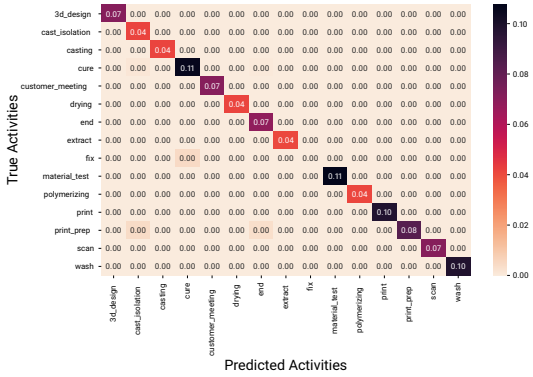
(i) Kalman Normalized RNN Hospital



(j) Kalman Normalized RNN Laboratory



(k) Kalman Normalized CNN Hospital



(l) Kalman Normalized CNN Laboratory

Figure A.9.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Flow Anomalies(cont.).

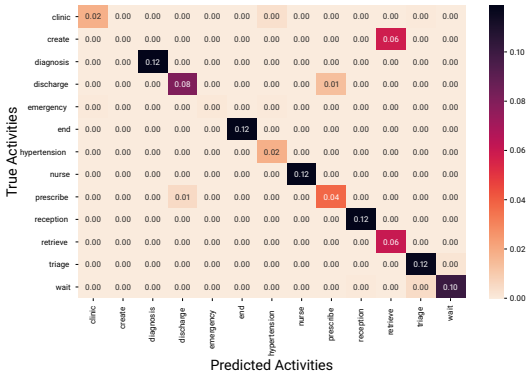
A.4.4. Experiment Results on the Train Datasets with 6000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies

Table A.19.: Performance of the model with the combined model configurations evaluated on L6000 and H6000 with 30% sensor anomalies and 30% flow anomalies.

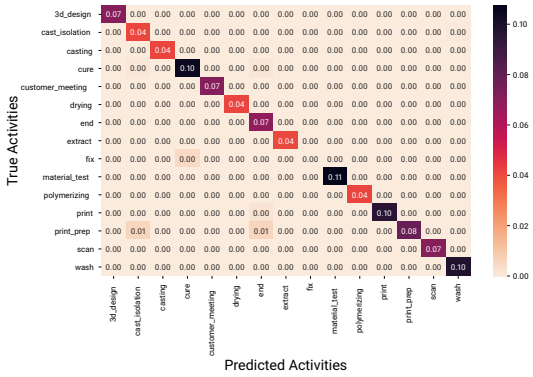
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000	norm_rnn	0.9743	0.9721	0.9727	0.9743	0.9722
	norm_cnn	0.9742	0.9720	0.9708	0.9742	0.9721
	kalman_rnn	0.9771	0.9751	0.9761	0.9771	0.9750
	kalman_cnn	0.9759	0.9738	0.9747	0.9759	0.9738
	kal_norm_rnn	0.9772	0.9752	0.9740	0.9772	0.9751
	kal_norm_cnn	0.9798	0.9780	0.9771	0.9798	0.9776
H6000	norm_rnn	0.9094	0.8993	0.8819	0.9094	0.8898
	norm_cnn	0.9084	0.8982	0.8793	0.9084	0.8887
	kalman_rnn	0.8996	0.8885	0.8705	0.8996	0.8799
	kalman_cnn	0.8689	0.8538	0.8263	0.8689	0.8398
	kal_norm_rnn	0.9174	0.9081	0.8881	0.9174	0.8975
	kal_norm_cnn	0.9109	0.9009	0.8832	0.9109	0.8918

Table A.20.: Performance of the model with the combined model configurations evaluated on **the decisions** of L6000 and H6000 with 30% sensor anomalies and 30% flow anomalies.

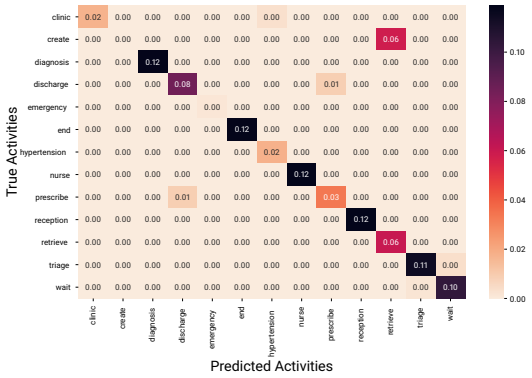
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Decisions	norm_rnn	0.8990	0.8548	0.8928	0.8990	0.8819
	norm_cnn	0.8969	0.8526	0.8826	0.8969	0.8860
	kalman_rnn	0.9078	0.8686	0.8989	0.9078	0.8949
	kalman_cnn	0.9029	0.8610	0.8916	0.9029	0.8848
	kal_norm_rnn	0.9105	0.8731	0.8988	0.9105	0.9022
	kal_norm_cnn	0.9190	0.8841	0.9058	0.9190	0.9055
H6000 Decisions	norm_rnn	0.7739	0.7289	0.7095	0.7739	0.7266
	norm_cnn	0.7752	0.7292	0.7048	0.7752	0.7270
	kalman_rnn	0.7492	0.6984	0.6756	0.7492	0.6994
	kalman_cnn	0.6754	0.6069	0.6221	0.6754	0.6143
	kal_norm_rnn	0.7932	0.7513	0.7192	0.7932	0.7430
	kal_norm_cnn	0.7775	0.7323	0.7078	0.7775	0.7295



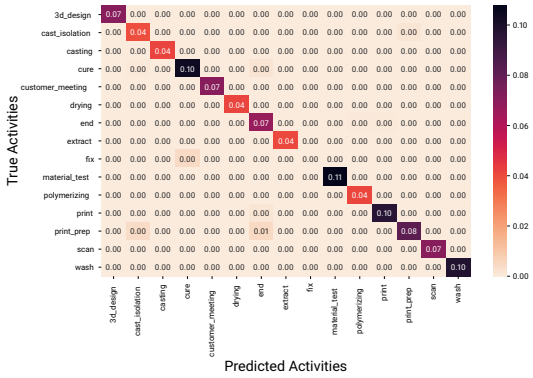
(a) Normalized RNN Hospital



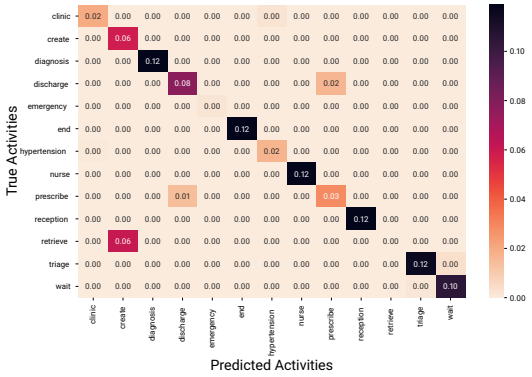
(b) Normalized RNN Laboratory



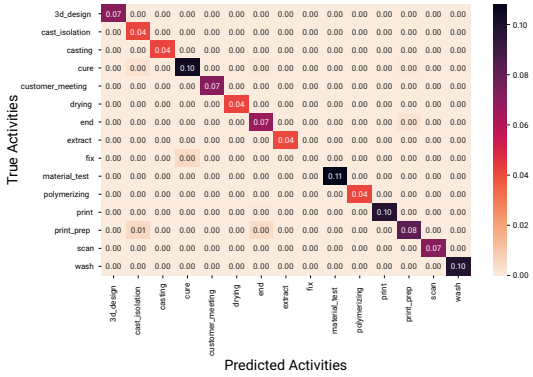
(c) Normalized CNN Hospital



(d) Normalized CNN Laboratory

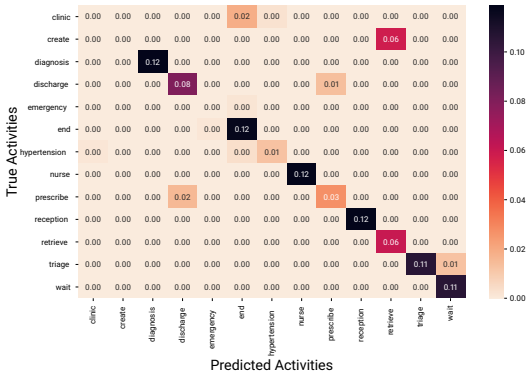


(e) Kalman RNN Hospital

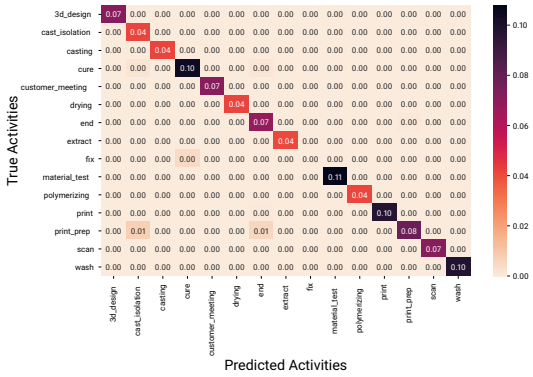


(f) Kalman RNN Laboratory

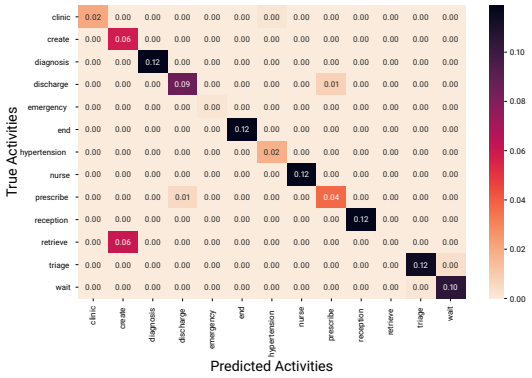
Figure A.10.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor and 30% Flow Anomalies.



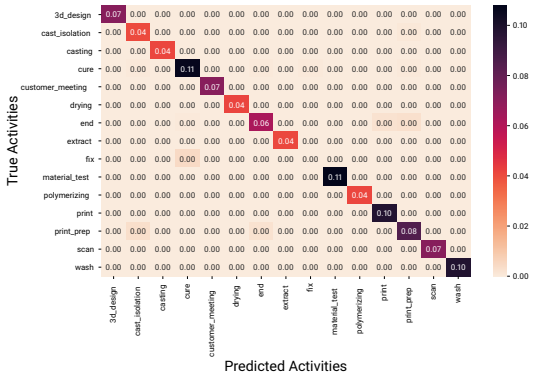
(g) Kalman CNN Hospital



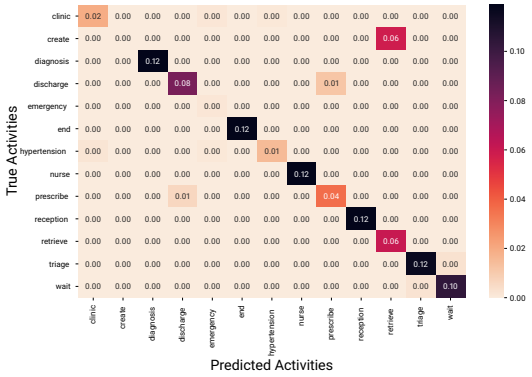
(h) Kalman CNN Laboratory



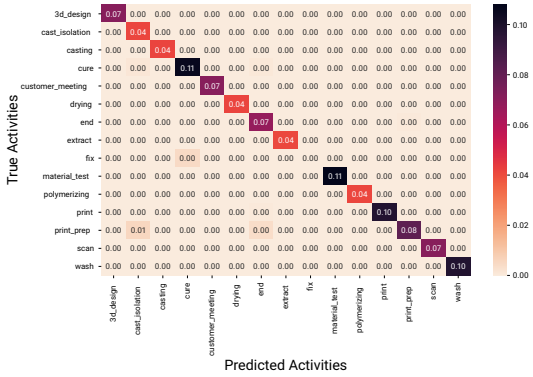
(i) Kalman Normalized RNN Hospital



(j) Kalman Normalized RNN Laboratory



(k) Kalman Normalized CNN Hospital



(l) Kalman Normalized CNN Laboratory

Figure A.10.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 6000 Instances with 30% Sensor and 30% Flow Anomalies (cont.).

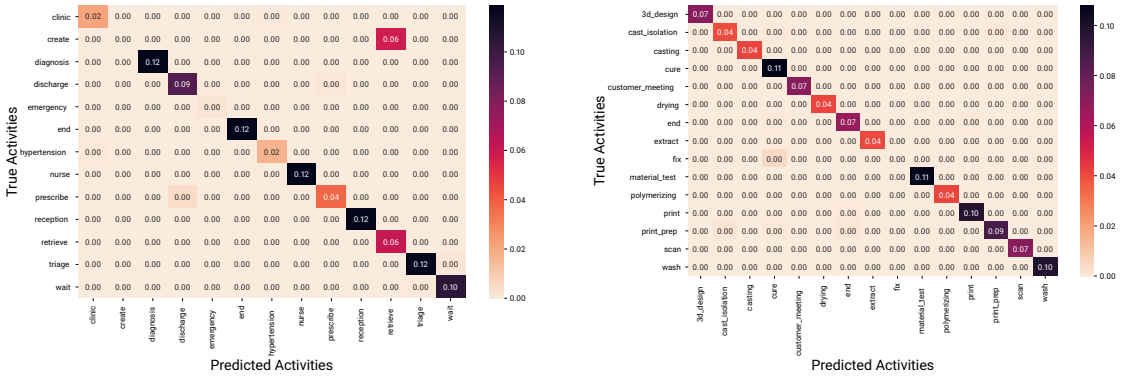
A.4.5. Experiment Results on the Train Datasets with 10000 Instances and no Anomalies

Table A.21.: Performance of the model with the combined model configurations evaluated on L10000 and H10000 without anomalies.

	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000	norm_rnn	0.9865	0.9854	0.9821	0.9865	0.9842
	norm_cnn	0.9911	0.9903	0.9882	0.9911	0.9890
	kalman_rnn	0.9836	0.9822	0.9798	0.9836	0.9812
	kalman_cnn	0.9808	0.9792	0.9767	0.9808	0.9785
	kal_norm_rnn	0.9777	0.9757	0.9775	0.9777	0.9753
	kal_norm_cnn	0.9894	0.9884	0.9848	0.9894	0.9870
H10000	norm_rnn	0.9306	0.9228	0.9010	0.9306	0.9109
	norm_cnn	0.9237	0.9152	0.8940	0.9237	0.9038
	kalman_rnn	0.9062	0.8958	0.8774	0.9062	0.8866
	kalman_cnn	0.8913	0.8790	0.8670	0.8913	0.8683
	kal_norm_rnn	0.9295	0.9216	0.8998	0.9295	0.9095
	kal_norm_cnn	0.9237	0.9152	0.8942	0.9237	0.9036

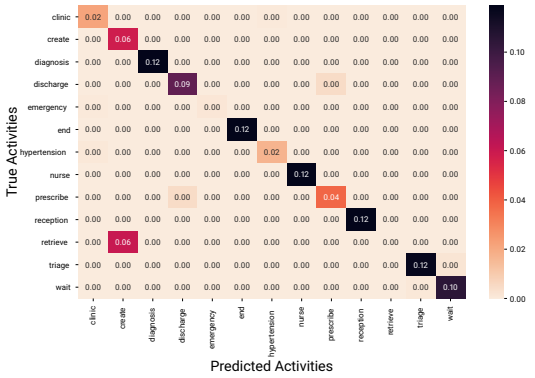
Table A.22.: Performance of the model with the combined model configurations evaluated on **the decisions** of L10000 and H10000 without anomalies.

	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000 Decisions	norm_rnn	0.9455	0.9219	0.9266	0.9455	0.9353
	norm_cnn	0.9638	0.9484	0.9519	0.9638	0.9553
	kalman_rnn	0.9337	0.9047	0.9181	0.9337	0.9191
	kalman_cnn	0.9226	0.8887	0.9031	0.9226	0.9102
	kal_norm_rnn	0.9098	0.8730	0.9268	0.9098	0.9066
	kal_norm_cnn	0.9569	0.9385	0.9382	0.9569	0.9473
H10000 Decisions	norm_rnn	0.8264	0.7907	0.7522	0.8264	0.7770
	norm_cnn	0.8092	0.7705	0.7347	0.8092	0.7591
	kalman_rnn	0.7654	0.7181	0.6927	0.7654	0.7159
	kalman_cnn	0.7279	0.6685	0.6736	0.7279	0.6586
	kal_norm_rnn	0.8236	0.7875	0.7491	0.8236	0.7733
	kal_norm_cnn	0.8092	0.7701	0.7351	0.8092	0.7586

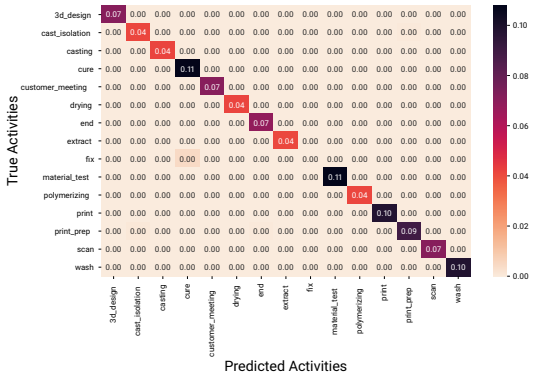


(a) Normalized RNN Hospital

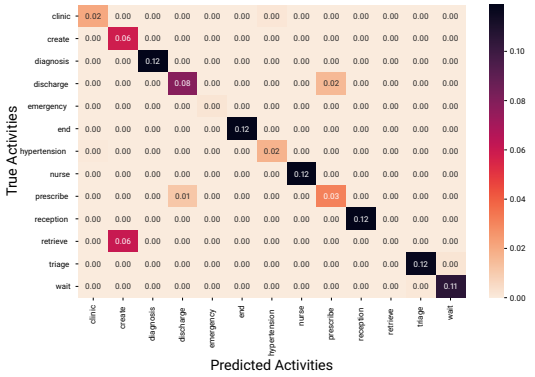
(b) Normalized RNN Laboratory



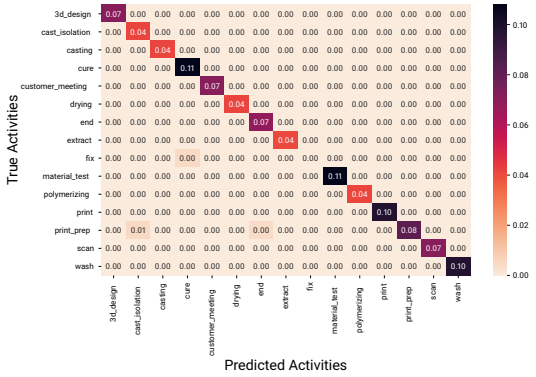
(c) Normalized CNN Hospital



(d) Normalized CNN Laboratory

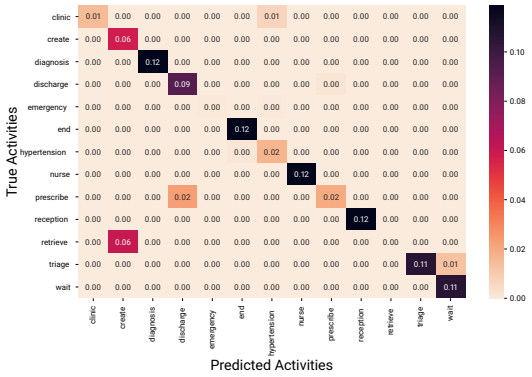


(e) Kalman RNN Hospital

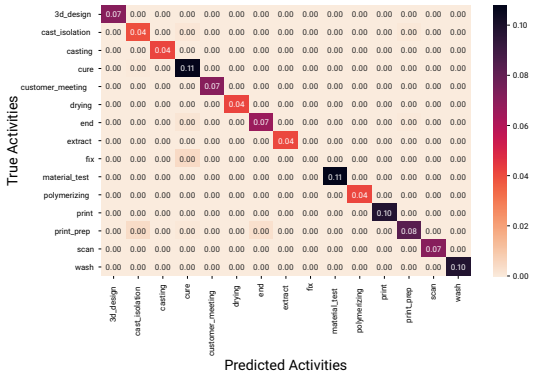


(f) Kalman RNN Laboratory

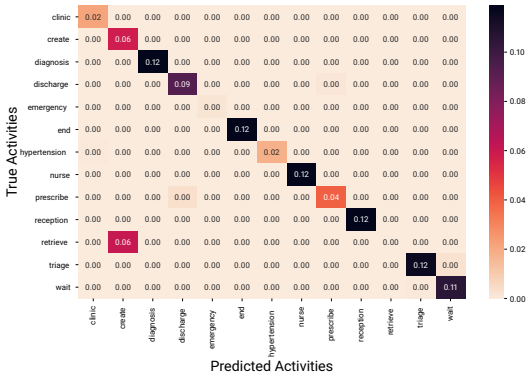
Figure A.11.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances and without Anomalies.



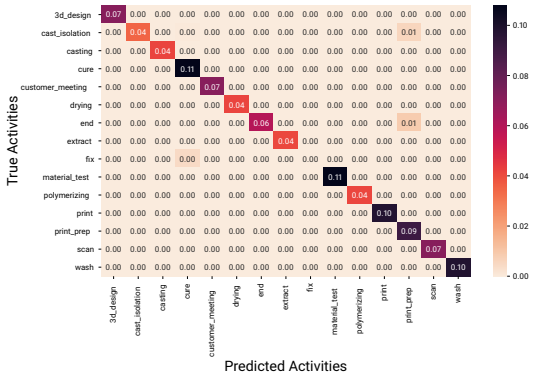
(g) Kalman CNN Hospital



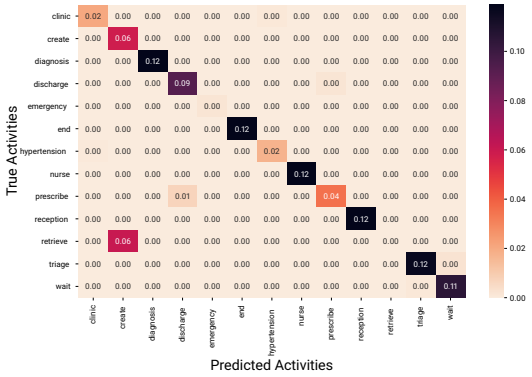
(h) Kalman CNN Laboratory



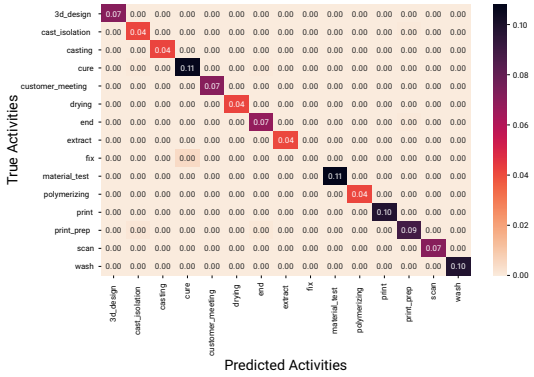
(i) Kalman Normalized RNN Hospital



(j) Kalman Normalized RNN Laboratory



(k) Kalman Normalized CNN Hospital



(l) Kalman Normalized CNN Laboratory

Figure A.11.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances and without Anomalies (cont.).

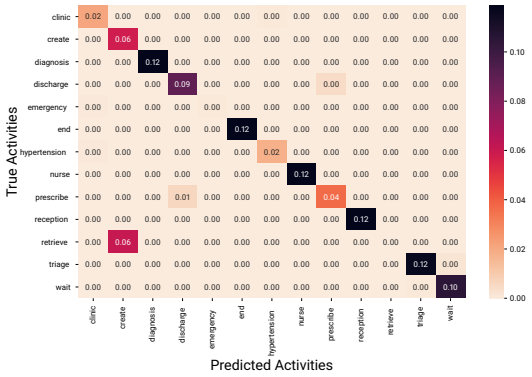
A.4.6. Experiment Results on the Train Datasets with 10000 Instances with 30% Sensor Anomalies and 30% Flow Anomalies

Table A.23.: Performance of the model with different model configurations evaluated on L10000 and H10000 with 30% sensor anomalies and 30% flow anomalies.

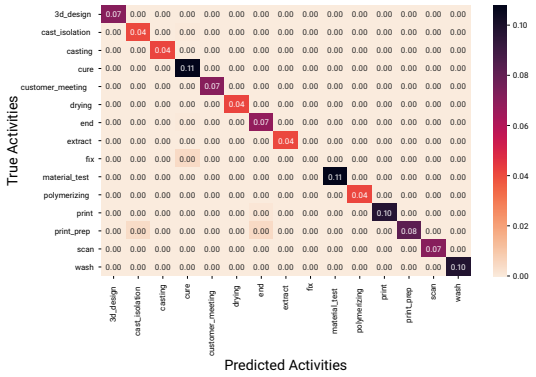
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000	norm_rnn	0.9820	0.9805	0.9783	0.9820	0.9797
	norm_cnn	0.9824	0.9809	0.9793	0.9824	0.9802
	kalman_rnn	0.9816	0.9800	0.9779	0.9816	0.9792
	kalman_cnn	0.9800	0.9783	0.9769	0.9800	0.9776
	kal_norm_rnn	0.9794	0.9776	0.9749	0.9794	0.9770
	kal_norm_cnn	0.9791	0.9773	0.9749	0.9791	0.9768
H10000	norm_rnn	0.9217	0.9130	0.8920	0.9217	0.9015
	norm_cnn	0.9173	0.9080	0.8882	0.9173	0.8970
	kalman_rnn	0.9078	0.8974	0.8776	0.9078	0.8869
	kalman_cnn	0.8719	0.8573	0.8248	0.8719	0.8398
	kal_norm_rnn	0.9247	0.9162	0.8952	0.9247	0.9045
	kal_norm_cnn	0.9164	0.9071	0.8867	0.9164	0.8962

Table A.24.: Performance of the model with the combined model configurations evaluated on **the decisions** of L10000 and H10000 with 30% sensor anomalies and 30% flow anomalies.

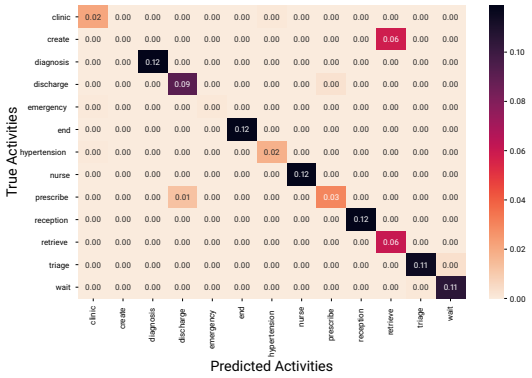
	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000 Decisions	norm_rnn	0.9278	0.8959	0.9124	0.9278	0.9133
	norm_cnn	0.9301	0.8996	0.9157	0.9301	0.9179
	kalman_rnn	0.9258	0.8930	0.9101	0.9258	0.9108
	kalman_cnn	0.9193	0.8838	0.9038	0.9193	0.8992
	kal_norm_rnn	0.9164	0.8805	0.8957	0.9164	0.9051
	kal_norm_cnn	0.9153	0.8789	0.8941	0.9153	0.9028
H10000 Decisions	norm_rnn	0.8041	0.7643	0.7296	0.8041	0.7535
	norm_cnn	0.7953	0.7524	0.7220	0.7953	0.7440
	kalman_rnn	0.7694	0.7213	0.6946	0.7694	0.7173
	kalman_cnn	0.6912	0.6253	0.6136	0.6912	0.6180
	kal_norm_rnn	0.8117	0.7733	0.7382	0.8117	0.7614
	kal_norm_cnn	0.7932	0.7514	0.7201	0.7932	0.7433



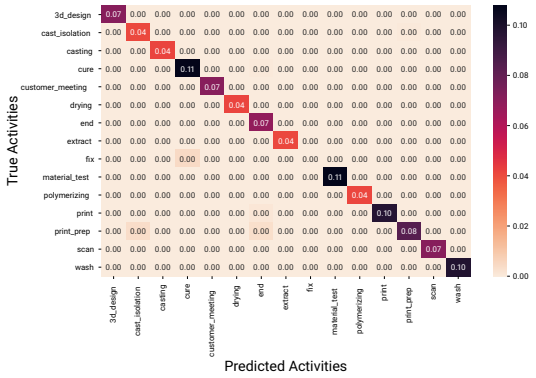
(a) Normalized RNN Hospital



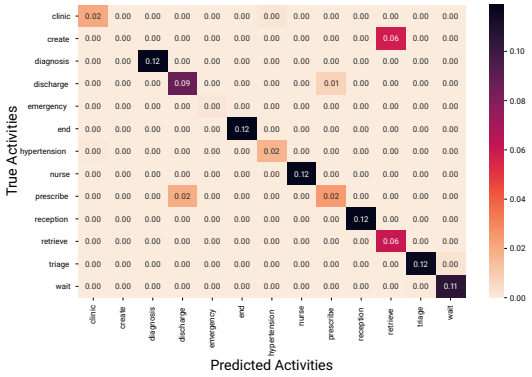
(b) Normalized RNN Laboratory



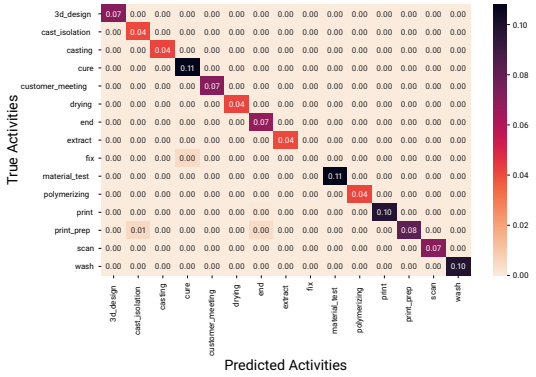
(c) Normalized CNN Hospital



(d) Normalized CNN Laboratory

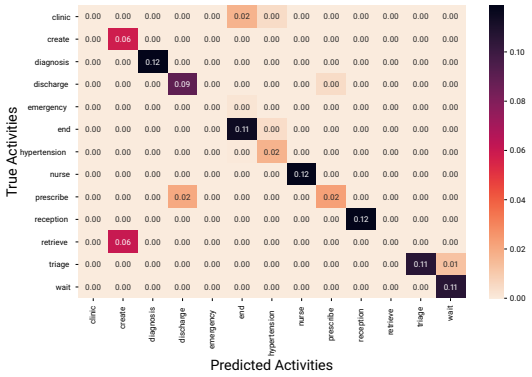


(e) Kalman RNN Hospital

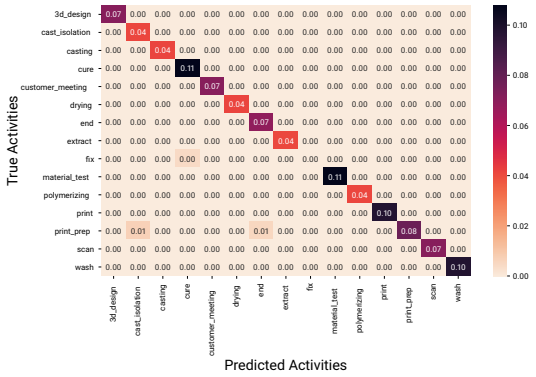


(f) Kalman RNN Laboratory

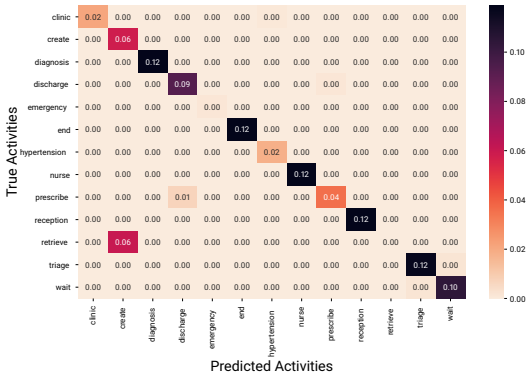
Figure A.12.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances and 30% Sensor and 30% Flow Anomalies.



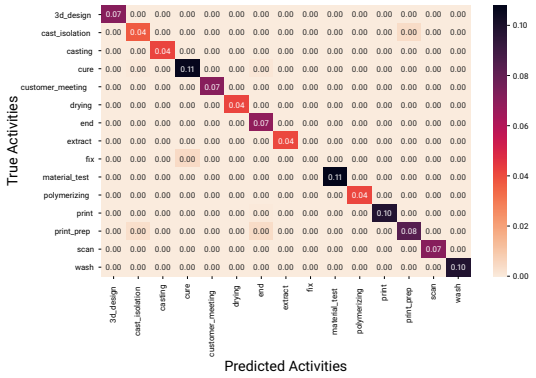
(g) Kalman CNN Hospital



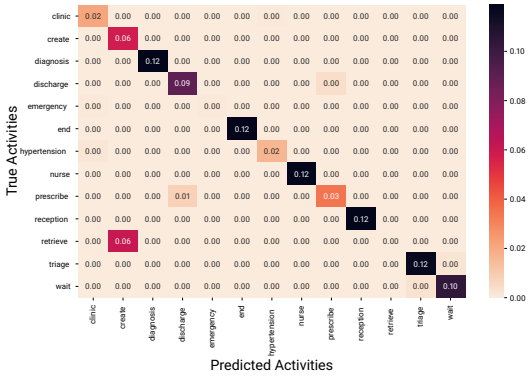
(h) Kalman CNN Laboratory



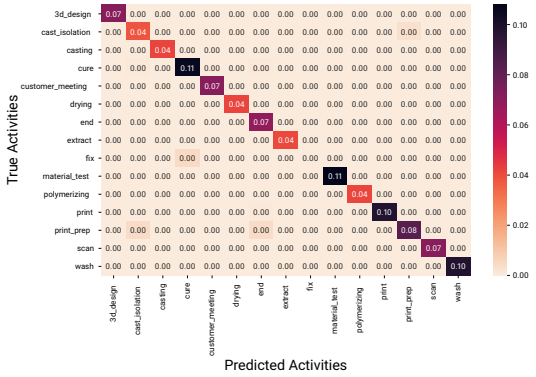
(i) Kalman Normalized RNN Hospital



(j) Kalman Normalized RNN Laboratory



(k) Kalman Normalized CNN Hospital



(l) Kalman Normalized CNN Laboratory

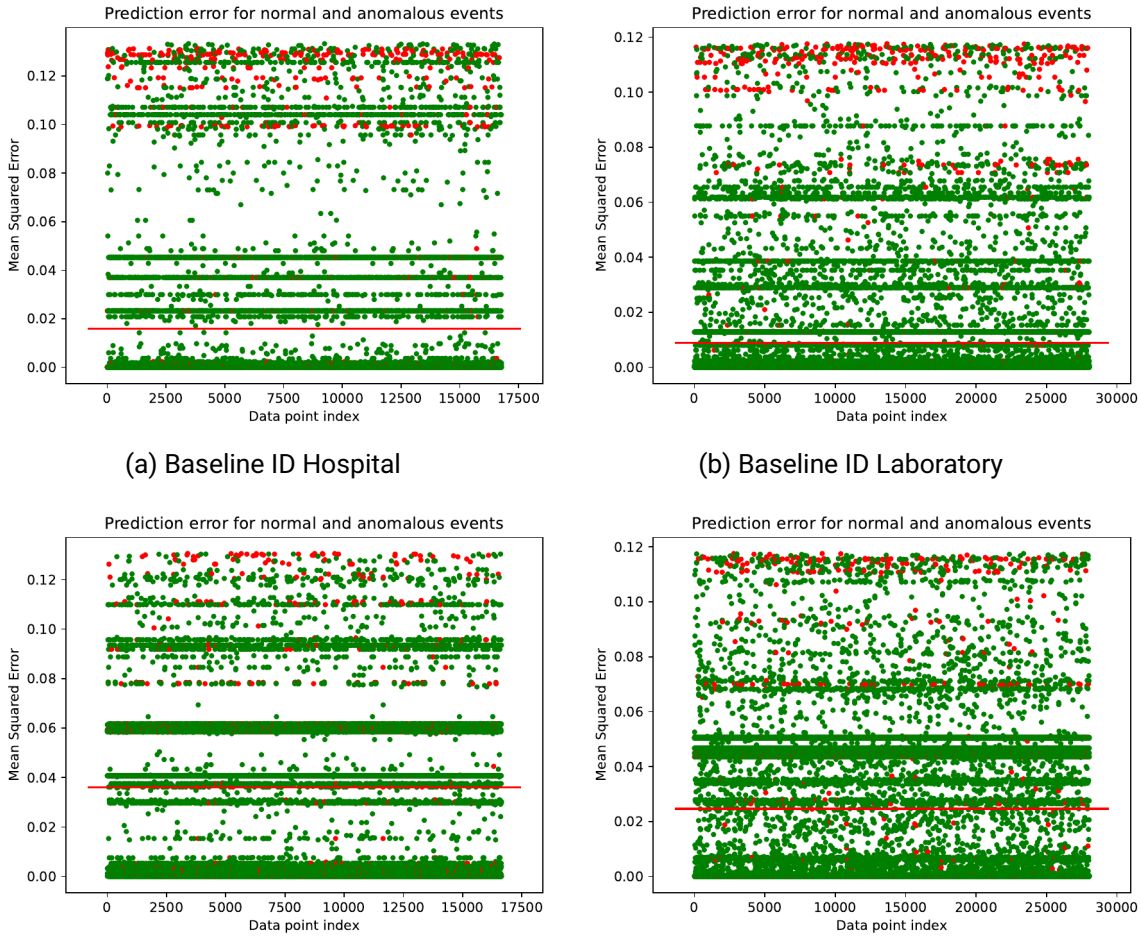
Figure A.12.: Normalized Confusion Matricies with the annotated Model Configuration on the annotated Dataset with 10000 Instances and 30% Sensor and 30% Flow Anomalies (cont.).

A.5. Anomaly Classification

A.5.1. Training on 6000 instances for event anomaly classification

Table A.25.: Performance of the anomaly classification system on 2000 instances for instence event anomaly detection. The prediction system is trained on L6000/H6000 with 30% sensor and 30% flow anomalies.

	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L6000 Model	norm_rnn	0.9277	0.2113	0.1751	0.3985	0.2433
	norm_cnn	0.9170	0.1895	0.1620	0.3744	0.2261
	kal_norm_rnn	0.9059	0.1729	0.1491	0.3740	0.2132
	kal_norm_cnn	0.9269	0.2055	0.1754	0.3733	0.2387
	baseline_sensor	0.5362	0.0238	0.0434	0.6528	0.0814
	baseline_id	0.7859	0.0622	0.0634	0.4502	0.1111
	baseline_attr	0.6525	0.0394	0.0514	0.5806	0.0945
H6000 Model	norm_rnn	0.7907	0.1125	0.1141	0.4723	0.1838
	norm_cnn	0.7922	0.1091	0.1128	0.4567	0.1810
	kal_norm_rnn	0.8112	0.1274	0.1220	0.4737	0.1940
	kal_norm_cnn	0.7972	0.1215	0.1231	0.4658	0.1947
	baseline_sensor	0.4733	0.0291	0.0690	0.6920	0.1255
	baseline_id	0.7413	0.0914	0.0983	0.5297	0.1659
	baseline_attr	0.6918	0.0572	0.0831	0.4847	0.1418



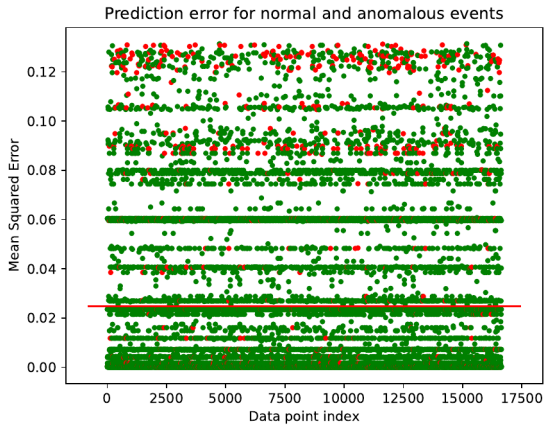
(a) Baseline ID Hospital

(b) Baseline ID Laboratory

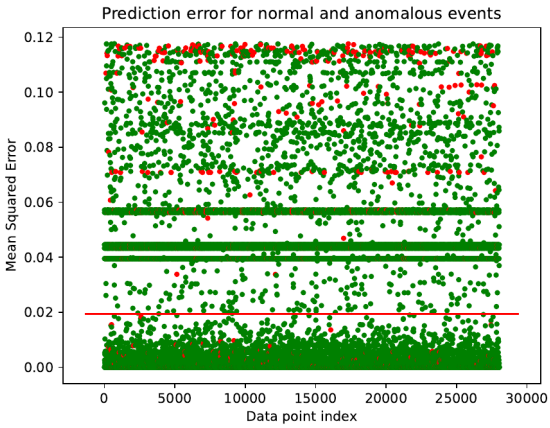
(c) Baseline Sensor Hospital

(d) Baseline Sensor Laboratory

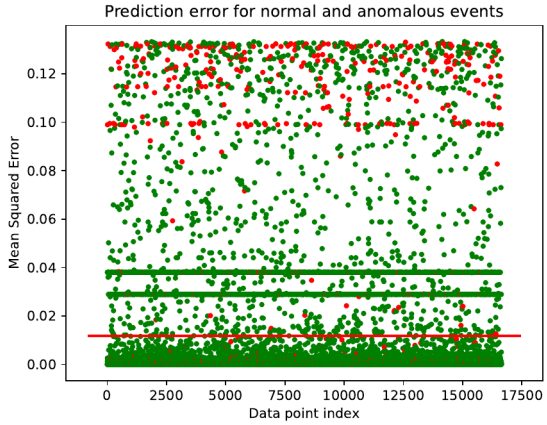
Figure A.13.: Scatterplots of the prediction error of events for anomaly classification. The green dots are normal events whereas the red dots are actual anoamlies. The line indicates the selected threshold by the average heuristic.



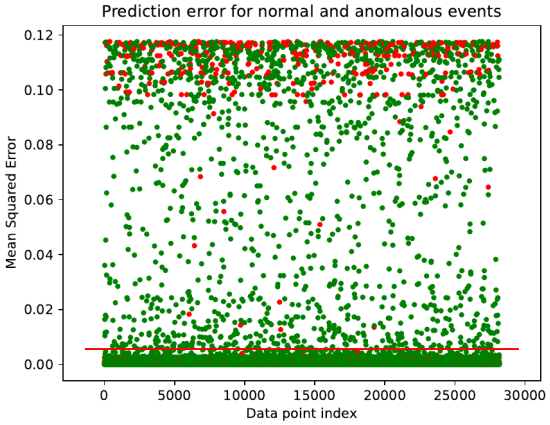
(e) Baseline Attributes Hospital



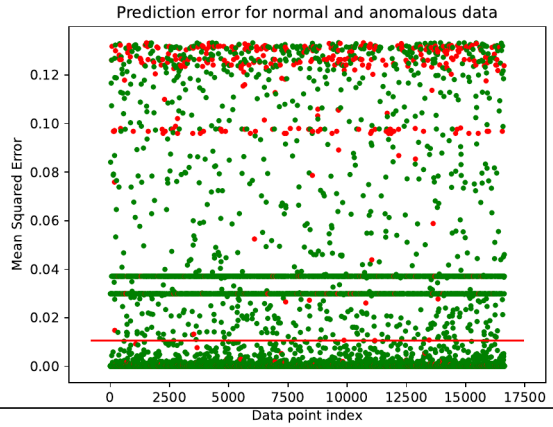
(f) Baseline Attributes Laboratory



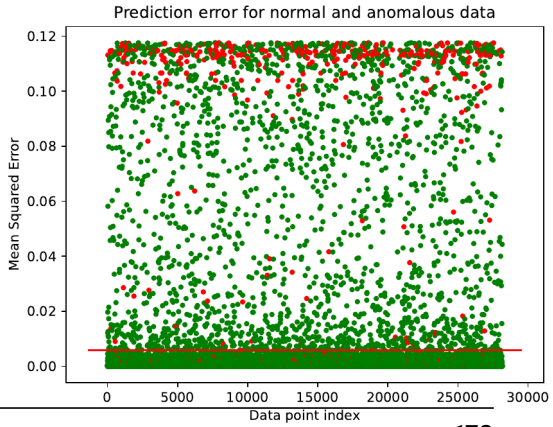
(g) Normalized RNN Hospital



(h) Normalized RNN Laboratory

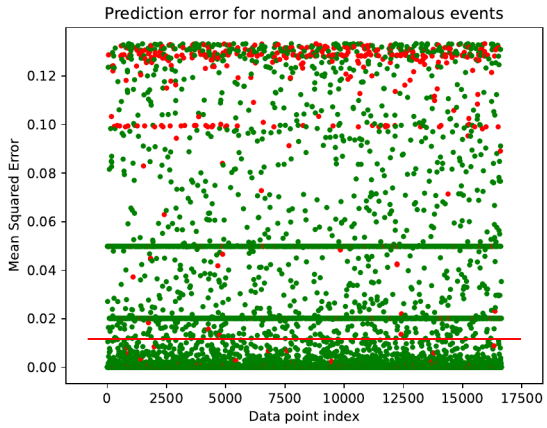


(i) Kalman Normalized RNN Hospital

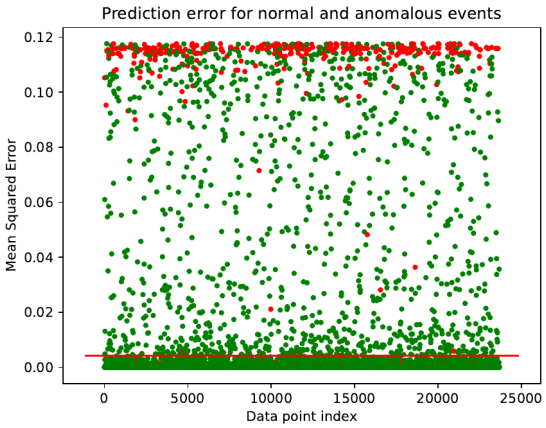


(j) Kalman Normalized RNN Laboratory

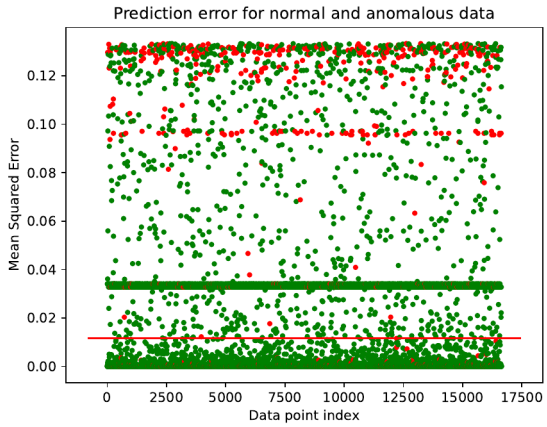
Figure A.13.: Scatterplots of the prediction error of events for anomaly classification. The green dots are normal events whereas the red dots are actual anoamlies. The line indicates the selected threshold by the average heuristic (cont.).



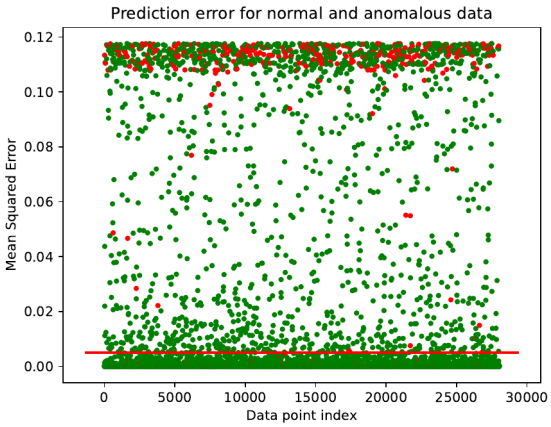
(k) Normalized CNN Hospital



(l) Normalized CNN Laboratory



(m) Kalman Normalized CNN Hospital



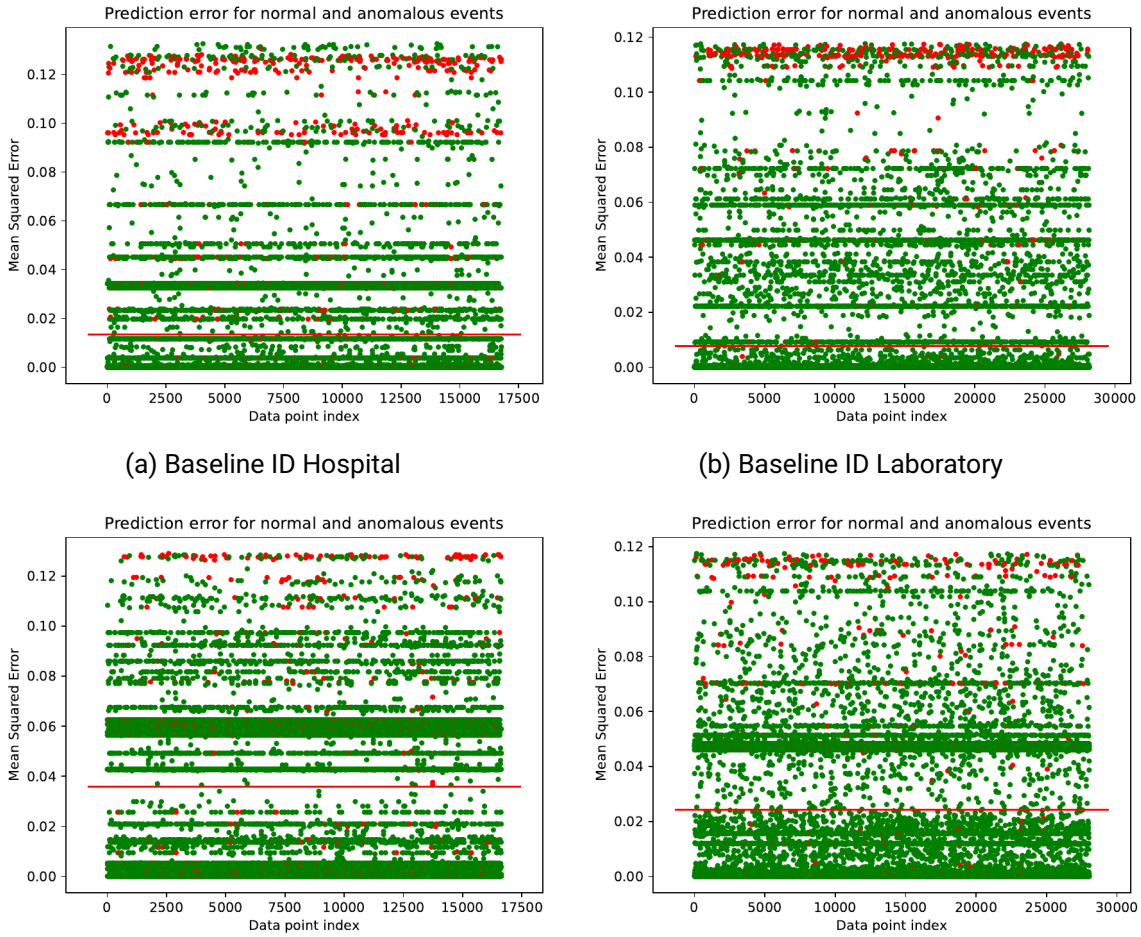
(n) Kalman Normalized CNN Laboratory

Figure A.13.: Scatterplots of the prediction error of events for anomaly classification. The green dots are normal events whereas the red dots are actual anoamlies. The line indicates the selected threshold by the average heuristic (cont.).

A.5.2. Training on 10000 instances for event anomaly classification

Table A.26.: Performance of the anomaly classification system on 2000 instances for instance event anomaly detection. The prediction system is trained on L10000/H10000 with 30% sensor and 30% flow anomalies.

	Model Configuration	Accuracy	Kappa	Precision	Recall	F_1
L10000 Model	norm_rnn	0.9258	0.1959	0.1658	0.3701	0.2290
	norm_cnn	0.9290	0.2056	0.1813	0.3504	0.2390
	kal_norm_rnn	0.9158	0.1787	0.1529	0.3645	0.2154
	kal_norm_cnn	0.9238	0.1889	0.1605	0.3634	0.2227
	baseline_sensor	0.5559	0.0277	0.0444	0.6616	0.0832
	baseline_id	0.8419	0.0977	0.0886	0.4155	0.1461
	baseline_attr	0.6550	0.0327	0.0462	0.5459	0.0851
H10000 Model	norm_rnn	0.8133	0.1376	0.1328	0.4715	0.2072
	norm_cnn	0.8100	0.1217	0.1188	0.4600	0.1888
	kal_norm_rnn	0.8172	0.1268	0.1266	0.4354	0.1961
	kal_norm_cnn	0.8062	0.1199	0.1223	0.4420	0.1915
	baseline_sensor	0.4608	0.0133	0.0544	0.6272	0.1001
	baseline_id	0.7430	0.0808	0.0939	0.4826	0.1572
	baseline_attr	0.6941	0.0630	0.0855	0.5046	0.1462



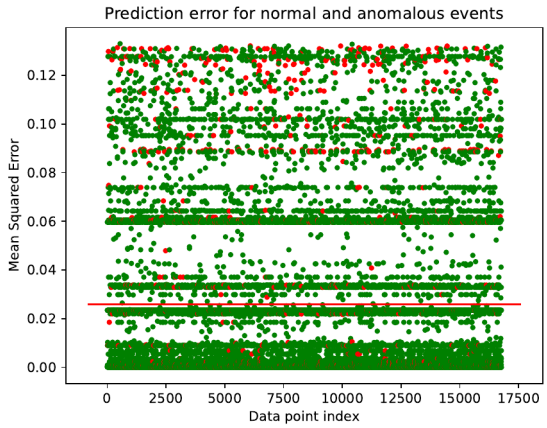
(a) Baseline ID Hospital

(b) Baseline ID Laboratory

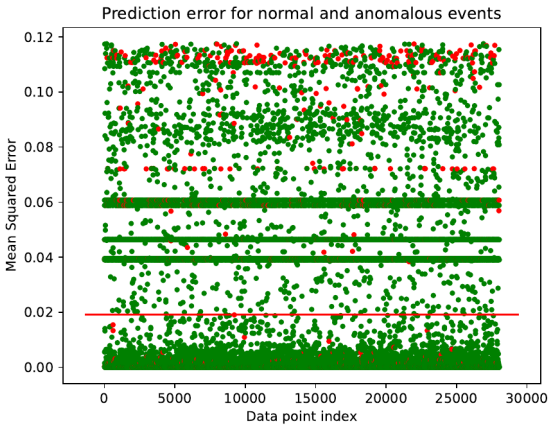
(c) Baseline Sensor Hospital

(d) Baseline Sensor Laboratory

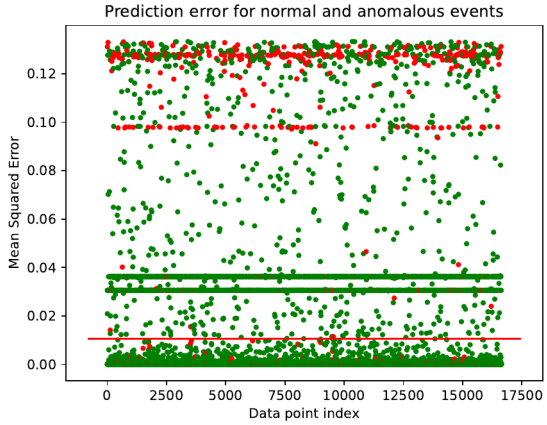
Figure A.14.: Scatterplots of the prediction error of events for anomaly classification. The green dots are normal events whereas the red dots are actual anoamlies. The line indicates the selected threshold by the average heuristic.



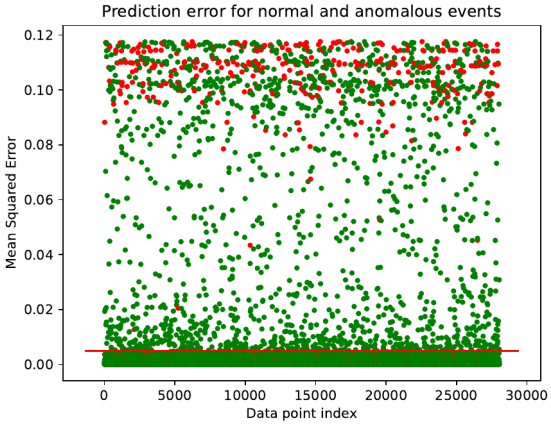
(e) Baseline Attributes Hospital



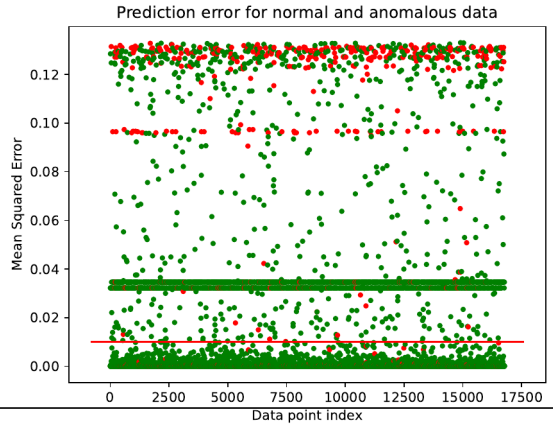
(f) Baseline Attributes Laboratory



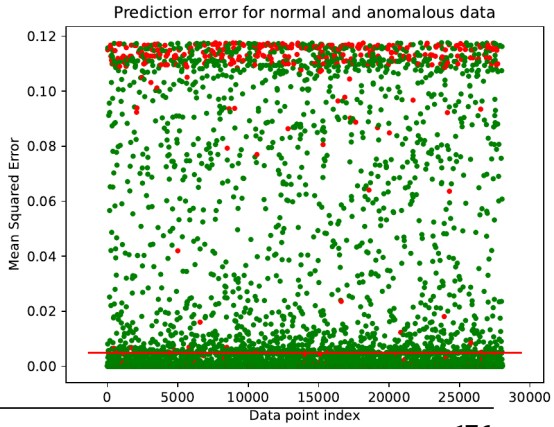
(g) Normalized RNN Hospital



(h) Normalized RNN Laboratory

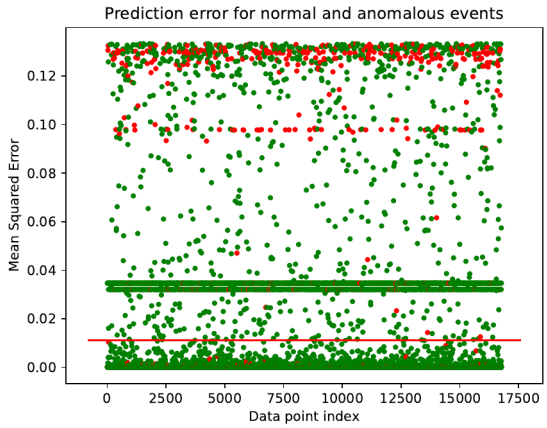


(i) Kalman Normalized RNN Hospital

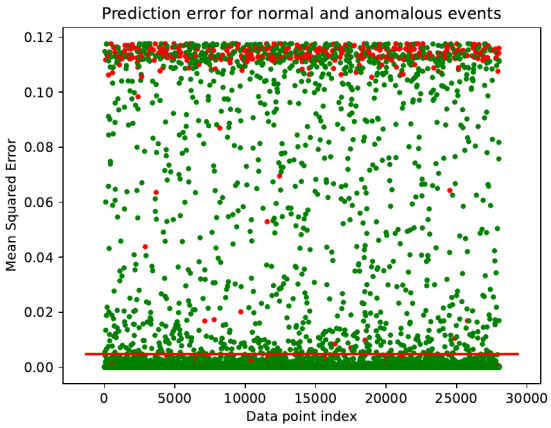


(j) Kalman Normalized RNN Laboratory

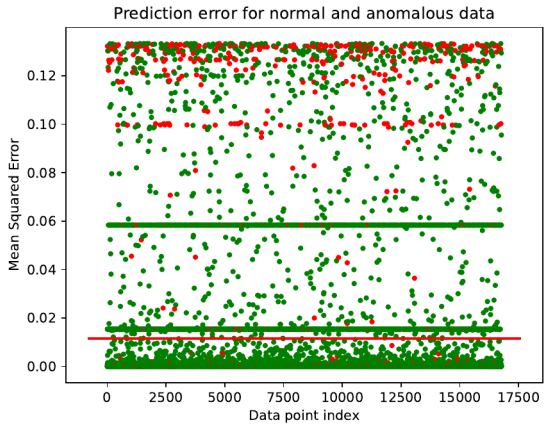
Figure A.14.: Scatterplots of the prediction error of events for anomaly classification. The green dots are normal events whereas the red dots are actual anoamlies. The line indicates the selected threshold by the average heuristic (cont.).



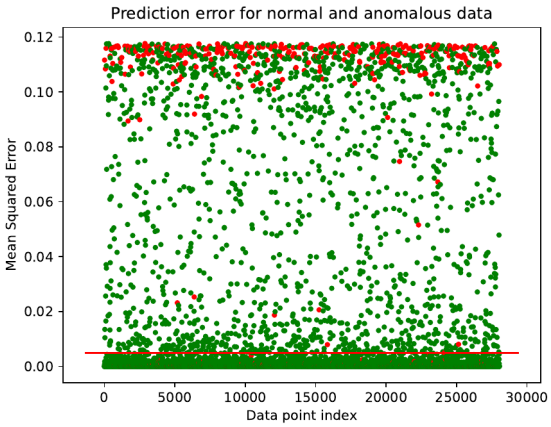
(k) Normalized CNN Hospital



(l) Normalized CNN Laboratory



(m) Kalman Normalized CNN Hospital



(n) Kalman Normalized CNN Laboratory

Figure A.14.: Scatterplots of the prediction error of events for anomaly classification. The green dots are normal events whereas the red dots are actual anoamlies. The line indicates the selected threshold by the average heuristic (cont.).

A.5.3. Grid Search for Anomaly Classification Task

Table A.27.: Performance of the model with different pre-processing configurations for anomaly classification. The threshold is set by a grid search. The model is trained on L1000 with 30% sensor and 30% flow anomalies. The evaluation data contain 30% flow and 30% sensor anomalies.

Model	Threshold	Accuracy	Kappa	Precision	Recall	F_1
Normalized RNN	0.1400	0.9701	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9701	0.0000	0.0000	0.0000	0.0000
	0.1200	0.9701	0.0000	0.0000	0.0000	0.0000
	0.1100	0.9689	0.2254	0.4444	0.1623	0.2378
	0.1000	0.9642	0.3215	0.3789	0.3079	0.3397
	0.0900	0.9605	0.3288	0.3441	0.3544	0.3492
	0.0800	0.9580	0.3198	0.3214	0.3640	0.3414
	0.0700	0.9560	0.3111	0.3047	0.3687	0.3337
	0.0600	0.9544	0.3027	0.2921	0.3687	0.3259
	0.0500	0.9526	0.2941	0.2790	0.3699	0.3181
	0.0400	0.9504	0.2847	0.2651	0.3723	0.3097
	0.0300	0.9477	0.2727	0.2494	0.3723	0.2987
	0.0200	0.9438	0.2581	0.2303	0.3759	0.2856
	0.0100	0.9371	0.2346	0.2032	0.3783	0.2644
	0.0000	0.0299	0.0000	0.0299	1.0000	0.0581
Normalized CNN	0.1400	0.9692	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9692	0.0000	0.0000	0.0000	0.0000
	0.1200	0.9692	0.0000	0.0000	0.0000	0.0000
	0.1100	0.9650	0.3399	0.4111	0.3163	0.3576
	0.1000	0.9602	0.3291	0.3517	0.3476	0.3497
	0.0900	0.9577	0.3169	0.3262	0.3523	0.3387
	0.0800	0.9558	0.3101	0.3109	0.3581	0.3328
	0.0700	0.9539	0.3001	0.2954	0.3581	0.3237
	0.0600	0.9523	0.2925	0.2834	0.3592	0.3168
	0.0500	0.9503	0.2840	0.2701	0.3615	0.3092
	0.0400	0.9486	0.2768	0.2598	0.3627	0.3027
	0.0300	0.9462	0.2666	0.2465	0.3627	0.2935
	0.0200	0.9431	0.2552	0.2314	0.3650	0.2833
	0.0100	0.9374	0.2348	0.2075	0.3662	0.2649
	0.0000	0.0308	0.0000	0.0308	1.0000	0.0597
	0.1400	0.9689	0.0000	0.0000	0.0000	0.0000

Kalman Normalized RNN	0.1300	0.9689	0.0000	0.0000	0.0000	0.0000
	0.1200	0.9689	0.0000	0.0000	0.0000	0.0000
	0.1100	0.9659	0.3246	0.4276	0.2841	0.3414
	0.1000	0.9613	0.3292	0.3665	0.3333	0.3491
	0.0900	0.9588	0.3201	0.3402	0.3425	0.3413
	0.0800	0.9557	0.3071	0.3116	0.3505	0.3299
	0.0700	0.9523	0.2898	0.2847	0.3505	0.3142
	0.0600	0.9504	0.2809	0.2712	0.3517	0.3062
	0.0500	0.9483	0.2714	0.2580	0.3517	0.2976
	0.0400	0.9459	0.2618	0.2446	0.3528	0.2889
	0.0300	0.9435	0.2519	0.2321	0.3528	0.2800
	0.0200	0.9401	0.2397	0.2170	0.3540	0.2690
	0.0100	0.9316	0.2177	0.1898	0.3654	0.2498
	0.0000	0.0311	0.0000	0.0311	1.0000	0.0604
Kalman Normalized CNN	0.1400	0.9700	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9700	0.0000	0.0000	0.0000	0.0000
	0.1200	0.9700	0.0000	0.0000	0.0000	0.0000
	0.1100	0.9697	0.3379	0.4904	0.2747	0.3521
	0.1000	0.9635	0.3425	0.3808	0.3436	0.3613
	0.0900	0.9590	0.3156	0.3270	0.3472	0.3368
	0.0800	0.9565	0.3062	0.3063	0.3543	0.3286
	0.0700	0.9543	0.2943	0.2879	0.3543	0.3177
	0.0600	0.9519	0.2830	0.2708	0.3555	0.3075
	0.0500	0.9497	0.2723	0.2560	0.3555	0.2977
	0.0400	0.9471	0.2623	0.2420	0.3579	0.2887
	0.0300	0.9443	0.2517	0.2283	0.3591	0.2791
	0.0200	0.9405	0.2375	0.2113	0.3603	0.2664
	0.0100	0.9325	0.2132	0.1841	0.3639	0.2445
	0.0000	0.0300	0.0000	0.0300	1.0000	0.0583
Baseline Sensor	0.1400	0.9680	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9680	0.0000	0.0000	0.0000	0.0000
	0.1200	0.9680	0.0000	0.0000	0.0000	0.0000
	0.1100	0.9658	0.1633	0.3849	0.1137	0.1756
	0.1000	0.9603	0.1743	0.2754	0.1483	0.1928
	0.0900	0.9579	0.1715	0.2491	0.1561	0.1919
	0.0800	0.9536	0.1805	0.2260	0.1862	0.2042
	0.0700	0.9442	0.1996	0.2048	0.2575	0.2281

	0.0600	0.9368	0.1802	0.1762	0.2653	0.2117
	0.0500	0.9082	0.1280	0.1180	0.2887	0.1676
	0.0400	0.5663	0.0302	0.0472	0.6544	0.0881
	0.0300	0.5601	0.0289	0.0466	0.6544	0.0869
	0.0200	0.5495	0.0274	0.0458	0.6589	0.0856
	0.0100	0.4949	0.0224	0.0432	0.6990	0.0814
	0.0000	0.0320	0.0000	0.0320	1.0000	0.0620
Baseline ID	0.1400	0.9666	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9666	0.0000	0.0000	0.0000	0.0000
	0.1200	0.9666	0.0000	0.0000	0.0000	0.0000
	0.1100	0.9669	0.3499	0.5086	0.2853	0.3655
	0.1000	0.9618	0.3320	0.4056	0.3098	0.3513
	0.0900	0.9611	0.3273	0.3946	0.3098	0.3471
	0.0800	0.9600	0.3211	0.3789	0.3109	0.3415
	0.0700	0.9514	0.2919	0.2987	0.3376	0.3170
	0.0600	0.9434	0.2561	0.2463	0.3376	0.2848
	0.0500	0.9294	0.2157	0.1934	0.3515	0.2495
	0.0400	0.9057	0.1757	0.1492	0.3878	0.2155
	0.0300	0.8880	0.1489	0.1265	0.3985	0.1920
	0.0200	0.8622	0.1204	0.1048	0.4145	0.1673
	0.0100	0.8587	0.1165	0.1021	0.4145	0.1638
	0.0000	0.0334	0.0000	0.0334	1.0000	0.0646
Baseline Attribute	0.1400	0.9688	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9688	0.0000	0.0000	0.0000	0.0000
	0.1200	0.9688	0.0000	0.0000	0.0000	0.0000
	0.1100	0.9661	0.2276	0.3990	0.1739	0.2422
	0.1000	0.9596	0.2159	0.2874	0.2002	0.2360
	0.0900	0.9527	0.1946	0.2255	0.2128	0.2190
	0.0800	0.9396	0.1511	0.1572	0.2151	0.1816
	0.0700	0.9369	0.1795	0.1728	0.2700	0.2107
	0.0600	0.9034	0.1400	0.1221	0.3387	0.1794
	0.0500	0.8701	0.1169	0.1007	0.3993	0.1609
	0.0400	0.7976	0.0679	0.0683	0.4348	0.1181
	0.0300	0.6621	0.0409	0.0522	0.5732	0.0957
	0.0200	0.6577	0.0402	0.0518	0.5767	0.0951
	0.0100	0.6498	0.0388	0.0511	0.5824	0.0939
	0.0000	0.0312	0.0000	0.0312	1.0000	0.0605

Table A.28.: Performance of the model with different pre-processing configurations for anomaly classification. The threshold is set by a grid search. The model is trained on H1000 with 30% sensor and 30% flow anomalies. The evaluation data contain 30% flow and 30% sensor anomalies.

Model	Threshold	Accuracy	Kappa	Precision	Recall	F_1
Normalized RNN	0.1400	0.9523	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9490	0.0912	0.3226	0.0630	0.1054
	0.1200	0.9499	0.3445	0.4630	0.3073	0.3694
	0.1100	0.9469	0.3387	0.4250	0.3212	0.3659
	0.1000	0.9447	0.3314	0.4016	0.3262	0.3600
	0.0900	0.9445	0.3751	0.4146	0.3942	0.4041
	0.0800	0.9426	0.3658	0.3977	0.3942	0.3960
	0.0700	0.9406	0.3567	0.3817	0.3942	0.3879
	0.0600	0.9386	0.3476	0.3665	0.3942	0.3799
	0.0500	0.9367	0.3409	0.3539	0.3967	0.3741
	0.0400	0.9350	0.3346	0.3435	0.3980	0.3687
	0.0300	0.8248	0.1367	0.1285	0.4622	0.2012
	0.0200	0.8213	0.1330	0.1260	0.4622	0.1980
	0.0100	0.8144	0.1277	0.1222	0.4673	0.1937
	0.0000	0.0477	0.0000	0.0477	1.0000	0.0911
Normalized CNN	0.1400	0.9476	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9408	0.1039	0.2773	0.0808	0.1251
	0.1200	0.9404	0.2846	0.3962	0.2605	0.3143
	0.1100	0.9371	0.2807	0.3664	0.2730	0.3129
	0.1000	0.9344	0.2750	0.3450	0.2799	0.3090
	0.0900	0.9343	0.3219	0.3666	0.3470	0.3565
	0.0800	0.9321	0.3132	0.3510	0.3470	0.3490
	0.0700	0.9308	0.3094	0.3426	0.3493	0.3459
	0.0600	0.9286	0.3019	0.3294	0.3504	0.3396
	0.0500	0.9266	0.2976	0.3197	0.3549	0.3364
	0.0400	0.9244	0.2909	0.3087	0.3561	0.3307
	0.0300	0.8160	0.1212	0.1248	0.4175	0.1921
	0.0200	0.8131	0.1188	0.1230	0.4187	0.1902
	0.0100	0.8056	0.1133	0.1190	0.4232	0.1858
	0.0000	0.0524	0.0000	0.0524	1.0000	0.0996
	0.1400	0.9479	0.0000	0.0000	0.0000	0.0000

Kalman Normalized RNN	0.1300	0.9490	0.1615	0.5549	0.1041	0.1753
	0.1200	0.9459	0.3298	0.4692	0.2872	0.3563
	0.1100	0.9423	0.3171	0.4231	0.2929	0.3462
	0.1000	0.9405	0.3115	0.4041	0.2963	0.3419
	0.0900	0.9402	0.3485	0.4132	0.3513	0.3797
	0.0800	0.9388	0.3429	0.4005	0.3524	0.3749
	0.0700	0.9369	0.3348	0.3850	0.3524	0.3680
	0.0600	0.9352	0.3286	0.3723	0.3535	0.3627
	0.0500	0.9328	0.3187	0.3548	0.3535	0.3542
	0.0400	0.9306	0.3123	0.3414	0.3570	0.3490
	0.0300	0.8237	0.1329	0.1322	0.4279	0.2019
	0.0200	0.8195	0.1284	0.1290	0.4279	0.1982
	0.0100	0.8132	0.1227	0.1249	0.4302	0.1936
	0.0000	0.0521	0.0000	0.0521	1.0000	0.0991
Kalman Normalized CNN	0.1400	0.9518	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9496	0.2279	0.4416	0.1733	0.2489
	0.1200	0.9447	0.2771	0.3867	0.2512	0.3046
	0.1100	0.9419	0.2935	0.3681	0.2884	0.3234
	0.1000	0.9402	0.3033	0.3605	0.3119	0.3344
	0.0900	0.9380	0.3134	0.3517	0.3403	0.3459
	0.0800	0.9356	0.3036	0.3345	0.3403	0.3374
	0.0700	0.9342	0.3011	0.3267	0.3453	0.3357
	0.0600	0.9323	0.2938	0.3149	0.3453	0.3294
	0.0500	0.8753	0.1680	0.1600	0.3738	0.2241
	0.0400	0.8726	0.1648	0.1570	0.3762	0.2216
	0.0300	0.8699	0.1616	0.1541	0.3787	0.2190
	0.0200	0.8648	0.1545	0.1481	0.3800	0.2131
	0.0100	0.8052	0.1089	0.1113	0.4356	0.1773
	0.0000	0.0482	0.0000	0.0482	1.0000	0.0919
Baseline Sensor	0.1400	0.9486	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9486	0.0000	0.0000	0.0000	0.0000
	0.1200	0.9495	0.1175	0.5688	0.0719	0.1277
	0.1100	0.9413	0.1313	0.2970	0.1044	0.1545
	0.1000	0.9383	0.1340	0.2676	0.1148	0.1607
	0.0900	0.9163	0.1348	0.1804	0.1775	0.1789
	0.0800	0.8939	0.1067	0.1358	0.1984	0.1612
	0.0700	0.8880	0.1324	0.1501	0.2529	0.1884

	0.0600	0.7715	0.0757	0.0943	0.4002	0.1526
	0.0500	0.6035	0.0432	0.0736	0.5789	0.1305
	0.0400	0.4640	0.0164	0.0595	0.6369	0.1089
	0.0300	0.4624	0.0162	0.0594	0.6381	0.1088
	0.0200	0.4403	0.0157	0.0592	0.6636	0.1087
	0.0100	0.4040	0.0132	0.0579	0.6937	0.1069
	0.0000	0.0514	0.0000	0.0514	1.0000	0.0978
Baseline ID	0.1400	0.9519	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9495	-0.0025	0.0233	0.0012	0.0024
	0.1200	0.9515	0.2518	0.4872	0.1886	0.2719
	0.1100	0.9506	0.2670	0.4693	0.2084	0.2887
	0.1000	0.9488	0.2716	0.4358	0.2233	0.2953
	0.0900	0.9409	0.3142	0.3692	0.3238	0.3450
	0.0800	0.9396	0.3085	0.3580	0.3238	0.3401
	0.0700	0.9372	0.2985	0.3394	0.3238	0.3314
	0.0600	0.9260	0.2676	0.2787	0.3400	0.3063
	0.0500	0.9169	0.2426	0.2432	0.3449	0.2853
	0.0400	0.8916	0.1961	0.1858	0.3710	0.2476
	0.0300	0.7841	0.0979	0.1034	0.4553	0.1686
	0.0200	0.7579	0.0870	0.0961	0.4801	0.1601
	0.0100	0.7052	0.0681	0.0844	0.5211	0.1452
	0.0000	0.0481	0.0000	0.0481	1.0000	0.0917
Baseline Attribute	0.1400	0.9516	0.0000	0.0000	0.0000	0.0000
	0.1300	0.9515	0.0815	0.4938	0.0493	0.0896
	0.1200	0.9371	0.1486	0.2431	0.1416	0.1790
	0.1100	0.9323	0.1808	0.2457	0.1921	0.2156
	0.1000	0.9145	0.1515	0.1802	0.2155	0.1963
	0.0900	0.8969	0.1339	0.1504	0.2426	0.1857
	0.0800	0.8919	0.1808	0.1774	0.3387	0.2329
	0.0700	0.8827	0.1668	0.1633	0.3448	0.2216
	0.0600	0.8031	0.0899	0.1007	0.3867	0.1598
	0.0500	0.7511	0.0741	0.0892	0.4495	0.1489
	0.0400	0.7464	0.0712	0.0875	0.4495	0.1465
	0.0300	0.7040	0.0587	0.0798	0.4852	0.1370
	0.0200	0.5643	0.0246	0.0609	0.5542	0.1097
	0.0100	0.5531	0.0235	0.0603	0.5640	0.1089
	0.0000	0.0484	0.0000	0.0484	1.0000	0.0924