Process Mining of Automation Services with Long Short-Term Memory Neural Networks

Karina Karapetyan

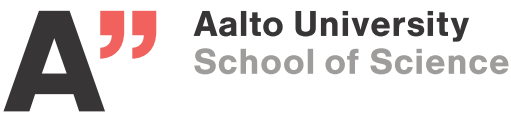
School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 15.12.2018

Thesis supervisor:

Assistant Professor Alex Jung



AALTO UNIVERSITY
SCHOOL OF SCIENCE

ABSTRACT OF THE
MASTER'S THESIS

Author: Karina Karapetyan

Title: Process Mining of Automation Services with Long Short-Term Memory Neural Networks

Date: 15.12.2018 Language: English Number of pages: 6+83

Department of Computer Science

Professorship: Big Data and Large-Scale Computing

Supervisor and advisor: Assistant Professor Alex Jung

Deep learning provides a wide scope of valuable tools for operating with and analysing various kinds of data sources. To optimise the workload of employees, Posti Group Oyj, the foremost organisation in the sphere of postal and logistics maintenance in Finland, was applied as a case study.

The primary target of the research was to elaborate a supervised machine learning model for the classification of eight-hour work shifts of mail carriers and the prediction of verification states for the inbound work shifts. Convolutional long short-term memory neural network was deployed as a baseline model since each work shift represents a sequence of postal tasks. The CNN LSTM network with the best-observed performance was deduced through carrying out the chain of the experiments, devoted to the network depth analysis and hyperparameter tuning procedure. The performance of the implemented models was assessed regarding a broad spectrum of the evaluation metrics, including classification accuracy, precision and recall measures, F-score, the area under the receiver operating characteristic curve score, and binary cross-entropy loss, utilising novel hyperparameter optimisation tool, called Talos. Furthermore, developed convolutional LSTM network substantially outperformed traditional machine learning approaches such as support vector machine, logistic regression, random decision forest, gradient boosting decision tree, k-nearest neighbours, and adaptive boosting classifiers. The devised CNN LSTM network will be integrated into a software as a service application for the prediction of the verification statuses of novel work shifts.

Moreover, supplementary analyses were conducted to determine incoming postal task and its timestamp within a work shift and predict potential postal jobs suffixes. Like in the previous research, LSTM network was maintained as a baseline algorithm. In both studies, the elaborated models obtained essentially better performance, compared to state-of-the-art algorithms.

Keywords: machine learning, deep learning, process mining, recurrent neural networks, long short-term neural networks, convolutional neural networks, network depth analysis, hyperparameter tuning, optimisation

Preface

This thesis was carried out for Posti Group Oyj. I would like to thank my company advisor, Riku Tapper, for his support throughout the thesis work and all the colleagues at Posti Group Oyj that have assisted me in exploring the machine learning field. Furthermore, I would like to thank and express the gratitude to my supervisor Alex Jung at Aalto University. Finally, I wish to thank my parents, Tengiz and Tatiana Karapetyan, for their love and encouragement.

Karina Karapetyan, December 2018

Contents

Symbols and abbreviations

Symbols

$\#_n(e)$	Value of attribute n of event e
ξ	Event universe
$\mathcal{R}_s$	Set of feasible state representations
$\mathcal{R}_e$	Set of feasible event representations
$\sigma_c = \langle e_1, e_2, \ldots, e_{\lvert\sigma\rvert} \rangle \; \forall \; 1 \leq i \leq \lvert\sigma\rvert$	Exemplary of trace
$\sum$	Set of all feasible (partial) traces
$f^{\text{state}} \in \xi \to \mathcal{R}_s$	State representation function
$f^{\text{event}} \in \xi \to \mathcal{R}_e$	Event representation function
η	Learning rate
$\mathcal{B}$	Batch size
$\phi(\cdot)$	Activation function

Operators

$\frac{\mathrm{d}}{\mathrm{d}t}$	Derivative with respect to variable t
$\frac{\partial}{\partial t}$	Partial derivative with respect to variable t
$\sum_i$	Sum over index i
$\mathbf{A} \cdot \mathbf{B}$	Dot product of vectors $\mathbf{A}$ and $\mathbf{B}$
$\sigma \uparrow X$	Projection of σ onto subset X
$\sigma_1 \oplus \sigma_2$	Merging sequence σ_2 to sequence σ_1

Abbreviations

ACC	Accuracy
AdaBoost	Adaptive boosting
AN	Attribute name
ANN	Artificial neural network
AUC	Area under receiver operating characteristic curve
CNN	Convolutional neural network
ELU	Exponential linear unit
GBDT	Gradient boosting decision tree
GRU	Gated recurrent unit
k-NN	k-nearest neighbours algorithm
LR	Logistic regression
LSTM	Long short-term memory network
MAE	Mean absolute error
MSE	Mean squared error
NHN	Number of hidden neurons
NHL	Number of hidden layers
NLP	Natural language processing
PAIS	Process-Aware Information Systems
PE	Precision
PPM	Probabilistic process model
RBF	Radial basis function
RE	Recall
ReLU	Rectified linear unit
RF	Random forest
RMSE	Root mean squared error
RNN	Recurrent neural network
ROC	Receiver operating characteristic curve
SaaS	Software as a Service
SVM	Support vector machine

1 Introduction

An escalating number of companies is deploying Process-Aware Information Systems (PAIS) [175] to handle their business workflow. Nowadays, these intelligence systems store all execution traces, or event logs, that contain the data regarding the states of implemented activities. Usually, logs indicate not only the carried out tasks but also the supplementary data attributes, depicting the process of task accomplishment. The retrieval of beneficial information from these tremendous data resources is the primary challenge of data mining. Consequently, process mining became an individual branch of the data mining field [173, 174]. Process mining anticipates that the maintained data is particularly related to the fulfilment of the business procedures. Methods of process mining are separated into several categories such as an elaboration of the models, e.g., control-flow, an evaluation of the conformance state of a certain log in regards with a derived process model (conformance verification), and an expansion of present models via utilisation of complementary data [134]. Moreover, except performing traditional operational activities, an event log can be used for an establishment of predictive models, capable of determining future features of the incomplete cases.

Gauging predictions is one of the most difficult problems in this field. The literature introduces a wide scope of researches targeting at the enhancement of the business methods and providing maintenance for their implementation. B. van Dongen, R. Crooy and W. van der Aalst describe a prediction model, employing all data records from an event log with an eye to realise a non-parametric regression and predict "the remaining cycle time" of the process, in [179]. The article "A Markov prediction model for data-driven semi-structured business processes" [95] portrays an instance-distinct probabilistic process model (PPM) where transition probabilities are adjusted in regards with the matching semi-structured business activity samples. PPM is considered an important tool for predicting the likelihood of diverse incoming activities in the executing processes. The paper "PEEK - An LSTM Recurrent Network for Motion Classification from Sparse Data" [45] conducts a study to determine the correlation between the perceived sparse data, gathered via a couple of portable measurement unit sensors, and the overall classification of body movements. This publication proposes a novel topology for a recurrent neural network (RNN), based on the concept of units in the long short-term memory net (LSTM), able of labelling motion sequences of arbitrary lengths.

Existing predictive process tracking methods are generally devised for certain types of problems, and, thus, can not be applied in other cases. In addition, accuracy significantly varies in regards to an input data set and a time point for which the prediction is estimated. One method can provide better performance on a particular event log, comparing to other machine learning approaches, while underperform on another set of event logs at the same prediction data example [40]. Frequently, a diversity of deep learning techniques is investigated along with carrying out hyperparameter tuning to achieve sufficient performance on the data set [113].

Recurrent neural networks with a long short-term memory architecture [81] have been acknowledged for deriving consistently high accuracy in a variety of sequence

modeling problems, e.g., speech and text recognition [87, 138, 152, 158], information retrieval [100], genome analysis [143, 181], etc.
Motivation: Posti Group Oyj, the foremost corporation of postal and logistics maintenance in Finland [83], yields an extensive amount of information every day concerning planning and execution of the work shifts for the employees across the country (each work shift represents a sequence of postal tasks), a broad range of sundry delivery routes, sorting schemes, etc. Inspired by the valuable key insights that can be extracted from this data source as well as concepts and approaches of process mining, this paper scrutinises the following issues:

1. How neural networks can be deployed in the predictive process monitoring problems? In particular, we explore the behaviour and effectiveness of long short-term memory nets, owing to their superior results on sequence classification.

2. What kind of results are gained by LSTM networks for a spectrum of prediction problems, data sets and prediction points? To review these inquiries, the paper describes LSTM networks for predicting:

   - the verification status of an incoming work shift,
   - the subsequent task in a work shift and its timestamp,
   - the chain of consecutive events in a work shift until its completion.

   Moreover, we compare the observed performance of the developed models with the results achieved by the state-of-the-art machine learning techniques.

The data, provided for the research, is stored in the Posti Timestamp Verification data set (PTVD log) that represents an event log, containing all of the planned and later performed work shifts of Posti Group's mail carriers during 2017. Every work shift in the data set denotes a sequence of tasks with the beginning and end timestamps where each task is designated with own identification number.
Scope: The primary goal of the Master's thesis is to deduce an LSTM-based neural network for the binary classification of the input work shifts via prediction of their confirmation states such as "OK" or "NOTOK" labels. With an eye to achieve the best performance of the baseline LSTM model, the experiments on the network architecture and tuning of diverse hyperparameters should be conducted. The performance of the developed models is assessed according to the well-known evaluation metrics, including classification accuracy, precision, recall, F-score and the area under the receiver operating characteristic (ROC) curve (AUC). Furthermore, the LSTM model with the best-observed performance is compared with 6 traditional machine learning approaches, including support vector machine (SVM), logistic regression (LR), random decision forest (RF), gradient boosting decision tree (GBDT), k-nearest neighbours (k-NN), and adaptive boosting (AdaBoost) classifiers. Currently, the verification process is manually handled by supervisors at the local departments every day and, consequently, it is overly time-consuming. Thus, applying a tailor-made LSTM network can essentially diminish the workload for the supervisors.
Moreover, in order to handle the additional studies on Posti Timestamp Verification

data set, the next research problems should be explored such as determination of the next activity and its timestamp and prediction of the postal suffixes within work shifts. Likewise the principal goal of the research, LSTM network is deployed as a baseline model with a mean absolute error (MAE) used as an evaluation measure.
In order to create a perfect machine learning model with the highest possible performance, the process of hyperparameter tuning and selection of the network structure must be investigated carefully. The recently published research article "Hyperparameter optimisation with Keras" [91] proposed a neoteric and easy-to-learn hyperparameter optimisation and tuning tool for machine learning algorithms, named Talos. It assists in arranging the hyperparameter settings for the designed Keras models and resolves the hyperparameter tuning problem.
Hence, the key objectives of the Master's thesis can be summarised as:

1. interactive data exploration and visualization,
2. learning from the past - training machine learning models based on daily, weekly and monthly collections of work shifts,
3. optimisation of the supervisors' workload via forecasting confirmation statuses of the work shifts by the devised LSTM model,
4. prognostication of the consequent postal task and its timestamp within a work shift,
5. prediction of the postal suffixes in work shifts,
6. scrutinising novel hyperparameter optimisation tool for Keras models, named Talos.

The remainder of this paper is structured as follows: Section 2 provides an overview of the latterly conducted studies concerning diverse prediction tasks in terms of process mining. Section 3 depicts the fundamental concepts and notations in process mining and machine learning fields, applied in the paper. Further, Section 4 describes the case study, Posti Group Oyj. Section 5 demonstrates the realisation of the research methods and indicates the obtained experimental results as well as the reasoning behind them. Finally, Section 6 shortly recaps the introduced content, concludes the paper and sketches future work routes.

2 Related Work

The following section provides an overview on existing concepts and methodologies, applied in the predictive process tracking for three types of the forecast problems: time-dependent classifications, supervised sequential data labelling tasks and estimate of the case suffix and its features.

2.1 Prediction of time-related characteristics

A wide scope of the researchers has scrutinised carefully the issue of forecasting time boundings, including delays and deadline violations, in the inter-organisational business workflow procedures. The studies [48, 56, 175] devise methods for predicting deadline violations. The research [117] develops a dynamic machine learning model for performing a thorough evaluation of the estimated time of the arrival of the bus at the indicated bus stop, employing global positioning system data. The paper "Queue Mining – Predicting Delays in Service Processes" [150] illustrates the eponymous techniques, elaborated for providing a solution for an operational problem of online delay determination to yield accurate online predictions of the case delays, based on the contemporary event logs.
Moreover, the remaining cycle time of the fulfilled cases can be defined. B. van Dongen, R. Crooy and W. van der Aalst demonstrate in their work [179] the manifold of the techniques for computing the remaining cycle time via application of the non-parametric regression on the data, stored in the event log. In the publication [177], W. van der Aalst, H. Schonenberg and M. Song propose a configurable machine learning approach to establish a process model from the transition system, enhance it with the time data, learned from the earlier samples, and apply the derived model to a prediction, e.g., the completion time. Finally, A. Rogge-Solti and M. Weske introduce a method for the forecast of the remaining service execution time by utilising stochastic Petri nets, capable of detecting arbitrary duration distributions, in their publication [140].
This paper does not address the problem of evaluating the remaining cycle times, even though the contrived approaches could be dilated to this extent.

2.2 Supervised sequential data labelling problems

In the machine learning, the definition "sequence labelling" comprises a variety of tasks with the data sequences, arranged in the form of a series of the discrete classes [64]. Sequence classification has a broad spectrum of applications such as speech and text recognition [87, 138, 152, 158], information retrieval [100], part-of-speech tagging [115, 160], genome analysis [143, 181], finance [151] and anomaly detection [189].
Supervised sequence labelling represents particularly those occurrences where a set of hand-decrypted continuities is given as an input for the algorithm's training stage. The principal difference between this type of the problems and the conventional scope of the supervised pattern classifications is the fact that the distinct instances cannot

be considered to be autonomous. Instead, inputs and categories produce highly correlated series. For instance, a voice signal in the speech recognition is generated by a ceaseless movement of the vocal tract, whereas the classes (word series) are mutually restrained in terms of the syntax and grammar rules. Furthermore, the adjustment among inputs and classes is frequently unknown. Thus, the employment of the algorithm is crucial to discover the position and magnitude of the output labels [64].

One of the goals of the Master's thesis is to actualise the prediction on the supervisor's conformance of the work shifts, denoted by a distinct data field in the PTVD log. The determination of a work shift "assessment" is based on the state of the implementation of a time-associated sequence of the postal tasks within a work shift. The first step in the elaboration of the solution, regarding the initial problem settings, is a selection of the appropriate machine learning model. Since Posti Timestamp Verification data set is maintained in a form of an event log that contains all of the planned and later completed work shifts of Posti Group Oyj's mail carriers during 2017, the recurrent neural network is selected as an initial research approach, by the virtue of RNNs operating as a powerful tool for treating sequential data. Eventually, LSTM architecture is fetched as a baseline method, since this type of the network contains "memory cells" which can retain the data for long periods of time [78].

LSTM neural networks have a wide range of employment, due to the diversity of the sequential data. The research article "Audio Scene Classification with Deep Recurrent Neural Networks" by H. Phan et al. [132] proposes an efficacious technique for audio scene categorisation, using deep learning networks, in particular, RNNs. To begin with, an audio signal is converted into a set of the top-ranking class trees of embedding features. Further, it is split into multifold subsequences, applied for training deep GRU-based recurrent neural networks with an eye to realising the sequence-to-label categorisation. The terminal predicted class for the whole sequence is eventually inferred from the accumulation of the outputs for the subsequent labels. The report "Classifying business process execution traces with LSTM" by I. Teinemaa [169] scrutinises the relevance of the LSTM neural nets for labelling among normal and erratic cases in the business operations, employing the Business Process Intelligence Challenge (BIC) data set from 2011. Each separate case embodies a sequence of accomplished activities. Another area of adaptation of the LSTM networks is protein analysis for identifying protein functional label and 3D architecture [143]. In the end, the research article "Combination of Convolutional and Recurrent Neural Network for Sentiment Analysis of Short Texts" [183] demonstrates the advantage of articulated convolutional neural networks (CNN) and recurrent neural nets (RNN) structures in sentiment analysis with regards to other machine learning models. The primary benefit of CNN architecture is the production of the coarse-grained local features, whereas the RNN's asset is the long-memory dependencies. Thus, they are deployed in the conjunction during the classification maintenance.

The aim of these studies is to predict global classes of the cases in regards to a sequence of activities, events, attributes, employed in each specific instance. F. Maggi et al. [113] introduce a framework to determine the output of a case (normal or erratic) with respect to the activity series, carried out in the indicated case, and

data parameters of the latter performed activity within the case. The following approach establishes a classifier on-the-fly, i.e., a random forest model, in regards to the historical data instances, analogical to the unfinished series of the completed case. Other methods derive a set of classifiers offline. For instance, the article "Complex Symbolic Sequence Encodings for Predictive Monitoring of Business Processes" [104] depicts a technique to develop a single classifier for each feasible forecast data instance to acquire the output after every subsequent event. The research approach, described in [40], employs the clustering model to collect cognate prefixes of the historical sequences and elaborates a single classifier for each cluster.
For every mentioned above method, the retrieval of a feature vector from a prefix of the running case is needed. M. de Leoni, W. van der Aalst and M. Dees presented in [36] an approach for classification of the feasible methods to withdraw these feature vectors.

2.3 Prediction of feasible set of future events

In order to conduct the supplementary studies on the Posti Timestamp Verification data set, the following research problems were explored:

- prediction of an incoming activity and the corresponding timestamp,
- and designation of postal suffixes in the course of work shifts.

In the paper "Comprehensible Predictive Models for Business Processes" [19], authors propose a next activity forecast method which fits a probabilistic model on a data set of the historical behaviour to discover the continuation of the executing operation instance. J. Evermann, J. Rehse and P. Fettke depict a deep learning approach, peculiarly recurrent neural networks, to the matter of predicting succeeding event in the case [49].
This research scrutinises the concepts, indicated in the articles, "Predictive Business Process Monitoring with LSTM Neural Networks" by N. Tax et al. [167] and "LSTM Networks for Data-Aware Remaining Time Prediction of Business Process Instances" by N. Navarin et al. [128], and conducts matching experiments on the PTVD log. The publication [167] examines the architecture of LSTM neural net as a technique to establish coherently accurate models for a broad compass of predictive process tracking issues. The paper illustrates the outstanding performance of LSTM nets on determining the next activity and its timestamp within the executed case as well as forecasting the suffix of the activities until the end of the operation is reached. Besides, the results of the experiments are compared with the research methods, devised by [19] and [49]. The publication [128] describes the machine learning approach, based on deep LSTMs, that is able to handle an arbitrary data, associated with a single event, to generate an accurate estimation of the next activity that a process instance is going to make.
The following studies conduct the forecasting of the incoming activities in the procedure samples through handling diverse machine learning techniques as baseline methods, excluding recurrent neural networks and, notably, long short-term memory

nets. The conference paper "Process Prediction in Noisy Data Sets: A Case Study in a Dutch Hospital" [155] illustrates an application of the process mining for estimating the money flow within the Dutch hospital. To start with, the stages of the healthcare methods are determined in relation to their influence on the choice of the care product. Further, the research explores the technique for assessing the duration of the care procedure and the product price. An event log is embodied in a form of a graph that denominates the operation sequences of the log as edges. The goal of the objective function is to identify the shortest path that obtains the maximum product of the edge weights. The authors have selected Floyd-Warshall's algorithm as a base method, since it does not calculate a sum of the edge weights as the metrics of length, but the product of edge weights. M. Polato et al. [134] have enhanced this research approach by deriving a transition system from an event log and assigning the transition probabilities to the edges.

3 Background

This section depicts definitions and research approaches, applied in the later parts of this paper.

3.1 Process mining

Information systems serve as a basic element in the architecture of the majority of organisations. Without them, companies would not be able to provide products or services, obtain resources, pay off with suppliers or submit tax reports. These systems observe valuable information regarding the process execution through the event logs, including activities, initiators, timestamps and case information, in order to extract and further analyse the gained data. Owing to discovered beneficial key insights, organisations can monitor and enhance their business processes [174]. The term "process mining" embodies this concept.
In 2002, W. van der Aalst, A. Weijters, and L. Măruşter introduced the following definition of the process mining, indicating it as a "method for distilling a structured process description from a set of real executions" [178]. Hence, the goal is to re-establish a formerly unknown procedure from a log, generated via the implementation of instances, and maintain outcomes in a structured modelling language, e.g., Petri nets or UML diagrams [38].
Due to the considerable progress in the field throughout the last decades, the above denotation became confined to cover the variety of present research activities in process mining [180]. In the book "Process discovery: capturing the invisible", W. van der Aalst indicated that "the idea of process mining is to discover, monitor and improve real processes (i.e. not assumed processes) by extracting knowledge from event logs" [172].
Starting from all diverse kinds of recorded process data, process mining tries to automatically detect design and attributes of business processes that can be visualized in business process models. **Figure 1** demonstrates standard data, necessary to be stored in the event log. The relation between event log and process model is founded on several levels such as model, instance and event. The model and event layers are independent of each other since models are derived without being referred to the (raw) data in the information management systems. The following denotations concerning event logs can be deduced [116]:

- The instance level includes cases and activity instances that incorporate various processes and different activities to the events in the log file.
- At the model stage, a process can obtain an arbitrary amount of activities, whereas every activity pertains to a single process.
- An event log retains the information regarding the particular process that can involve several cases. However, every case refers to exactly one process.
- Every event in the log resides to a uniform case and is stored according to the order.

- Each event belongs to an activity instance, while every activity instance corresponds to a single activity and a unique case.

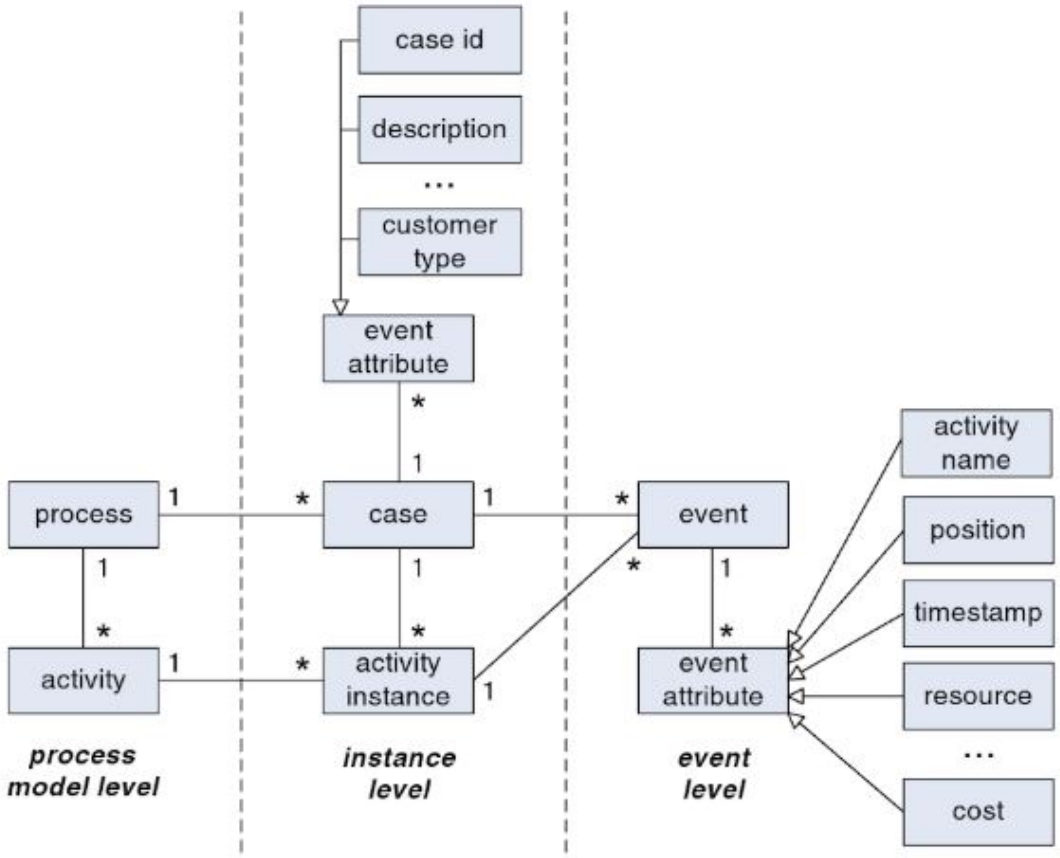


Figure 1: The illustration of exemplary data, required to be present in the event log, and the connection with a process model [116]

In addition, supplementary data regarding the events can be collected in the event logs. Process mining techniques employ event attributes such as the resource (person or device), performing or initiating the activity, timestamps, cost, and other various data properties, e.g., order dimension. The process mining includes three phases [8]:

1. Process discovery is applied by analysts to deduce a model from an event log. Moreover, specialists can develop an original process model manually. Essentially, the discovery technique takes as an input an event log and generates a model without utilising any preliminary data. Prominent data mining methods are deployed to explore processes, based on instance executions in the event logs.

2. After the previous step, scientists perform conformance verification to detect deflections among event log and original process model. This stage is assigned for checking if processes are executed and stored in log files, as well as they correspond to the model (and contrariwise).

3. Lastly, throughout model enhancement, experts employ data from logs to fix or evolve the model. For example, timestamps are engaged to provide timing information (anticipation and service times) for the model. The resulting reinforced process model contributes to the decision making. In other words, the concept is to propagate or refine an existing process model through maintaining information concerning the present processes, stored in the event log. Whereas the conformance verification evaluates the concurrence between model and reality, model development aims at modifying or elaborating the model. For example, applying timestamps in the event log can allow the model to identify bottlenecks, service degrees, capacity times, and frequencies.

Since manual modelling is time-consuming and dependent on skills and experience of domain experts, process mining provides such advantages as correctness (no errors), completeness (no deficient paths) and speed [55]. These designations infer that the process mining is not bounded by "mining" in the narrow meaning, e.g., process discovery, but it also involves onward tasks as compliance verification and enhancement of process models [172].
Process mining creates the bridge among the conventional model-based process researches, e.g., simulation and other business process management approaches, and data-oriented scrutinising methods such as machine learning and data mining. Process mining looks for the collation between the event data, for instance, observed behaviour, and process models (manually designed or established automatically). Process mining takes an event log as an input and produces the end-to-end process model [173].

3.2 Algebra of event logs, traces and sequences

Table 1 illustrates an exemplary information, extracted from an event log. Generally, process mining techniques consider event logs as well-structured data sources where every event of a business process is stored and corresponds to an activity of a distinct case. Methods in the process mining might require other supplementary data, e.g., initiator of activity, matching timestamp, etc. Nowadays, a majority of organisations is utilising software programs that monitor business process execution via event logs, i.e., databases, transaction journals, spreadsheets [134].

Table 1: Snippet of an event log with events, grouped by "Case ID" and ordered with respect to the accordant "Timestamps"

Case ID	Timestamp	Resource	Activity	Category	Cost
780644	13-07-2018:10.15	*David*	*A*	*Premium*	800
780644	14-07-2018:06.07	*Alice*	*C*	*Premium*	700
780644	14-07-2018:20.18	*Alice*	*F*	–	200
780645	12-07-2018:19.44	*David*	*B*	*Regular*	400
780645	13-07-2018:09.53	*Alice*	*F*	*Regular*	400
780645	14-07-2018:20.00	*Michael*	*G*	*Regular*	400

The essential concepts of process mining include such terms as event, trace (partial trace), event log, state and event log representation functions, elucidated below in the following section.

Definition 1. (Event, attribute) An event is denoted by a tuple $e = (a, c, t, s_1, \ldots, s_q)$ where $a \in \mathcal{A}$ is a process activity, related to the event, $c \in \mathcal{C}$ defines a case identification number, $t \in \mathcal{T}$ denominates event timestamp, and $s_1, \ldots, s_q$ is a list of supplementary attributes with $\forall\ 1 \leq i \leq q$, $s_i \in \mathcal{S}_i$. The projection functions can be devised for the event e [134]:

$$\pi_{\mathcal{A}}(e) = a$$
$$\pi_{\mathcal{C}}(e) = c$$
$$\pi_{\mathcal{T}}(e) = t$$
$$\pi_{\mathcal{S}_i}(e) = s_i\ \forall\ 1 \leq i \leq q$$

The denomination AN is used for the set of the attribute names. The event universe (the set of all feasible event identifiers) is determined as:

$$\xi = \mathcal{A} \times \mathcal{C} \times \mathcal{T} \times \mathcal{D}_1 \times \cdots \times \mathcal{D}_q$$

For any event $e \in \xi$ and attribute name $n \in AN$, $\#_n(e)$ designates the value of the attribute n of the event e. For instance, the timestamp of the event e is noted as $\#_{\text{timestamp}}(e)$. In case an event e does not contain an attribute n, then $\#_n(e) = \perp$ (undefined value) [70]. A standard series of attributes, applied for outlining events comprises [2]:

1. $\#_{\text{resource}}(e)$ - the resource, associated with the event e,
2. $\#_{\text{activity}}(e)$ - the activity, bounded to the event e,
3. $\#_{\text{timestamp}}(e)$ - the timestamp of the event e,
4. $\#_{\text{trans}}(e)$ - the transition pattern, tied to the event e (schedule, start, and complete).

Several conventions are established for the set of standard attributes, e.g., timestamps are allocated in the non-descending order in the event log [54]. Also, transition pattern attribute, $\#_{\text{trans}}(e)$, alludes to the life-cycle of activities (**Figure 2**), since in the majority of situations the performance of the activities requires time. Hence, the start or the completion of the activities can be concluded from the events [173].

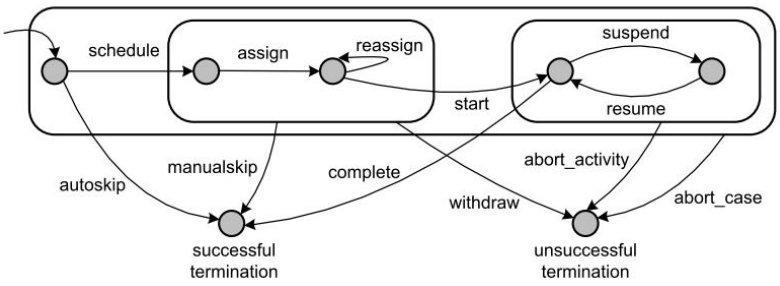


Figure 2: Standard transactional life-cycle model [173]

Another substantial concept in process mining is the term "sequence" that allows to render traces in the event log [102]. As the information systems are depicted in regards with the operational semantics, the behaviour is also simulated in terms of sequences.

Definition 2. (Sequence) For a defined set P, P^* is the set of all finite sequences across P. Each finite sequence across P with length n incarnates a mapping $\sigma \in \{1, \ldots, n\} \rightarrow P$ [11]. The following sequence is maintained as a string, e.g., $\sigma = \langle p_1, p_2, \ldots, p_n \rangle$ where $p_i = \sigma(i)$ for $1 \leq i \leq n$. The length of the sequence is implied as $|\sigma|$ and $|\sigma| = n$. The sequence with an element $p^{'}$, joined at the end, is denoted as $\sigma + p^{'} = \langle p_1, \ldots, p_n, p^{'} \rangle$. Likewise, a sequence σ_2 can be merged to a sequence σ_1 as $\sigma_1 \oplus \sigma_2$, resultant to a sequence with length $|\sigma_1| + |\sigma_2|$ [92].
The next statements regarding the sequence σ can be formulated:

- $hd^k(\sigma) = \langle p_1, p_2, \ldots, p_{k \min n} \rangle$ - a head of the sequence σ, involving the first k elements (if feasible). $hd^0(\sigma)$ represents an empty sequence, and $hd^k(\sigma) = \sigma$ for $k \geq n$. The set of prefixes of σ is determined as $\text{pref}(\sigma) = \{hd^k(\sigma) \mid 0 \leq k \leq n\}$ [170].

- $tl^k(\sigma) = \langle p_{(n-k+1) \max 1}, p_{k+2}, \ldots, p_k \rangle$ - a tail of the sequence σ, constituted of the latter k elements (if feasible). The empty set is denoted as $tl^0(\sigma)$, and $tl^k(\sigma) = \sigma$ for $k \geq n$ [176].

- The projection of sequence σ onto a subset $X \subseteq P$ is specified as $\sigma \uparrow X$. For instance, $\langle a, b, c, d, a, b, b \rangle \uparrow \{a, b\} = \langle a, b, a, b, b \rangle$ and $\langle c, b, b, b, b, c \rangle \uparrow \{c\} = \langle c, c \rangle$ [174].

- For any sequence $\sigma = \langle p_1, p_2, \ldots, p_n \rangle$ over P, $\partial_{\text{set}}(\sigma) = \{p_1, p_2, \ldots p_n\}$ and $\partial_{\text{multiset}}(\sigma) = [p_1, p_2, \ldots p_n]$. ∂_{set} transfers a sequence into a set, i.e., $\partial(\langle b, d, d, b, d \rangle) = \{b, d\}$. An element p of the sequence σ is designated as $p \in \sigma$ if and only if $p \in \partial_{\text{set}}(\sigma)$. $\partial_{\text{multiset}}$ transforms a sequence into a multi-set, i.e., $\partial_{\text{multiset}}(\langle b, d, a, d, a, b, d, b, d, d \rangle) = [a^2, b^3, d^5]$. $\partial_{\text{multiset}}(\sigma)$ is also named as Parikh vector of sequence σ [170]. These transitions provide an opportunity to handle sequences as sets or bags, when required [173].

Definition 3. (Trace, Partial Trace). Assume $\mathcal{C}$ is the case universe, redacting the set of all possible case identifiers. As events, cases possess attributes. For any case $c \in \mathcal{C}$ and attribute name $n \in AN$, $\#_n(c)$ stands for the value of attribute n for the case c. Every case has a particular mandatory attribute, trace, such that $\#_{\text{trace}}(c) \in \xi^*$ [110]. A trace designates a finite sequence of events [134]:

$$\begin{aligned} \sigma_c = \langle e_1, e_2, \ldots, e_{|\sigma_c|} \rangle \in \xi^* \ \forall \ 1 \leq i \leq |\sigma_c| \\ \pi_{\mathcal{C}}(e_i) = c \wedge \forall \ 1 \leq j \leq |\sigma_c| \\ \pi_{\mathcal{T}}(\sigma_c(j)) \leq \pi_{\mathcal{T}}(\sigma_c(j+1)) \end{aligned}$$

In other words, every event in the trace appears once, i.e., $\sigma(i) \neq \sigma(j)$ for $1 \leq i < j \leq |\sigma|$.
Whilst a trace refers to an utter process instance (initiated and terminated), a partial

trace depicts a process instance that is still under implementation and, thus, it has not been finished yet [111]. A partial trace of length m is denoted as $\sigma_c^m = hd^m(\sigma_c)$ for $m \in [1, |\sigma_c|] \subset \mathbb{N}$. The set of all feasible (partial) traces is defined as $\sum$ [161]. Across a trace $\sigma_c = \langle e_1, e_2, \dots e_{|\sigma_c|} \rangle$, the next projection functions can be determined:

- $\Pi_{\mathcal{A}}(\sigma_c) = \langle \pi_{\mathcal{A}}(e_1), \pi_{\mathcal{A}}(e_2), \dots, \pi_{\mathcal{A}}(e_\sigma) \rangle$,
- $\Pi_{\mathcal{T}}(\sigma_c) = \langle \pi_{\mathcal{T}}(e_1), \pi_{\mathcal{T}}(e_2), \dots, \pi_{\mathcal{T}}(e_\sigma) \rangle$,
- $\Pi_{\mathcal{S}_i}(\sigma_c) = \langle \pi_{\mathcal{S}_i}(e_1), \pi_{\mathcal{S}_i}(e_2), \dots, \pi_{\mathcal{S}_i}(e_\sigma) \rangle$ for all $1 \leq i \leq |S|$.

Definition 4. (Event log) An event log typifies a set of traces $L \subseteq \mathcal{C}$ where every event occurs at maximum once in the whole event log, e.g., for any $c_1, c_2 \in L$ such that $c_1 \neq c_2 : \partial_{\text{set}}(\hat{c}_1) \cap \partial_{\text{set}}(\hat{c}_2) = \emptyset$ [173].

Definition 5. (Transition System) A transition system is declared as a triplet $TS = (S, A, T)$ where S represents the set of states, $A \subseteq \mathcal{A}$ implies the set of attributes and $T \subseteq S \times A \times S$ defines the set of transitions. The set of initial states is denoted by $S^{\text{start}} \subseteq S$, and the set of complete states - $S^{\text{end}} \subseteq S$ [134].

A walk in a transition system conveys a sequence of transitions $\langle p_1, p_2, \dots, p_n \rangle$ such that [186]:

$$
\begin{aligned}
p_1 &= (s_1 \in S^{\text{start}}, e, s_1^{'}) \\
p_n &= (s_n, e, s_n^{'} \in S^{\text{end}}) \\
p_h &= (s_h, e, s_{h+1}) \ \forall \ 1 < h < n
\end{aligned}
$$

where $s \in S$ is an indicated state, and the set of attainable states from s is determined as $s = \{s^{'} \in S \mid \exists \ p \in T, \exists \ e \in E \text{ s.t. } p = (s, e, s^{'})\}$.

In order to form a transition model that maps every partial trace in the event log to a state, state and event representation functions are deployed.

Definition 6. (State representation function) Consider $\mathcal{R}_s$ is a set of feasible state representations, a state representation function $f^{\text{state}} \in \sum \rightarrow \mathcal{R}_s$ derives a function where an obtained (partial) trace σ results into some type of its form, i.e., sequences, sets, multisets across event attributes, etc. [130].

Definition 7. (Event representation function) Assume $\mathcal{R}_e$ is a set of feasible event representations, an event representation function $f^{\text{event}} \in \xi \rightarrow \mathcal{R}_e$ implicates a function where an indicated event e generates own form, i.e., projection functions across $e \in \xi$ such as $\pi_{\mathcal{A}}(e)$, $\pi_{\mathcal{T}}(e)$, etc. [37].

3.3 Recurrent neural networks: concept, design and applications

An incalculable amount of learning tasks requires operation with the information in the sequential format. Image descriptions, speech generations, music production, all demand that a model generates outputs that are sequenced. In other fields, such as time-series predictions, video processing, and music information extraction, model has to learn from inputs given in a form of sequences. Interactive issues, including

interpreting natural language, participating in a dialogue, and maintaining a robot, oftentimes require both facilities.

Recurrent neural networks (RNN) are models, founded on the principle of connectionism, that maintain the changeover of sequences through cycles in the net of nodes [108]. Recurrent neural networks have been a significant target for the research and elaboration since the 1990s, based on the established fundamental work in the late 1980s by Rumelhart, Hinton, and Williams [121]. In the contradistinction to the regular feed-forward networks, RNN preserves a state which can elicit data from an arbitrarily sized context window [42]. Even though recurrent neural nets are traditionally difficult to train and frequently involve millions of parameters, the latter progress in network architectures, optimisation approaches, and wide capabilities of parallel computation have contributed to the large-scale learning [108]. Bidirectional associative memory, Hopfield, Boltzmann machine, and recurrent backpropagation networks illustrate some examples of an application of recurrent neural nets [75].

Recurrent neural networks include high dimensional hidden states with non-linear behaviour [163]. The design of hidden states performs as the memory of the net, and the state of the current hidden layer is dependent on its preceding state [120]. This architecture allows the recurrent neural networks to retain, keep and handle antecedent compound signals for prolonged periods of time. Recurrent neural network maps an entry sequence to the output sequence at the present time point and predicts the sequence in the succeeding time point [144].

In a traditional neural network, the inputs, as well as outputs, are considered to be independent of each other. However, the following idea fails for the prediction of the incoming word in the sentence. Recurrent neural networks are named "recurrent" since they execute the similar task for each element of a sequence where the output is calculated, based on the previous computations. In addition, RNNs are considered to have "memory" that gathers information regarding previously implemented calculations. Theoretically, RNNs are capable of "memorizing" arbitrarily long sequences. In practice, recurrent neural networks are bounded to look back only a few steps behind. **Figure 3** demonstrates the process of unrolling (unfolding) a recurrent neural network into a complete net, deriving the network for the full sequence [99].

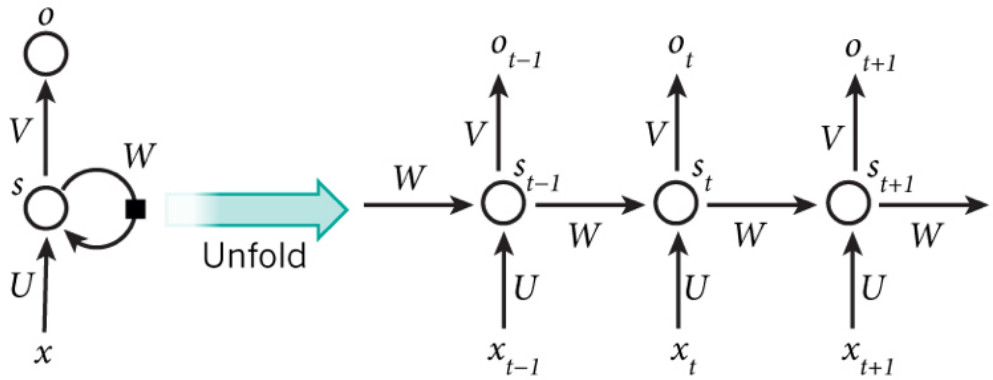


Figure 3: The process of unfolding a simple recurrent neural network [99]

The next equations form the computations, executed by a RNN [42]:

- x_t - the input at a time point t. For example, x_1 typifies a one-hot vector, associated with the second word in a sentence.

- s_t - the hidden state at a time point t, performing "memory" function of the network. Based on the preceding hidden state and the input at the present stage, s_t is estimated as [108]:

$$s_t = f(Ux_t + Ws_{t-1})$$

, where U and W are the weight vectors across novel inputs and the hidden state accordingly.
Commonly, the activation function f is non-linear, i.e., hyperbolic tangent or rectified linear unit (ReLU). s_{t-1}, required for the computation of the first hidden layer, is originally initiaised with all zeros.

- o_s - the output at a time point t. For instance, the predicted incoming word in the sentence is determined as a vector of probabilities over the vocabulary:

$$o_t = \text{softmax}(Vs_t)$$

Sequential RNNs are remarkably efficient in natural language processing (NLP). In the past years, the results of the language models have been substantially elaborated, owing to the recurrent neural networks, in comparison with the long-mounted state-of-the-art baselines [90]. Due to the performance boost, gained by the recurrent neural networks, they have been widely utilised in diverse conditional language modelling issues as machine interpretation [26], picture heading generation [187], and handwriting recognition [64].

3.4 Long short-term memory networks for sequence classification

As discussed in the previous section, the significant advantage of recurrent neural networks is their capability to maintain contextual data, when mapping input and output sequences. For regular RNN structures, the scope of context that can be in fact employed is rather bounded. The issue is that the impact of the defined input on the hidden layer, and, consequently, on the output of the net, either diminishes or explodes exponentially as it implements cycles across the net's recurrent conjunctions. This effect is depicted in the literature as the "vanishing gradient descent problem" [13, 77, 79]. **Figure 4** displays this issue schematically.
The substantial number of attempts was executed in the 1990s to resolve the vanishing gradient issue for the recurrent neural networks [78]. The potential solutions included non-gradient founded training methods, i.e., simulated annealing and discrete error propagation [13], thoroughly injected time delays [74, 96, 106], or time variables [124], and hierarchical sequence shrinkage [147]. Thus, long short-term memory network

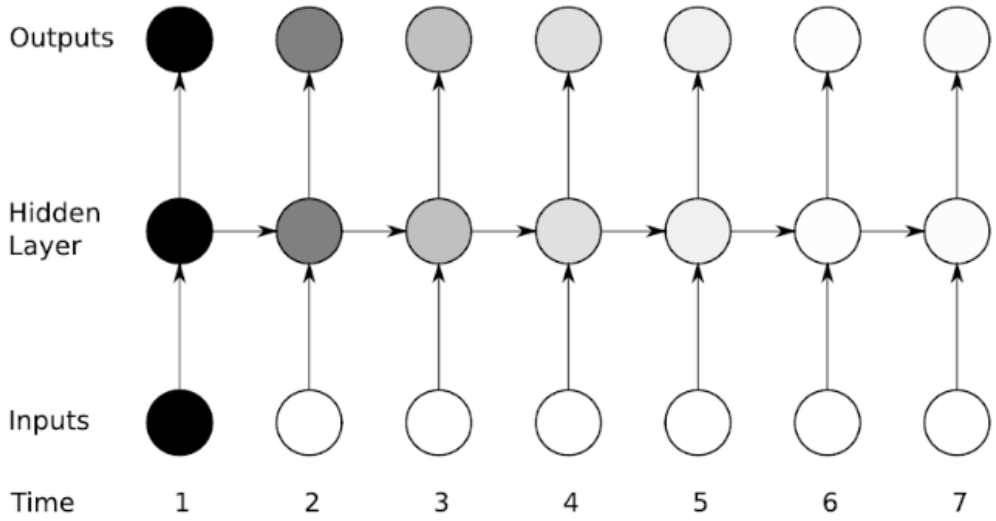


Figure 4: The vanishing gradient problem in the recurrent neural networks [94]. The colour gradient of the nodes in the unrolled net implies their responsiveness to the inputs at the first time point. The colour intensity indicates the degree of responsiveness of the node to the inputs. The sensitivity diminishes during the time as the novel inputs overwrite the activation of the hidden layer, and the net 'forgets' the initial inputs.

(LSTM) was proposed in 1997 by S. Hochreiter and J. Schmidhuber to overcome the major problem of recurrent nets via adding gates and memory cells in the hidden layer to maintain the information flow [78].
The LSTM structure includes a set of recurrently linked sub-networks, alias memory units. The following blocks can be interpreted as a distinctive variant of the memory circuit in a digital computer [64]. Every unit involves at least one self-connected memory element and three multiplicative modules, including input, output and forget gates, which maintain enduring analogues of basic operations for the cells such as read, write and reset [46]. **Figure 5** displays a LSTM memory block with one cell.
LSTM network has a similar structure with a regular RNN besides the summation units in the hidden layer that are substituted with memory blocks (**Figure 6**). The cell state is changed by the outputs of input, output and forget gates. LSTM cell contains four internal hidden layers that learn suitable weights, necessary to properly forget and update (elements of) the cell state [41]. The outputs are evaluated through this set of equations [67]:

$$
\begin{aligned}
i_t &= \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \\
f_t &= \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \\
c_t &= f_t c_{t-1} + i_t \tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \\
o_t &= \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \\
h_t &= o_t \cdot \tanh(c_t)
\end{aligned}
$$

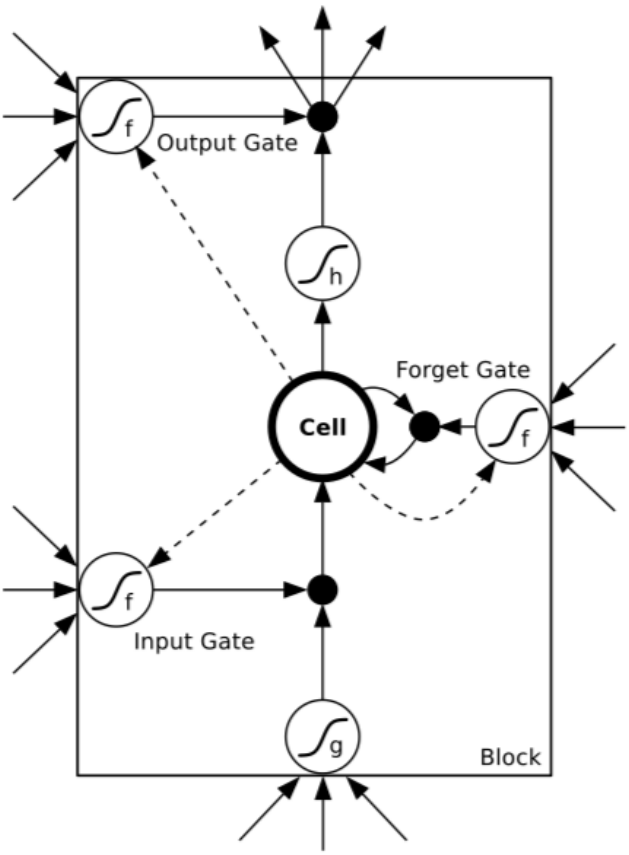


Figure 5: LSTM memory unit with a single cell [64]. The input, output and forget gates represent non-linear summation units which gather activations from both inside and outside of the memory block and handle the process of activation of the cell through the performance of multiplications, denoted as black circles [109]. The input and output gates multiply input and output of the cell, whereas the forget gate executes multiplication on the preceding state of the cell [15]. The activation function is not deployed inside of the cell. The gate activation function f is oftentimes the logistic sigmoid. Hence, the gate activations can obtain values among 0 (gate is closed) and 1 (gate is opened) [84]. The cell input and output activations functions, g and h respectively, are usually hyperbolic tangent or logistic sigmoid [64]. Sometimes, the output activation function is employed as an identity function. The weighted "peephole" conjunctions from the cell to the gates are displayed via dashed lines [109]. All the remaining joints inside the block are unweighted (or, correspondingly, have an assigned fixed weight 1.0) [60]. The only outputs from the unit to the remaining part of the net take place from the output gate multiplication.

, where i, o, f, c incarnate activations of input, output, forget gates and cell state accordingly, $x_1, x_2, \ldots, x_n$ denotes a sequence of inputs, W is a matrix, holding the conjunction weights between a couple of layers, b stands for the "bias term", σ designates the sigmoid activation function and tanh defines the hyperbolic tangent activation function.

LSTM blocks also can be mixed with conventional summation units, even though it is generally not required [64]. The identical output layers can be utilised for the LSTM nets as for regular RNNs [39].

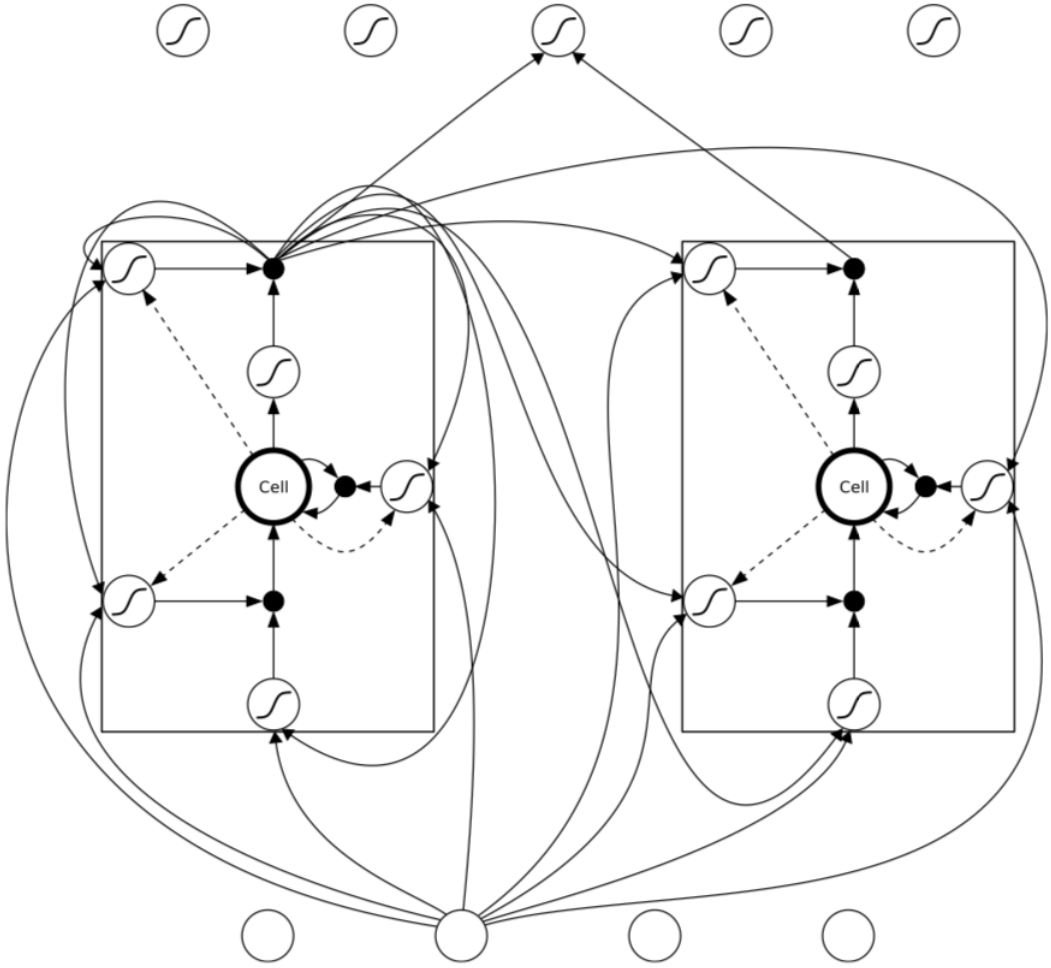


Figure 6: LSTM neural net [64]

Due to the application of the multiplicative gates in the architecture, LSTM memory cells are capable of retaining and accessing data over long periods of time, thus, resolving the vanishing gradient problem. While the input gate stays closed, i.e., the value of the activation function is approximate to 0, the cell's activation will not be overwritten by the novel inputs, occurring in the net. Therefore, it can be approachable by the network significantly later in the sequence through unblocking the output gate [190].
Over the latter decade, LSTM net has proved efficient at a wide spectrum of the synthetic problems, demanding large-scale memory, involving learning context-free languages [148], revoking sublime precision real numbers across prolonged noisy sequences [81] and diverse tasks, requiring accurate timing and estimation [61]. Notably, LSTM assists in resolving different artificial tasks that remain impracticable with other RNN structures [64].

Furthermore, LSTM has been deployed in various real-world issues such as protein subordinate composition prediction [80, 154], music generation [47], reinforcement learning [10], speech detection [65, 68], and handwriting processing [66, 109]. As expected, LSTM's benefits are most designated in terms of problems, demanding utilisation of long-scale contextual data.

3.5 Combination of convolutional and recurrent neural networks

The primary advantage of convolutional neural network (CNN) is the retrieval of the local correlations of temporal or dimensional patterns. Thereby, CNN has attained competitive performance in computer vision, natural language processing and speech recognition. Nevertheless, the weakest side of the CNN is operating with long dependencies between words. Reviewing relation categorisation, the nominal couples frequently spread in an extended series of contextual words [165]. Another example of a deep neural net is a recurrent neural network. Even though recurrent neural networks can overcome this disadvantage of CNN via operating with long-term dependencies, this neural net cannot gather the local features and faces vanishing gradient problem. In order to resolve the following problem, LSTMs have been proposed [78].
On one hand, CNN is good at learning local features from dimensional or temporal data and diminishing frequency variations. However, its major weakness is learning sequential conjunctions. On the other hand, RNN retrieves features from prolonging sequences and carries out temporal modelling but is not able to gather features in a parallel manner. Therefore, the combination of convolutional and recurrent neural nets allows deriving the model that is capable of learning spatial and temporal information in the input. Moreover, convolutional long short-term memory net is often applied in the visual time series prediction problems, notably, in the activity recognition, that is a major branch in our research.

4 Case Study: Posti Group Oyj

Posti Group Oyj was deployed as a case study for the Master's thesis where the ultimate goal was to acquire supervised machine learning model(s) for classification of the eight-hour work shifts of mail carriers and prediction of verification conformance for incoming work shifts, based on the quality of implementation of each separate task in a work shift. In addition, the concepts of predicting new task and its timestamp in the sequence of activities and forecasting forthcoming events in the work shift have been scrutinised.

Figure 7 illustrates the process of delivering mail volumes in Finland. To begin with, the foreign mail arrives at the particular Finnish postal centre ("ULK"), allocated in Vantaa, where it is further transferred to the closest postal centre ("POK"). In case an internal postal delivery is carried out, the mail volume is distributed from the shipment place to the nearby postal centre. Hereafter, the mail is sorted and transported via the chain of the postal centres to a local post service ("depo") according to the pre-calculated route. Eventually, the mail volumes are sorted in the depo, and the delivery is brought by a postal worker to the final destination. Altogether, the local transfer takes a couple of days at maximum. The data source, offered by Posti Group Oyj for elaborating the thesis, represents an annual event log of all planned and executed work shifts of the company's mail carriers across Finland.

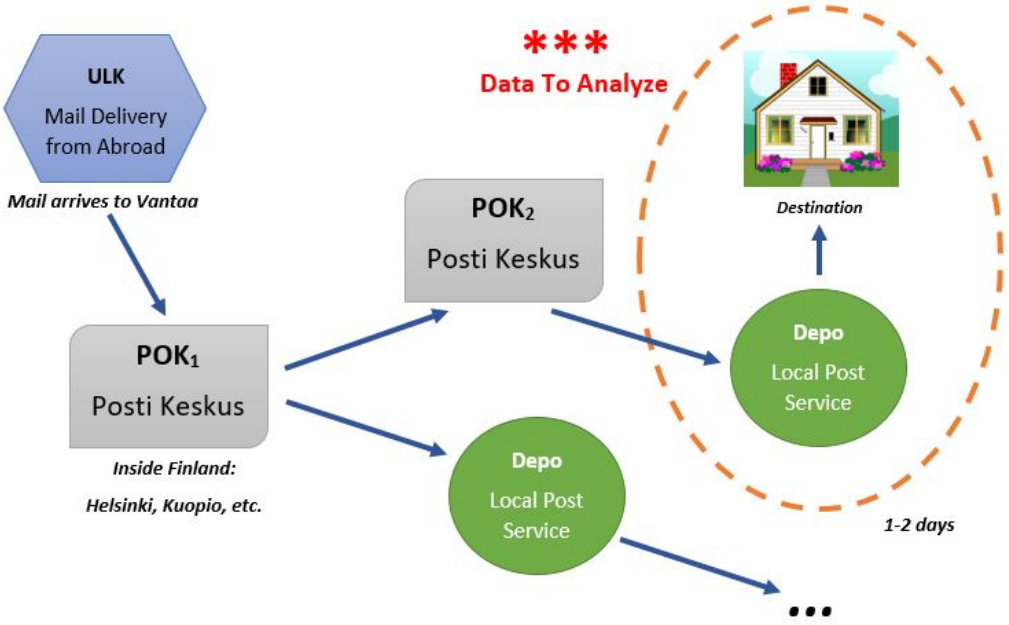


Figure 7: The process of delivering mail volumes

Figure 8 demonstrates the sorting stage at the local postal service which is an initial step in the local mail delivery process. Originally, the mail volumes are received in containers where the mail is onwards sorted by the postal employees regarding to three categories:

- presorted mail is sorted beforehand by the postal centers,
- other addressed mail is sorted within the local postal center,
- non-addressed mail implies batch distribution of newspapers, brochures, advertisements, etc.

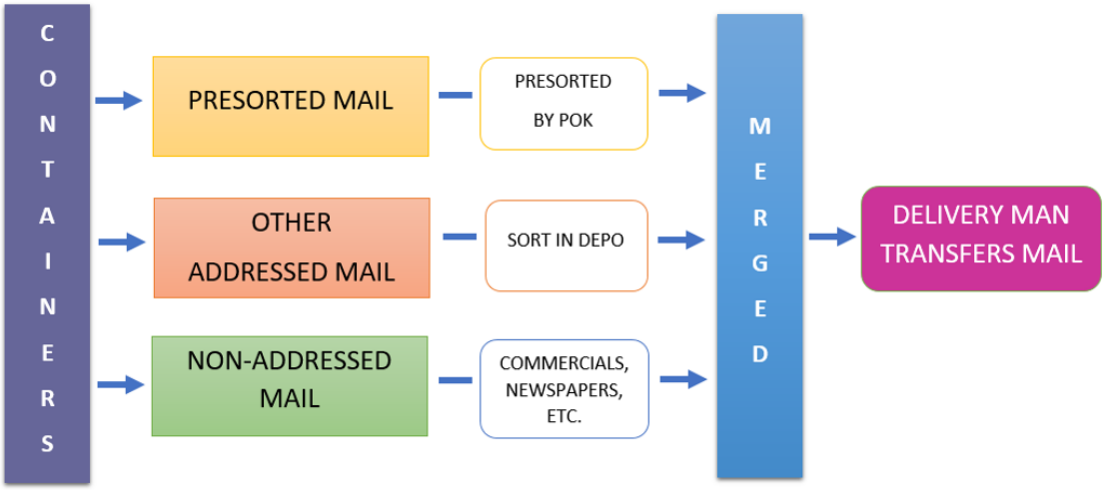


Figure 8: The sorting phase at the local postal service

Next, the sorted mail volumes are merged and delivered to the final destinations according to the pre-computed routes, devised to minimise the travelling distance. **Figure 9** displays the work shifts of mail carriers in the local postal centres where each shift is processed in the form of a task sequence with the marked timestamps. The supervisor (the head of the local postal service) examines the state of the completion of every task in each performed work shift and, based on the results, deduces the overall assessment of the shift:

- "OK" label - the tasks are executed successfully and the timestamps are accepted for this working shift.
- "NOTOK" label - issues occurred, while conducting the work shift.

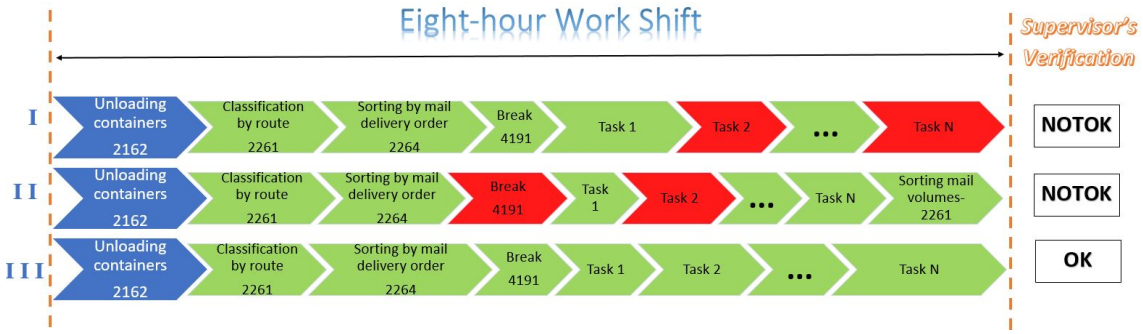


Figure 9: The structure of the eight-hour work shifts of mail carriers

Supervisor checks that all of the information concerning maintained tasks in a work shift is correct, otherwise, modifies a record by adding discovered observations, e.g., a timestamp for a task, and selects a "NOTOK" label for a work shift.
Table 2 demonstrates various tasks, implemented by postal workers during the work shifts, and their identification numbers.

Table 2: Diverse postal tasks with corresponding identification numbers

Task ID	Description
2262	Unloading containers
2261	Sorting mail volumes by addresses
2213	Pre-process picked up mail
2267	Sorting PO box mail
2361	Processing packages
4191	Break
3123	Meal transport or similar service
4112/4112	Outdoor delivery by a bike/car
4311	Home service
4161	Supervisor's verification
4112	Basic distribution (extra time for parcel distribution)
3123	Pickup service for a customer
3122	Delivery to distribution offices

Figure 10 illustrates the data flow during handling planned and executed work shifts in Posti Group Oyj.

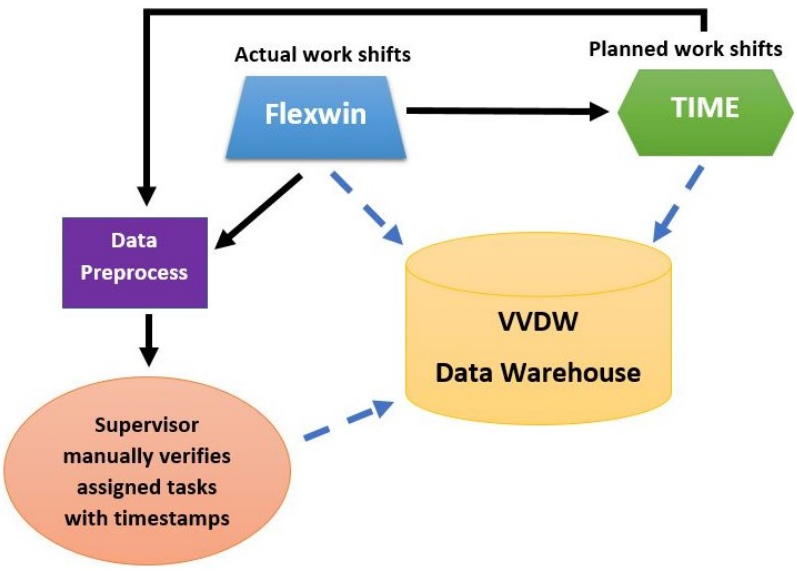


Figure 10: The data flow within Posti Group Oyj's organisation

Initially, the planned work shifts are stored in the "TIME" system, and a supervisor has an access to this structured data source. The information regarding the performed

work shifts is initially gained by "Flexwin" system which further transfers the processed data to the supervisors to verify the state of the implemented work shifts, also considering the primordial scheduling of the shifts. Originally planned work shifts and carried out ones retain in the "VVDW" data warehouse, the chief data storage and management system of the company.
Posti Timestamp Verification data set (PTVD log) depicts an event log, including all of the planned and later performed work shifts of Posti Group Oyj's mail carriers during 2017. The work shift is considered as a process with tasks, denoting various activities. Every task is marked with own identification number and respective timestamp. Moreover, the data set includes the work shift verification field, specifying the conformance state of a work shift, deduced a supervisor. **Figure 11** designates an instance of the processed PTVD log.

```
In [82]: whole_data[3434]
Out[82]:
{'WRKS_ID': 'XYZ',
 'class_label': 1,
 'wkrs_orig_clocks': {'codes': {0: {'DPT': '48290400',
    'timestamp': ['2017112006315807', 'TZ=15']},
   1: {'PRJ': '2261', 'timestamp': ['2017112006315908', 'TZ=15']},
   2: {'PRJ': '4191', 'timestamp': ['2017112007272108', 'TZ=15']},
   3: {'DPT': '48290400', 'timestamp': ['2017112007372407', 'TZ=15']},
   4: {'PRJ': '2264', 'timestamp': ['2017112007372508', 'TZ=15']},
   5: {'PRJ': '4191', 'timestamp': ['2017112009043908', 'TZ=15']},
   6: {'DPT': '48290400', 'timestamp': ['2017112009171407', 'TZ=15']},
   7: {'PRJ': '2264', 'timestamp': ['2017112009171508', 'TZ=15']},
   8: {'DPT': '48290400', 'timestamp': ['2017112009512907', 'TZ=15']},
   9: {'PRJ': '2261', 'timestamp': ['2017112009513308', 'TZ=15']},
   10: {'DPT': '48290400', 'timestamp': ['2017112010084707', 'TZ=15']},
   11: {'PRJ': '4114', 'timestamp': ['2017112010084908', 'TZ=15']},
   12: {'DKT': '90400038', 'timestamp': ['2017112010085104', 'TZ=15']},
   13: {'DPT': '48290400', 'timestamp': ['2017112014543407', 'TZ=15']},
   14: {'PRJ': '2261', 'timestamp': ['2017112014543608', 'TZ=15']}},
  'timezone1': ['2017112006315401Dtz=Europe/London', 'TZ=15'],
  'timezone2': ['2017112015060502Dtz=Europe/London', 'TZ=15']},
 'wrks_clocks': {'codes': {0: {'DPT': '48290400',
    'timestamp': ['2017112006315807', 'TZ=15']},
   1: {'PRJ': '2261', 'timestamp': ['2017112006315908', 'TZ=15']},
   2: {'PRJ': '4191', 'timestamp': ['2017112007272108', 'TZ=15']},
   3: {'DPT': '48290400', 'timestamp': ['2017112007372407', 'TZ=15']},
   4: {'PRJ': '2264', 'timestamp': ['2017112007372508', 'TZ=15']}},
  'timezone1': ['2017112006300001Dtz=Europe/London', 'TZ=15'],
  'timezone2': ['2017112015100002']}}
```

Figure 11: An example record from Posti Timestamp Verification data set

Each data record depicts a single work shift of an employee and contains the following information:

1. 'WRKS_ID' field - the identification number of the postal employee,
2. 'class_label' field - the binary conformance state, derived by a supervisor, where class 0 implies "OK" label and, otherwise, "NOTOK" label is indicated,
3. 'timezone1' and 'timezone1' fields - start and end times of the shift,

4. 'wkrs_orig_clocks' field includes the initial plan of the work shift, and 'wrks_clocks' field designates the structure of the performed work shift with the next attributes:
    - 'PRJ' code - the registered key for a postal task,
    - 'DPT' code - the department number,
    - 'DKT' code - the digital key, and
    - 'timestamp' code - the start time of the executed activity.

Ubiquitously, the data set contains 2086169 cases (work shifts) and 7245390 events (postal tasks). The longest work shift includes 72 activities.
In order to proceed with the data set, several processing rules were applied to the initial data source:

- tasks with the identification number 4191, denoting a break, cannot endure longer than two hours,
- the work shift must comprise more than one activity,
- the tasks must be engaged with the valid identification numbers, i.e., a task identification number cannot be empty or contain value -1,
- the tasks must involve valid timestamps, e.g., a timestamp with the specified year of 1900 is not congruent.

In case a work shift fulfils at least one of the indicated rules, it is labelled as "NOTOK". **Figure 12** exhibits the obtained histogram for the Posti Timestamp Verification data set, exemplifying accepted and non-accepted work shifts from 2017.

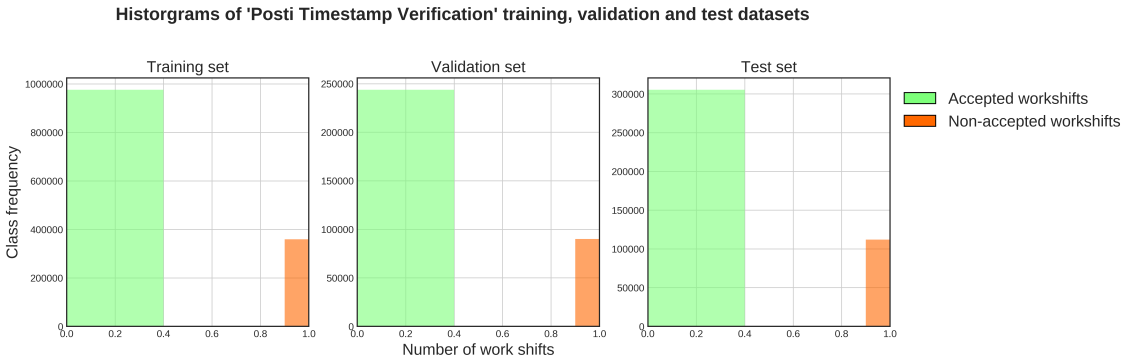


Figure 12: Class distribution in training, validation and test sets

The following class distributions are determined for the PTVD log:

- training data set - 975905 accepted and 359243 non-accepted work shifts,
- validation data set - 243758 accepted and 90029 non-accepted work shifts,
- test data set - 305307 accepted and 111927 non-accepted work shifts.

Figure 13 displays the histogram for the accepted and non-accepted work shifts regarding their lengths (the amount of tasks carried out during the work shift).

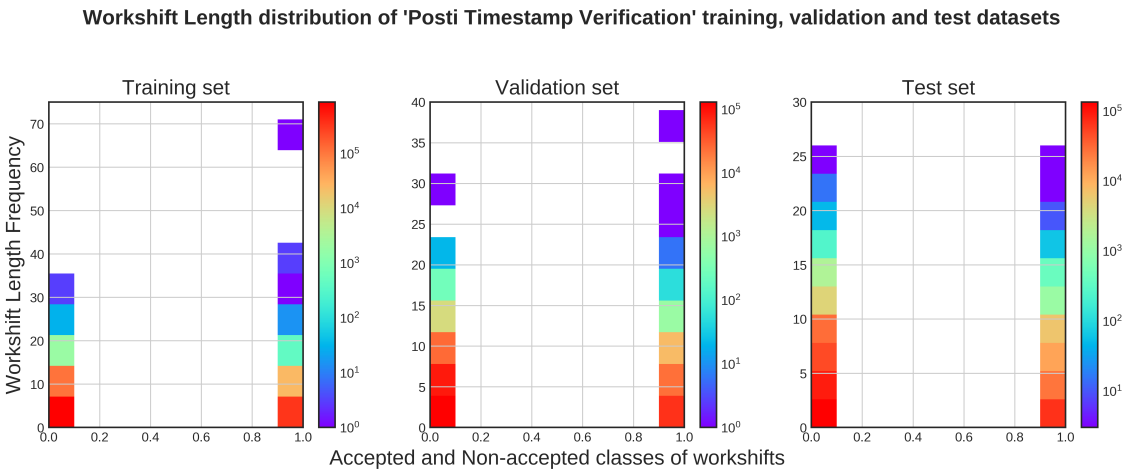


Figure 13: Class distribution of accepted and non-accepted shifts according to the length of work shifts

In addition, several denotations can be elicited, e.g., postal tasks are implemented regarding the logic of the delivery process where a work shift usually starts with a task "unloading containers". If a mail carrier comes from the route earlier than the end of the shift and the mail for the next day arrives - the sorting of the mail volumes is performed as well as the supervisor's work. Besides, if an employee fulfils all of the tasks before the end of the shift, Posti Group Oyj is forced to overpay for unoptimised scheduling of the work shift.
The overall objective of utilising the elaborated supervised machine learning algorithms is to develop software as service (SaaS) application that handles model predictions concerning verification conformance labels of the upcoming work shifts, earlier deduced by supervisors at the local postal centres manually. **Figure 14** picturise an exemplary interface of the SaaS application.

Employee ID	05.11.2018	06.11.2018	07.11.2018	08.11.2018	09.11.2018
234670					
448144					
686312					
944923					

Figure 14: A declaratory SaaS interface that uses the predictions regarding the supervisor's verifications for eight-hour work shifts of the postal workers at the local postal centres, obtained via derived machine learning approach. Green colour indicates that a work shift for an employee at the specified date is going to be accepted, whereas red colour marks non-accepted work shifts where the blanket assessment of the shift is categorised with label "NOTOK".

5 Research Methods

This section provides the contemplation of the research methods, developed and refined during the implementation of the thesis, as well as an overview on the obtained results. To develop recurrent and long short-term memory neural networks, the high-level deep learning library, Keras [27], was utilised. The experiments were performed in Microsoft Azure, a cloud computing service designed by Microsoft Corporation [33], on NVIDIA Tesla K80 GPU computing processor. Each experiment took on average between 120 to 250 seconds per training step, depending on the complexity of the selected structure of the neural net.

5.1 Predicting verification status of incoming work shifts

The following section describes the conducted experiments regarding the modifications of network architecture and hyperparameter tuning of the convolutional long short-term memory network (CNN LSTM) and their impact on the overall performance of the models in order to resolve a binary classification problem - the prediction of the conformance state of new coming work shifts. CNN LSTM represents a combination of a one-dimensional convolutional and long short-term memory neural networks that was deployed as a baseline machine learning approach in this research method. The procedure of determining the optimal set of hyperparameters is handled via a neoteric hyperparameter optimisation tool for Keras models, called Talos. Furthermore, the performance of the CNN LSTM network is compared with the state-of-the-art machine learning techniques such as a support vector machine (SVM), a logistic regression (LR), a random forest (RF), an adaptive boosting (AdaBoost), and k-nearest neighbours (k-NN) classifiers. The convolutional LSTM network with the best-attained performance was integrated into a software as a service application in Posti Group Oyj. Earlier, supervisors at the local postal centres have manually verified the quality of the execution of the work shifts on a daily basis.

5.1.1 Designed approach

Problem formulation: The vital goal of the Master's thesis is to predict supervisor's conformance on the arriving work shifts, based on the quality of the implementation of the postal task sequence within each case. The prediction of the supervisor's verification status can be formalised as a sequential binary classification problem that takes as an input a sequence (with length M) of previously carried out work shifts (during a month) or historical features and their corresponding conformance states $\mathcal{D} = \{(\mathbf{x}_i, p_i), i = 1, 2, \cdots, M\}$. Each work shift $\mathbf{x}_i \in \mathbb{R}^N$ is an input variable vector, denoting a sequence of tasks in a work shift. Assume that $\mathbf{x}_i$ includes T tasks where T is a length of the longest work shift in the data set. In case the length of a work shift is shorter, it is padded with zeros at the end to achieve the same lengths for the work shifts in the data set. Each task $\mathbf{x}_i^t$ is a one-dimensional input variable vector that indicates a specific task, executed in this work shift, e.g., a 'home service' is defined by $\mathbf{x}_i^t = 4311$. Therefore, a work shift can be implied as $\mathbf{x}_i = \{\mathbf{x}_i^1, \mathbf{x}_i^2, \cdots, \mathbf{x}_i^T\}$.

Furthermore, the conformance state $p_i \in \{0, 1\}$ is a binary predicting objective that designates the supervisor's verification for a work shift with "OK" label marked as 0, and "NOTOK" label otherwise. Besides, the verification status after fulfiling a set of serial tasks with length T within a work shift, p_i^T, can be prescribed simply as p_i. The output from a deep learning model is intrinsically a probability of successful execution of an incoming work shift. The resulting conformance state is evaluated in relation with the pre-defined threshold ε.
In order to devise a suitable machine learning model for the specified problem, the initial model option originally must respond to the format and contents of the data set. According to the work shift verification issue, the choice of the corresponding model falls on recurrent neural networks (RNNs) that represent a category of artificial neural nets, induced by the cyclical interrelation of neurons in the cerebrum [164], which utilises rolling function loops to preserve data.
Recurrent neural nets detect patterns in the data sequences such as texts [90, 105], audio signals [132], genome structures [143, 181], time series data, received from sensors [45], etc.. They highlight the interim and sequential properties of the data set, emanating into a temporal dimensionality of RNNs. Recurrent nets have congruent characteristics that make them a top choice for sequence labelling [64]:

- flexible in the maintenance of the context data - the data policies are adjusted regarding the type of the information stored or ignored,
- capable of operating with various kinds and abstractions of a data source,
- and retrieving the sequential patterns even in a case of observing the sequential distortions.

The main disadvantage of the traditional RNNs is inability to retain data for long time periods [79]. Thus, the availability of the context compass is bounded which is a crucial feature for the sequence labelling. Long short-term memory [81] is an advanced version of RNN architecture, founded on the principal of distinctive "memory cell" units. In the variety of synthetic issues, LSTM has demonstrated capability to keep and maintain data over long timespans [61, 62]. Consequently, the long short-term memory neural net is deployed as a baseline model for the conducted research method.
Figure 15 displays the baseline prediction structure of the LSTM neural network, employed in this research. An input vector $\mathbf{x}_i = (x_i^1, \ldots, x_i^T)$ is forwarded to a stack of several hidden layers, recurrently linked via weighed connections, to calculate the hidden vector $\mathbf{h}_i = (h_i^1, \ldots, h_i^T)$ and, therefore, the output vector $\mathbf{y}_i = (y_i^1, \ldots, y_i^T)$ where $\forall\ i \in [1, M]$. The outcome vector y_i^T is applied as an argument in the probability distribution $Pr(p_i|y_i^T)$ of the goal state p_i.
LSTM includes in its structure a memory unit that retains the data concerning a sequence of previously implemented steps. Thus, the developed model can carry out more context-sensitive decisions, meaning that the order and the results from the tasks, accomplished in the past within a work shift, affect the overall conformance state of this work shift and assist in the prediction of the verification property for

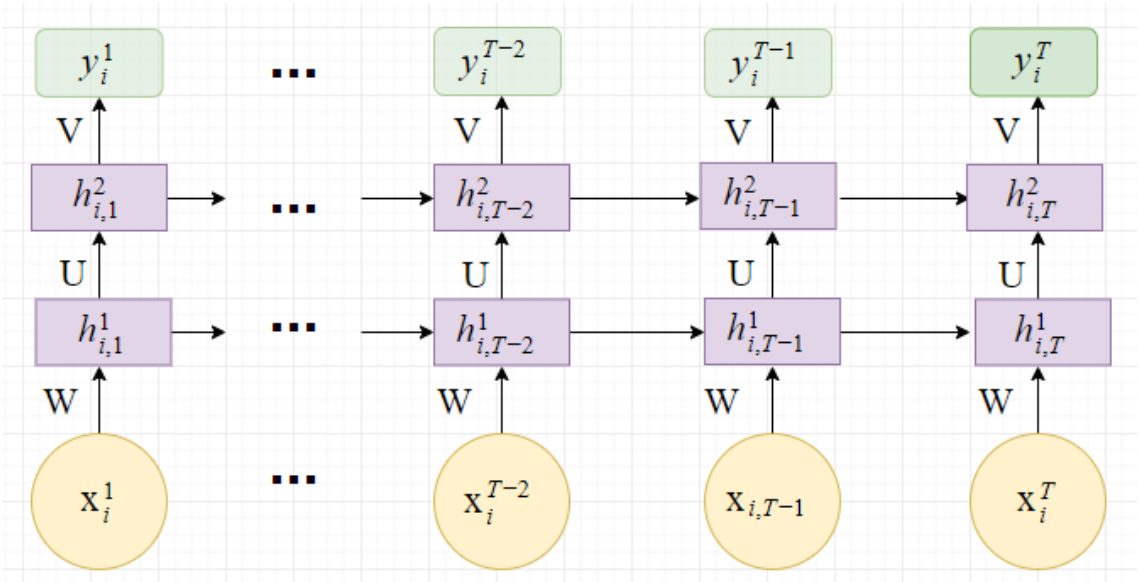


Figure 15: A baseline LSTM neural net with a "many-to-one" prediction architecture where the circles on the bottom present the input vectors, the central rectangles determine the hidden layers, and rounded rectangles on the top denote the output layer. The solid lines define the weighted connections, and $\mathcal{V}, \mathcal{U}, \mathcal{W}$ are the shared weight parameters.

other work shifts. LSTM memory unit with a single cell is presented on **Figure 5**. Also, the memory unit in the LSTM net is maintained through the iteration on the next set of functions [192]:

$$\begin{cases} i_t & = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{Ci}C_{t-1} + b_i) \\ f_t & = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}C_{t-1} + b_f) \\ z_t & = W_{xC}x_t + W_{Cf}h_{t-1} + b_C \\ C_t & = f_t c_{t-1} + i_t \cdot \phi(z_t) \\ o_t & = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}C_t + b_o) \\ h_t & = o_t \cdot \phi(C_t) \end{cases}$$

with σ incarnates the logistic sigmoid function, b-terms designate biases, i,o,f imply input, output and forget gates, and C depicts a cell vector, whereas all of them have a uniform dimension as a hidden layer vector h. The weight matrix W specifies the connections among the components, including input and hidden vectors, gates and cell. For example, W_{hf} represents a hidden-forget gate matrix, whereas W_{xo} points to an input-output gate matrix. The weight matrices that indicate connections between a cell and gates have the diagonal shape where the n^{th} component in every gate only takes an input from the n^{th} element of the cell.

Objective function: The work shift verification issue typifies an example of the binary classification problem where the outcome from the devised LSTM network, y_i^T, is a binary vector that determines a conformance state of the whole work shift i after "iterating" through all the T tasks within the work shift. For simplicity, the

predicted probability of the supervisor's conformance state for a work shift with T tasks matches the predicted probability of the whole work shift. Hence, $y_i^T = y_i$. Next, a binomial distribution $Pr(p_i = l|y_i)$ can be assessed via a softmax activation function at the output layer. The activation function also embodies itself as a hyperparameter, employed in the hyperparameter tuning process and depicted in the next section:

$$Pr(p_i = l|y_i) = \hat{y}_i^l = \frac{\exp(y_i^l)}{\sum_{l'=1}^{L} \exp(y_i^{l'})}$$

, where L is the amount of labels and, thus, in our problem $L = 2$. For the objective function, the binary cross-entropy cost function is utilised for determining the supervisor's verification status for a work shift i:

$$L = -\sum_{l=1}^{L} w_l \cdot [p_i^l \log(\hat{y}_i^l) + (1 - p_i^l) \log(1 - \hat{y}_i^l)]$$

, where label p_i^l is a binary value, w_l is the computed weight for the class l that is an important parameter to consider for training the network, due to the possibility of high imbalance of the classes in the data set, ($\sum_{l=1}^{L} w_l = 1$).
According to the programming side of the work shift verification problem, LSTM model is constructed as a traditional Keras neural network, utilising the sequential model API [43], where diverse network layers are attached to the originally blank sequential model regarding their intended functionality inside the model. The basic LSTM neural net has the following structure where the layers are specified in the order of inculcation:

1. Embedding layer serves as an initial hidden layer in the LSTM net that acquires an embedding for the entire set of postal tasks in the work shifts from the training set. This layer is depicted by several parameters such as [85]:

   - an input dimension - the manifold of the postal tasks,
   - an output dimension - dimensionality of the vector space where the postal tasks are interposed,
   - and an input length - the length of the input work shift sequences.

   The output of this layer is defined by a two-dimensional vector with a single embedding in the input work shift sequence for every post task.

2. Dropout layer designates a regularisation approach that is usually deployed in the deep learning to avoid overfitting [25]. Typically, the 25% dropout probability is applied for constructing the machine learning models where one-fourth of all weights in the LSTM layer is reset per every training iteration. Restraining neural networks from learning repeating data patterns allows gaining more robust nets, owing to more colligated regulations.

3. LSTM layer receives as an initial parameter the collection of the task embeddings, defined by the second attribute in the first hidden layer of the LSTM, to make the model learn higher-level temporal task sequences.

4. Flatten layer unrolls the values starting at the latter dimension to devise a prolonged feature vector to be employed by a dense layer for the terminal labelling.

5. Dense layer is the simplest layer in the architecture of the LSTM network since it returns as an output the final sum of the activations from the previous layer [1]. The size of the output of this layer is typically set to 1 to obtain the predictions with binary values where "OK" label is implied with value 0, and "NOTOK" label is embodied with value 1.

Traditional machine learning approaches fulfil classification tasks without using the transient correlations among data instances. Convolutional neural networks resolve this problem by applying convolutions throughout a temporal data sequence to detect dependencies within the data set. Despite, the range of convolution kernels limits the revealed scope of correlations among data instances [4]. Hence, conventional models are not adjusted to a broad set of activity-categorisation objectives and demand windows (filters) with the established dimensions. However, the LSTM network can be employed for learning the long-term dependencies in a sequence of an arbitrary length [78]. Therefore, the convolutional model is conjoined on the top of the LSTM net to extract features of input data and accelerate the operational time of the original LSTM network, responsible for the maintenance sequence prediction task. **Figure 16** illustrates the final structure of the designed CNN LSTM network. The LSTM neural net with a one-dimensional convolutional module with a standard set of arguments gains higher time performance on the training data set, due to the diversity of filters, exploited independently. Moreover, the parallelisation of the LSTM nets is a sophisticated process, since each computational step is dependent on the outcomes of the previous steps [103].

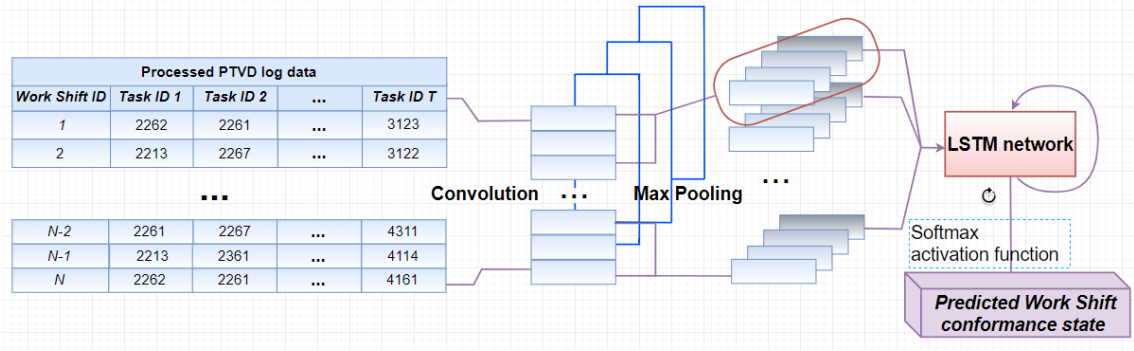


Figure 16: Ultimate design of the scrutinised convolutional LSTM network

In theory, the LSTM architecture can sketch any specified dynamical system. Although, the practical issues occur in the training process when the model parameters are contrived from the data to resolve a target problem. Due to the substantial impact on the overall model's behaviour, network architecture and hyperparameter tuning process must be investigated thoroughly to deduce the ideal machine learning

model with the highest possible performance. The essential difficulty in the choice of hyperparameters and global structure of the network arises owing to the shortcoming of firmly established techniques and methodologies for training various deep learning models. A broad range of new strategies and approaches has emerged from the literature in the last years [122, 146]. In the majority of situations, they require an extensive amount of proficiency from a developer to be properly implemented. The recently published research article "Hyperparameter optimisation with Keras" [91] proposed a novel and easy-to-learn hyperparameter optimisation and tuning tool for the machine learning algorithms, named Talos, that assists in detecting the hyperparameter settings for derived Keras models. Talos supports the functionality of Keras. Thus, any hyperparameter can be involved in the scanning process, and its influence is evaluated concerning the entire model's performance.
Moreover, one of the most remarkable attributes of Talos is an ability to remove the hyperparameters from the scanning if they decrease the model's performance. Since the parameter space extends in the obedience to $n!$ factorial, where n is a number of hyperparameters selected for tuning, the amount of possible permutations occurs to be immense. Therefore, Talos offers a couple of methods to diminish the number of possible scanning iterations such as:

- random reduction which eliminates the amount of transformations from the beginning,
- and non-random decline that lessens the number of permutations while scanning.

Since in our research the convolutional LSTM network is a baseline classification method practised for work shift verification problem, it is assessed with respect to a pair of categories of evaluation metrics:

1. Classification performance estimates the capability of the carried out model to deduce the verification states of the work shifts. In other words, classification performance depicts the frequency at which the model determines correctly or not if a work shift is accepted by a supervisor. Classification accuracy is portrayed via F1-score [71], precision [112] and recall [127] measures that are highly effective for the processing of imbalanced data sets, as in our study.
   Consider a binomial classification problem with distinct sets of positive and negative classes where true positives (TP) and true negatives (TN) designate the correctly forecasted positive and negative training instances respectively. False positives (FP) and false negatives (FN) denote the incorrectly categorised positive and negative training samples. Intuitively, positive and negative terms define the classifier's success. Contrariwise, true and false expressions designate whether the classifier's predictions match the labels of the ground-truth observations. The precision metric [112] is the ratio of the sum of true positives to the sum of positive observations. Thus, it can be taken into account as a probability for a positive training instance to be labelled correctly. The recall measure [127] is the ratio of the training observations that are forecasted positive and, in fact, are positive, e.g., true positives among all training samples that are ground-truth positive.

The F1-score, $f(\cdot)$, also known as F-score and F-measure, represents a metric for assessing the efficiency of the sample retrieval in regards with the cost of extraction [71]. In other words, F1-score is a scaled harmonic mean of precision and recall [127]. F1-score belongs to the interval $[0, 1]$ where the best-achieved score is 1, and the worst-obtained score is defined with 0. In the binary classification problem, both precision and recall conduce equivalently to the F1-score. In the multi-class issues, the global F1-score is computed by applying the weighted mean of the F-measure of every class. The metrics are reckoned as [133]:

$$\begin{aligned} \texttt{Precision} &= \frac{\texttt{TP}}{\texttt{TP} + \texttt{FP}} \\ \texttt{Recall} &= \frac{\texttt{TP}}{\texttt{TP} + \texttt{FN}} \\ \texttt{F1-score} &= 2 \times \frac{\texttt{Precision} \times \texttt{Recall}}{\texttt{Precision} + \texttt{Recall}} \end{aligned}$$

Furthermore, the receiver operating characteristic (ROC) analysis was deployed for computing the diagnostic performance of the developed machine learning models on a multi-class test data set, since it allows to collate the scoring classifiers regardless their categorisation functions, i.e., solely scoring functions [29]. ROC graph outlines the relative compromises among the true positives (profit) and false positives (loss) [52]. The ROC curve depicts the graph of sensitivity to 1-specificity [193]:

$$\begin{aligned} \texttt{Sensitivity} &= \frac{\texttt{TP}}{\texttt{TP} + \texttt{FN}} = \texttt{Recall} \\ \texttt{1-Specificity} &= \frac{\texttt{TN}}{\texttt{TN} + \texttt{FP}} \end{aligned}$$

Sometimes, a onefold correlation criterion is required for the evaluation of diverse ROC curves, introduced in the ROC analysis, especially in sophisticated cases. When there is a multitude of machine learning models, generated via distinctive algorithms or parameter configurations, a fast and smooth method of assessing them in regards to their predictive utility (without taking into account any specific observation) is necessary [29]. The area under the curve (AUC) score satisfies the mentioned above description of the criterion. AUC is utilised to totalize the ROC curve to a uniform value as a metric for the anticipated performance [53, 73]. AUC score can be measured via the posterior probabilities as [119]:

$$\texttt{AUC} = \frac{1}{n_k(n_k - 1)} \sum_{m=1}^{n_k - 1} \sum_{q=m+1}^{n_k} \frac{1}{n_m n_q} \left[\texttt{SR}_m - \frac{n_m(n_m + 1)}{2} + \texttt{SR}_q - \frac{n_q(n_q + 1)}{2} \right]$$

, where n_k implicates the amount of peculiar classes ($n_k = 2$), n_m designates the number of accessible points appurtenant to the m-th label, $\texttt{SR}_m$ denotes the sum of the ranks of posterior probabilities $p(k_m|\hat{y}_m)$ after ordering all

associated posteriors $p(k_m|\hat{y}_m)$ and $p(k_m|\hat{y}_q)$ in the incremental order with $\hat{y}_m$ and $\hat{y}_q$, serving as vectors of scores that refer to the actual classes k_m and k_q, respectively.
The AUC score of a model indicates the probability that a randomly selected positive sample is categorised higher than a randomly selected negative sample [157]. For an ideal detection method, AUC score achieves value 1.0, whereas AUC score is 0.5 for a random classifier in the binary classification problem [53]. In case the AUC score is below 0.5, no substantial classifier is found [157].

2. Predictive performance discovers the efficiency of the developed approach with respect to how accurately the model is capable of predicting the supervisor's verification statuses for the incoming work shifts. The following predictive performance measures are deployed in the evaluation of the experiments on network architecture and hyperparameter tuning where y_i defines a class of an observed sample i, $\hat{y}_i$ represents the predicted class of an observation i, and N embodies the overall quantity of predictions:

   - Accuracy denotes the ratio of the correctly labelled samples towards the whole amount of predictions:

     $$\texttt{Accuracy} = \frac{\texttt{TP} + \texttt{TN}}{\texttt{TP} + \texttt{FP} + \texttt{TN} + \texttt{FN}}$$

     The accuracy assesses how well the classifier fulfils the classification problem, and it is commonly employed as a statistical metric for sundry kinds of classifiers.

   - The binary cross-entropy or logarithmic loss (LogLoss) measure is applied as a basis evaluation metric. LogLoss can determine the performance of a multi-class categorisation model where a forecast is a probability distribution over the classes in the range $[0, 1]$. Mathematically, binary cross-entropy can be formulated as:

     $$\texttt{LogLoss} = -\frac{1}{N}\sum_{i=1}^{N} y_i \log_2(\hat{y}_i) + (1 - y_i)\log_2(1 - \hat{y}_i)$$

     , where N defines the amount of samples in the data set, y_i and $\hat{y}_i$ are an actual label of the observation i and the predicted one, respectively.
     Through minimisation of the logarithmic loss, the accuracy of the machine learning model is maximised. In other words, a lower value of this metric defines a more accurate classifier.

Besides, other machine learning techniques, engaged in the comparison analysis, including a support vector machine, a logistic regression, a random forest, an adaptive boosting, and k-nearest neighbours classifiers, are evaluated regarding the same set of estimation metrics.

5.1.2 Experimental setup

The standard classification machine learning problem can mathematically be formalised as:

$$\mathcal{F}(\mathbf{w}) = \min_{\mathbf{w}\in\mathbb{R}^n} \Big\{ \frac{1}{|\mathcal{X}|} \sum_{x\in\mathcal{X}} f(x, \mathbf{w}) \Big\},$$

, where $\mathcal{X}$ represents the categorised training set, $\mathcal{F}(\mathbf{w})$ illustrates the practised loss function, $\mathbf{w}$ is the weight vector (with a dimension n) that must be optimised, and $f(x, \mathbf{w})$ is the loss, deduced from training samples $x \in \mathcal{X}$ and their classes.
The procedure of determining the minimum value of loss function $\mathcal{F}(\mathbf{w})$ is named as "training" the neural net. Instead of assessing the loss function over the complete training set to maintain a single parameter update, a traditional approach implies calculation of the gradient over the batches in the training set [14]. A broad variety of optimisation algorithms can be used for training the LSTMs such as stochastic gradient descent (SGD) [184], Adam [89], RMSProp [3], etc. The main goal of these optimisation approaches is to minimise the objective function $\mathcal{F}(\mathbf{w})$ via iteratively executing the steps, defined as:

$$\mathbf{w}_{p+1} = \mathbf{w}_p - \eta \Big(\frac{1}{|\mathcal{B}|} \sum_{x\in\mathcal{B}} \nabla f(x, \mathbf{w}_p) \Big),$$

, where $\mathcal{B}$ denotes a batch, taken from a training data set $\mathcal{X}$, with a size $|\mathcal{B}|$, a step size (or learning rate) is designated as η, and the iteration index is indicated by p. These techniques are considered as gradient descent, using noisy gradients, which are frequently mentioned as minibatch gradients with fixed batch size [136].
Training a deep learning model is a complex and time-consuming process where the hyperparameters must be initialised beforehand. The neural networks, e.g., CNN and RNN, can attain several hundred hyperparameters that provide a high impact on the training process and the entire performance of the model. The structure-level parameters of the neural net involve, for instance, the number of hidden layers and the magnitude of neurons per hidden layer. The training-relevant parameters or, hyperparameters, include learning rate, optimisation techniques, regularisation methods, dropout probability, etc. [145]. The selected set of potential hyperparameters is converted into a dictionary with keys, exposing names of parameters, and values, used for tuning and further optimisation of a model. LSTM net is trained with respect to the received combination of hyperparameters.
In deep learning, scattered models and layers are frequently superimposed on top of each other. Hence, it is important to scrutinise the network depth, i.e., the number of hidden neurons/layers, to comprehend the behaviour of the model towards these structure-relevant parameters and explore the influence from the choice of diverse hyperparameters on model's efficacy.
The structure of a baseline convolutional LSTM neural net has been modified with regard to the following most regularly adjusted hyperparameters and network architecture parameters:

1. Learning rate (η) is one of the primary hyperparameters that maintains the pace at which the model's parameters need to be updated to deduce a neural

net with a satisfactory performance [9]. The process of deriving the optimal value can become an entangled endeavour in various practical problems. A choice of good learning rate can turn out to be a difference between a model, not capable of learning from the data, and a model, delivering outstanding results.
On the one hand, a neural net with a small learning rate scrutinises data slowly, requiring a significant amount of time to converge. On the other hand, overly high learning rate leads to weight dispersion and fluctuation around the minimum. Learning rate magnifies the potential for the training stage to achieve the global minimum, eliminating the possibility to be trapped at the local minimum that can give rise to erroneous outputs [137]. To deflect both of the extremes, the range of values for a learning rate is usually tested to observe the impact of learning on the cost function via utilisation of the validation data set. The learning rate that provides the inferior loss on the cross-validated set is sustained for the present state of the experiment. According to our process of devising the best LSTM network, the learning rate on the scale from 10^{-4} to 10^{-1} was explored.
Moreover, instead of operating with the fixed values, the adaptive learning rate can be employed where the learning rate is originally assigned with a high value, further reduced gradually after a number of epochs. Handling adaptive learning rate hastens the training process, although the decision regarding when to modify the step size should be carefully considered since it is encoded manually. To avoid using identical learning rate for each parameter update, various compound techniques for selecting step sizes can be deployed, including AdaGrad [32], AdaDelta [191], Nesterov-accelerated adaptive moment estimation (Nadam) [14], and root mean square propogation (RMSProp) [3] optimisation algorithms [9]. These and other optimisation methods were also explored during conducting experiments. The more detailed exegesis of optimisation methods is drawn later in this section. The possibility of carrying out an adaptive learning rate becomes a significant advantage in case of high sparsity of input data.

2. Batch size ($\mathcal{B}$) designates the amount of training samples propagated via a neural net in single forward or backward pass. On paper, there is no upper bound on batch dimension for a function approximation or regression problem. Although, the greater amount of training samples always assists in devising a better network, due to asymptotic convergence feature [51]. The only reason to search for the appropriate batch size is the limitations on refining data and computational capacity.
   The optimality of a batch size can be assessed according to training efficiency, depicted by a couple of peculiarities:

   - model's performance on an indicated task after training procedure,
   - and the number of steps, needed for training to attain an adequate performance to derive a feasible solution to a problem.

Owing to the concept of maintaining batches, a neural network is trained on fewer amount of samples with an overall training process taking less computational memory. This advantage is particularly substantial in case the computer system is not capable of processing the entire data set in the memory. There is no thorough guideline to determine the optimal size of a batch for efficient training. However, smaller batch size causes less accurate gradient evaluation, due to a negligible number of updating model parameters for every step. Thus, the large amount of steps is required during training, until the model reaches good performance. With a large batch size, the computational time taken by each step becomes longer. Consequently, there is a trade-off between the number of training steps and training time, necessary for each step, that makes the decision of selecting the appropriate size of the batch for training to be intricate.
Deciding the proper sample size for training a neural network has been little explored in the literature. Nonetheless, Lawrence and Fredrickson proposed empirical bounds to reckon the most accurate number of the training samples as [97]:

$$2(i + h + o) \leq N \leq 10(i + h + o)$$

where N denotes a batch size for training, i, h and o define the amounts of inputs, hidden and output units correspondingly. In our experiments, the batch sizes in the range from 2^5 to 2^{11} were investigated to observe their impact on the overall model's performance.

3. Choosing a number of hidden neurons (NHN) depends on a type of a problem. In detail, the amount of hidden units originally relies on the complexity of a baseline method to be approximated. An excessively small number of hidden neurons leads to the development of the overly mere model with a severe statistical bias. In other words, this model configuration causes an underfitting problem. In supervised machine learning and statistical learning theory, overly simple model obtains a significant training (or learning) error, contributing to the generalisation error and inducing poor model's performance. On the other hand, an unduly great number of hidden neurons summons to an overly complicated machine learning model with a low training error, but substantial generalisation error, owing to high variance, obtained from data overfitting. Stuart Geman described this phenomenon of model behaviour in regards to a number of hidden nodes as a "bias-variance dilemma" [59]. The other parameters that can affect the choice of sufficient number of hidden neurons include the amount of input and output neurons, the number of training samples, noise in data, nature of activation function, training method, etc. [51].
To define the number of hidden units for feed-forward neural nets, several theoretical "rules of thumb" have been designed. To define the number of hidden nodes for feed-forward neural nets, several theoretical "rules of thumb" have been designed. For instance, Hecht-Nelson used Kolmogorov's theorem to determine the upper bound for the necessary amount of hidden units [23]. This theorem asserts that any function of i variables can be deduced by a

superposition of a set of $2i+1$ univariate functions where i denotes the number of inputs. Thus, it specifies that $2i+1$ must be deployed as an upper bound on the number of hidden nodes, required for a backpropagation net with a single hidden layer.
Lippmann designated in 1987 that the maximum quantity of hidden units for a network with one hidden layer was $O(i+1)$ [107]. In 1989, Marchandani and Cao [118] discovered the dependency $h = i \ \log_2 P$ with the number of training patterns P . In the same year, Baum and Haussler [12] introduced criteria regarding the training sample size, units, and weights with respect to diverse statistical confidence levels.
In 1998, Lawrence and Fredrickson [97] proposed that the most precise evaluation for the number of hidden nodes was a moiety of the sum of inputs and outputs. Afterwards, they elicited another measure that matched the size of the training data set to the number of hidden units as:

$$\frac{N}{10} - i - o \leq h \leq \frac{N}{2} - r - o$$

, where N states the size of the training sample, and o expresses the number of output neurons. Ultimately, BrainMaker Neural Network Software company [98] provides a remarkably sophisticated software for producing large-scale data analyses with detailed documentation, including a whole roster of "thumb rules" for choosing suitable neural network structure. Unfortunately, as the documentation states, in the variety of cases, the rules of thumb are impractical, due to their non-consideration of several important factors such as the number of training samples, noise in the target data, and the complexity of the model. In implemented experiments, the range of hidden nodes $[2^6, 2^{10}]$ was investigated to determine the optimal performance for the developed convolutional long short-term memory networks.

4. The network architecture has a vital impact on the achievements of the model. A decision on the amount of applied hidden layers (NHL) plays a significant role on a par with the chosen number of hidden nodes per layer. The fundamental error of a model is composed of a couple of elements [88]:

   - a bias error that illustrates the systematic loss, induced by the bounded model flexibility,
   - a variance error that represents the stochastic loss, owing to the restricted accuracy of the parameter evaluations.

   These elements of the model loss reside in conflict since a bias error diminishes, while variance error magnifies for growing model's complexity. To find the trade-off between bias and variance errors, the network's architecture must be optimised thoroughly. In the conducted experiments, the number of hidden layers has varied from a single hidden layer up to 3 hidden layers.
   There are several research approaches that can be handled to optimise the neural net structure, known as "network growing" and "network pruning" [149].

In the network enlarging techniques, the number of hidden nodes or hidden layers can be augmented, beginning with a plain structured network, until the grown architecture fulfils the stated requirements. A prominent example of an enlarging approach is the cascade-correlation learning algorithm [50]. In the pruning processes, the original architecture is large. Further, it is pruned by weakening or removing some indicated weights. The concept of pruning is founded on the consideration that there is an immense number of redundant data, stored in the network. Neural net pruning is frequently implemented in one of the following possible ways [149]:

- the model's complexity regularisation,
- or elimination of weights via utilisation of the data on the second-order derivatives of the cost metric.

5. The backpropagation technique searches for the minimum of the error function in weight space, utilising the gradient descent approach. The integration of weights, minimising the error function, is deemed as a resolution of the learning issue. Since the following method performs the calculation of the gradient of the error function per iteration step, the global asymptotic and exponential stability conditions, i.e., continuity and differentiability of the error function, must be established. Another type of the activation function, other than the learning rate practised in perceptrons, has to be employed, since compound function, generated by interrelated perceptrons, is intermittent [141]. Thus, the error function is also intermittent. An activation function (ϕ) evaluates the transformations among layers,e.g., from the input layer to the hidden one. Initially, the neuron output function is derived as a threshold technique. Although linear, signum, sigmoid functions and step approaches are broadly deployed as output functions . Usually, inputs (x_j), weights (w_{ij}), thresholds and neuron outcome are considered as real values, binary or bipolar. The set of inputs is multiplied by corresponding weights and folded together to shape the net input to the perceptron yclept net [188]:

$$\texttt{net} = w_{i1}x_1 + w_{i2}x_2 + \cdots + w_{ij}x_{ij} + \theta$$

, where θ represents a threshold value, summated to the perceptrons. The neuron conducts as an activation function or mapping $\phi(\texttt{net})$ to generate an output y expressed as:

$$y = \phi(\texttt{net}) = \phi\Big(\sum_{j=1}^{n} w_{ij}x_j + \theta\Big)$$

, where ϕ is named as a neuron activation or a transfer function. The simplest activation function is a linear neuron transfer function (ramp function):

$$y = \phi(\texttt{net}) = \phi\Big(\sum_{j=1}^{n} w_{ij}x_j + \theta\Big) = \texttt{net}$$

The signum activation function obtains strictly bounded output values as $+1$ and -1, or, occasionally, 0 regarding the sign of a net:

$$y = \begin{cases} +1, & \texttt{if net} \geq 0 \\ -1, & \texttt{if net} < 0 \end{cases}$$

The most frequently used activation method is a sigmoid function, endowed with properties as non-decreasing, confined, and monotonic. Moreover, it elicits calibrated non-linear surface. Sigmoid function is computed in the following manner with c parameter, chosen arbitrarily, and its responsive $\frac{1}{c}$, named as the temperature parameter in stochastic neural nets [141]:

$$y = \frac{1}{1 + e^{-cx}}$$

Figure 17 illustrates diverse types of activation functions, including linear, signum and sigmoid functions. In addition, it displays changes in the shape of sigmoid function in relation to the value of c variable. Greater values of c parameter attain closer shape of sigmoid to the learning rate method. Moreover, in the limit $c \to \infty$, the sigmoid function converges to the learning rate at the origin.

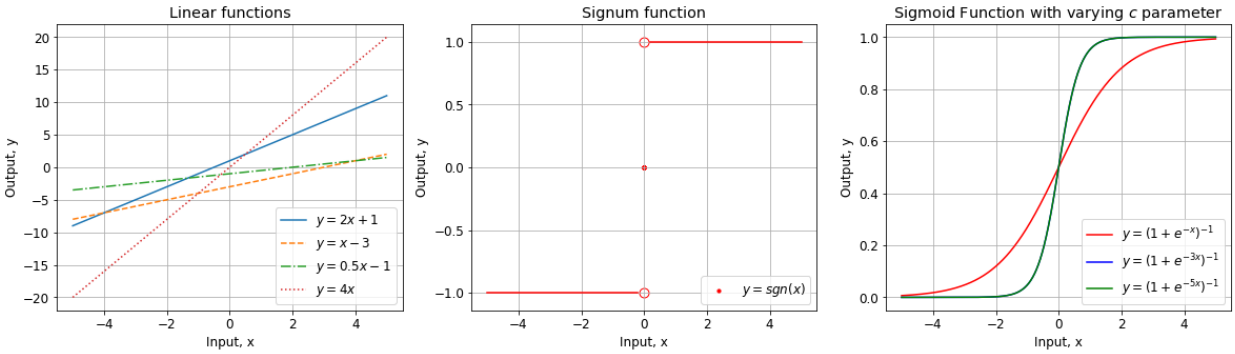


Figure 17: Examples of linear, signum and sigmoid activation functions

A variety of other activation functions has been introduced, and the backpropagation method is deployable with all of them. A differentiable activation function allows the function, evaluated by the neural net, to obtain the differentiability characteristic, considering that the integration function at every unit is defined as the sum of inputs since the net itself calculates only the function compositions. Hence, the error function turns to be also differentiable. Since the gradient direction is tracked to detect the minimum of this function, it is crucial that no areas occur in which the error function is ultimately flat. Forasmuch as the sigmoidal function always holds a positive derivative, the slope of the error function grants a higher or lower descent direction that can be monitored [141].
In 2011, rectivied linear unit or RELU activation function has been proposed

by [72], based on strong mathematical and physical concepts. It was introduced with an eye to enhancing the training process of deep neural nets. ReLU is founded on thresholding values at 0, e.g., $f(x) = \max(0, x)$ [5]. In other words, the outcome of the neural net is 0 when $x < 0$, and, inversely, model results into a linear function in case of $x \geq 0$. In contradiction with ReLU, an exponential linear unit (ELU) functions were devised with a capability of handling the negative values that allows this activation function to promote average unit activations closer to zero with less computational intricacy [44]. ELU can be estimated as [69]:

$$f(x) = \begin{cases} \alpha(\exp(x) - 1), & \texttt{if } x \leq 0 \\ x, & \texttt{otherwise} \end{cases}$$

, where $\alpha > 0$ is a hyperparameter that manages the value to which an ELU saturates for negative network inputs.
Figure 18 displays the expository graphs or ReLU and ELU activation functions with altering α hyperparameter.

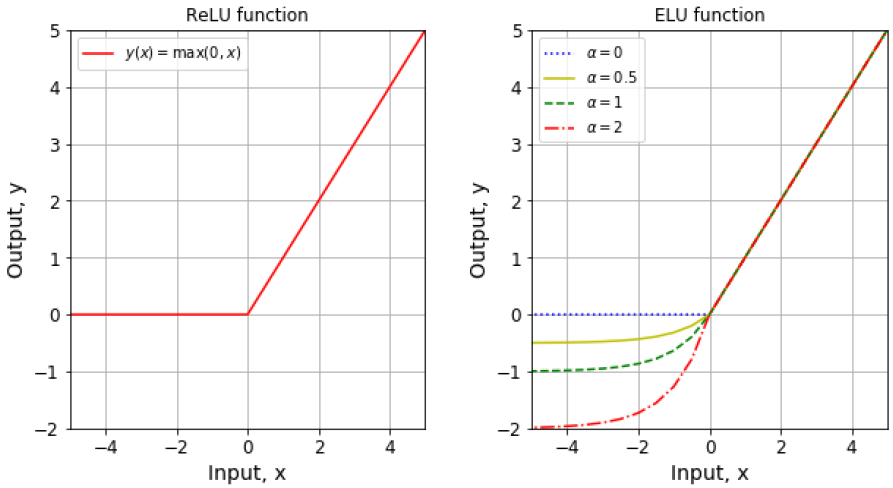


Figure 18: ReLU and ELU activation functions

Finally, the softmax function is generally applied in the last layer of a neural network-founded classifier. These networks are usually trained via cross-entropy condition, providing a non-linear alternative of multinomial logistic regression [168]. Softmax function extends the sigmoid method to several variables. The j-th term of a softmax approach is depicted by $\sigma(x)_j$, and in case an input includes N terms, a softmax function of x_j can be established by [63]:

$$\sigma(x)_j = \frac{\mathrm{e}^{x_j}}{\sum_{i=0}^{N-1} \mathrm{e}^{x_i}}$$

While carrying out experiments, the performance of the developed convolutional LSTM network was explored regarding the hyperbolic tangent, ReLU and ELU

activation functions. In the final layer of the devised neural network, softmax activation function was employed to render a categorical probability distribution over class labels.

6. In 2012, Hinton [156] proposed a new regularisation technique, named dropout, to avert neural networks from overfitting. Dropout precludes adaptation of neural network's weights to the training samples. Instead of denoting a detailed regularisation function $R_{\lambda}(\cdot)$, dropout is executed by maintaining a neuron activation during every forward pass in the training stage with some probability [14]. Explicitly, a randomly produced mask is employed to the outcome of the neurons in the hidden layer. The probability of every mask component to be 0 or 1 is determined by a hyperparameter $p_{\texttt{dropout}}$. When the training phase is completed, the activations are balanced by $p_{\texttt{dropout}}$ to handling the same anticipated outcome. Consequently, every hidden unit in the network is omitted with a specified probability, e.g., 50 %, for any training instance. Unlike the feedforward structures, a vanilla dropout in recurrent layers ordinarily generates low performance and, thus, it has mostly been engaged only in input and output layers of recurrent neural network [131]. Although, Gal and Ghahramani demonstrated in [57] that this disadvantage could be avoided by discarding the same nodes in every epoch of the gradient descent. The dropout probabilities $[0.25, 0.5, 0.75]$ were scrutinised in our set of experiments.

7. The epoch in backpropagation research represents a weight update or a training step. For every epoch, the backpropagation method establishes a sundry model, i.e., a network with a complex set of weights. If a neural net is trained on 500 epochs, the backpropagation approach scrutinises or traverses through 500 distinct models. Hence, a training phase of a neural network is observed as a scanning process through a tremendous amount of models, seeking for a model that has a set of weights that obtains the best generalisation performance [21]. Consequently, determining the optimal number of training epochs is a substantial decision for the neural net learning procedure.
Consider training data set $\mathcal{D}$ that is applied in combination with a backpropagation method to evaluate weights $\hat{w}$ of a model with an arbitrary number of hidden nodes. The indicated net is designated as:

$$\phi(\mathcal{D}; \hat{w}) \tag{1}$$

According to [76], a certain $\hat{w}$ is considered as a data instance in an abstract space of all feasible $\hat{w}$'s, named weight space. During the training stage, a wide range of weights $\hat{w}$ is examined by a backpropagation method. The selected weights $\hat{w}$ should derive the superior generalisation performance. To attain this result, every location, "taken" by the backpropagation algorithm in the weight space while training, must be "indexed". While the training converges and overfitting occurs, the model must be re-trained, finishing training procedure at the moment where the weights $\hat{w}$ match the allocation in the

weight space that gains reckoned optimal generalisation performance for a training process [21]. Hence, an epoch is utilised to inquire indexes in the weight space. The development of neural network models within the deployment of a backpropagation approach can be determined as the trace:

$$\Psi = \{\phi(\mathcal{D}; \hat{w}_e)\}_{e=1}^{E} \tag{2}$$

, where e represents present epoch, and E denotes the maximum amount of epochs, implied by a developer. Thereby, the application of epochs provides an opportunity for incorporating the abstract paradigm of a neural net's traverse via weight space during the training phase with a practical approach. The key benefit is a direct mapping of an epoch to the parameters of the neural network and their values.
Furthermore, Equation 2 does not take into account the number of hidden nodes in the network. To integrate the number of hidden units into this configuration, the Equation 2 is expanded as a trace of traces:

$$\Psi = \{\{\phi(\mathcal{D}; \hat{w}_e)\}_{e=1}^{E}\}_{h=1}^{H} \tag{3}$$

, where H serves as a maximum amount of hidden nodes, indicated by a developer. The method discovers a route through distinct weight space for each h in the Equation 3. The developer's purpose is to determine the values e and h, granting highest generalisation performance.
Also, high-power neural networks frequently tend to overfit during the optimisation procedure. While the training loss reduces through the optimisation process, the test loss infuses at some moment and begins to magnify again. This ineligible phenomenon is generally confronted by a early stopping optimisation technique, where the optimiser is interrupted for a particularised model, in case a user-defined early stopping criterion is fulfilled [114]. The early stopping solely guarantees that the empirical error of a current neural network is not minimised beyond the point of superior generalisation. In fact, neural network designers prefer to apply early stopping to high-power models for algorithmic and computational objectives and perform limitations for a specific model's label.
The fundamental target of deploying the early stopping is to track the modifications of the loss on the validation data set. Commonly, a small part of the training data is split. Its loss is handled as an evaluation of the generalisation error, relinquishing less efficient training data to determine the training error. A continuous assessment of this generalisation performance is further monitored, and the optimiser is paused in case the generalisation performance diminishes once more. This process has a wide set of benefits, particularly for large-scale data sets, where dividing a data set has an insignificant impact on the generalisation performance of the deduced model. Although, there are several evident disadvantages. Assessing the model on the validation data set in permanent spacing can be computationally intensive. Moreover, the selection of the size of the validation set represents a trade-off: a small validation

set derives an immense stochastic loss that can cause an improper stopping decision. Augmenting the validation set brings a more authentic assessment of the generalisation, but lessens the remaining quantity of the training data, depriving the model of presumable beneficial data insights. This trade-off is not airily eventuated, due to dependency on the features of the data distribution and decisions on the practical reasoning, e.g., exorbitance in the data set [114]. The dependency of the performance of the developed deep neural network has been studied in relation to the application of the various numbers of epochs, starting from 10 epochs up to 5000 epochs.

8. The diversity of optimisation methods is available for modifying the weights and biases within the neural network by executing iterations across the training data set. These optimisation algorithms include:

   - Stochastic Gradient Descent or SGD is a gradient-update method which guarantees convergence to a local minimum for a non-convex method if the learning rate is adequately small [14]. The parameter update equation for a network is defined as [17]:

     $$\mathbf{w}_{p+1} = \mathbf{w}_p + \eta \nabla F_p(\mathbf{w}_p)$$

     , where η represents the learning rate, one of the most important hyperparameters that must be mindfully evaluated to attain an effective result from the training. In practice, an immense learning rate produces a great quantity of kinetic energy in the gradient descent that causes the parameter vector to spring, precluding the entry to the confined field of the search space, where the cost function is lower [14].
     Step decay is one of the latest optimisation techniques, proposed to enhance the convergence with an eye to derive the optimal solution [17]. Step decay decreases the learning rate by a regular coefficient α, in case the loss has not diminished after performing an indicated amount of epochs [14]. Exponential and fractional decays can be depicted by this mathematical formulation [93]:

     $$\eta = \eta_0 e^{-\alpha p}$$
     $$\eta = \frac{\eta_0}{1 + \alpha p}$$

     , where α and η_0 represent decay hyperparameters, whereas p denotes present epoch.
     Even though stochastic gradient descent is considered as a reliable optimisation approach, its convergence rate is rather slow. Gradient descent is credible to wallow in a saddle point of the cost function [35]. These disadvantages have been scrutinised in the literature, and a few optimisation methods have been proposed, capable of overcoming these issues.

   - Momentum is a first-order optimisation technique that modifies the weights $\mathbf{w}_p$ via a linear conjunction of the present gradient $\nabla F_p(\mathbf{w}_p)$ and the

preceding update $\mathbf{V}_{p-1}$, scaled through the hyperparameter μ [14]:

$$\begin{aligned}\mathbf{V}_p &= \mu\mathbf{V}_{p-1} - \eta\nabla F_p(\mathbf{w}_p)\\ \mathbf{w}_{p+1} &= \mathbf{w}_p + \mathbf{V}_p\end{aligned}$$

According to this concept, the performed updates establish a velocity's direction toward a route of persistent gradient [162]. An adaption of the initial paradigma is the Nesterov momentum that frequently reaches a better convergence rate, particularly for smoother loss functions. Instead of computing the gradient at the present position, Nesterov momentum derives the gradient at an approximate forthcoming allocation where the update rules are [129]:

$$\begin{aligned}\mathbf{V}_p &= \mu\mathbf{V}_{p-1} - \eta\nabla F_p(\mathbf{w}_p + \mu\mathbf{V}_{p-1})\\ \mathbf{w}_{p+1} &= \mathbf{w}_p + \mathbf{V}_p\end{aligned}$$

- AdaGrad is a method for a first-order iterative optimisation that adjusts the differential learning rate to model parameters, executing greater updates for scarce parameters and negligible updates for common ones [32]. AdaGrad significantly enhances the robustness of stochastic gradient descent and is applied in training of large-scale neural networks [34]. Regarding a specified update on the data from earlier iterations $\nabla F_p(\mathbf{w}_i)$ and $i \in \{0, 1, \ldots, p\}$, a diverse update is deduced for every parameter q of the corresponding weight matrix [14]:

$$\mathbf{w}_{p+1}^{(q)} = \mathbf{w}_p^{(q)} - \eta\frac{\nabla F_p(\mathbf{w}_p^{(q)})}{\sqrt{\sum_i^p \nabla F_p(\mathbf{w}_i^{(q)})^2 + \varepsilon}}$$

  , where ε designates a small bias to preclude the denominator from being 0.
  One of the AdaGrad's main advantages is that it diminishes the necessity in manual regulation of the learning rate. However, AdaGrad's major defect is indefinite growth of denominator, due to storage of the squared gradients. Each appended term is positive, and the sustained sum proceeds on increasing while training. Hence, the learning rate reduces and finally turns to be infinitesimally small, whereas the method becomes incapable of obtaining supplementary knowledge from data. [86]

- Root mean square propogation or RMSprop was developed by Hinton as an adaptive technique, taking into account the notion of momentum [171] and resolving AdaGrad's disadvantage regarding drastically declining learning rates. Instead of employing the complete gradient vector, RMSprop optimises every parameter independently to magnify the corrections of slowly altering weights and diminish the update of magnitudes of rapidly modifying ones [16]. This optimisation method calculates the exponentially weighted transmitting mean of the transformation speed of each net

parameter in terms of the square of the gradient [14]:

$$v_p^{(q)} = \begin{cases} (1-\delta) \cdot v_{p-1}^{(q)} + \delta \nabla F_p(\mathbf{w}_p^{(q)})^2, & \texttt{if } \nabla F_p(\mathbf{w}_p^{(q)}) > 0 \\ (1-\delta) \cdot v_{p-1}^{(q)}, & \texttt{otherwise} \end{cases}$$
$$\mathbf{w}_{p+1}^{(q)} = \mathbf{w}_p^{(q)} - \eta v_p^{(q)}$$

Concerning the update rule, if the oscillation in the gradient updates is present, the step size is decreased by $1 - \delta$, in the opposite case, it is intensified by δ.

- Adam optimiser is a conjunction of the concepts of AdaGrad and momentum update which is employed for deriving a minimum of the objective function, deduced as a sum of differentiable functions [126]. Generally, Adam represents the adaptive learning rate technique that produces better results and, thus, it can be viewed as the gradient descent paradigma which is the most utilised, in practice. As RMSprop, Adam calculates the adaptive learning rates for every network parameter and keeps an exponentially diminishing mean of previous gradients squared. However, it also accumulates an exponentially diminishing mean of the gradient's moments [89]. The parameter update strategy for Adam is defined as [14]:

$$\begin{aligned} m_p &= \beta_1 m_{p-1} + (1-\beta_1) \nabla F_p(\mathbf{w}_p^{(q)}) \\ v_p &= \beta_2 v_{p-1} + (1-\beta_2) \nabla F_p(\mathbf{w}_p^{(q)})^2 \\ \hat{m}_p &= \frac{m_p}{1-\beta_1^p} \\ \hat{v}_p &= \frac{v_p}{1-\beta_2^p} \\ \mathbf{w}_{p+1} &= \mathbf{w}_p + \frac{\eta}{\sqrt{\hat{v}_p + \varepsilon}} \hat{m}_p \end{aligned}$$

, where m and v denote the first and second moments, relatively. Although m and v are originally set as zero-vectors, they are biased to 0 within the first epochs. To obviate the following situation, the first and second moments are rearranged with the help of $\hat{m}$ and $\hat{v}$.

- AdaDelta is a modified version of AdaGrad, aimed at decreasing its monotonically reducing learning rate [191]. AdaDelta constrains a window of stored previous gradients to an established magnitude w, instead of prodigally reserving W past squared gradients. The sum of gradients is periodically designated as a decaying mean of the whole set of previous squared gradients [34]. The following update rule corresponds to the AdaDelta optimisation algorithm:

$$\begin{aligned} m_p &= \beta_1 m_{p-1} + (1-\beta_1) \nabla F_p(\mathbf{w}_p^{(q)})^2 \\ v_{p+1} &= (1-\beta_2) t_p^2 + \beta_2 t_p \\ t_p &= \mu \frac{\sqrt{v_p}}{\sqrt{m_p}} \nabla F_p(\mathbf{w}_p^{(q)}) \\ \mathbf{w}_{p+1} &= \mathbf{w}_p - t_p \end{aligned}$$

- AdaMax is a first-order gradient optimisation technique that represents an alternative form of Adam, based on the infinity norm, with the following parameter update strategy [89]:

$$
\begin{aligned}
m_p &= \beta_1 m_{p-1} + (1-\beta_1)\nabla F_p(\mathbf{w}_p^{(q)}) \\
v_p &= \max(\beta_2 v_{p-1}, |\mathbf{w}_p^{(q)}|) \\
\hat{m}_p &= \frac{m_p}{(1-(1-\beta_1)^p)} \\
t_p &= \mu \frac{\hat{m}_p}{v_p} \\
\mathbf{w}_{p+1} &= \mathbf{w}_p - t_p
\end{aligned}
$$

  Each optimisation approach has own pros and cons, and there is no explicit directive for choosing an optimiser for a proprietary issue. Thus, all of these optimisation algorithms were estimated, since the best-to-perform LSTM neural network had to be elaborated.

In summary, the network architecture parameters, as well as hyperparameters, are thoroughly investigated to devise the LSTM network with the best feasible performance regarding diverse evaluation metrics, including overall accuracy classification score, F1-score, precision, recall, binary cross-entropy loss, and AUC score. The values for these parameters were selected on the grounds of the discussed earlier in this section literature approaches. The model parameters and hyperparameters of the neural networks with the most sharply correlated evaluation metrics that vividly depict the behaviour of models in terms of their configuration, are presented in the tables and elucidated in the next section.

5.1.3 Results

In order to explore the impact of the hyperparameter selection and the decisions on the network architecture, the convolutional LSTM network was elaborated as a baseline model with the parameters passed as input variables. The experiments were carried out regarding alternating learning rates, dropout probabilities, numbers of hidden layers and hidden units per layer, amount of epochs, batch sizes, activation functions, and optimisers. To refine the process of devising CNN LSTM network with the best-observed performance on evaluation metrics, Before feeding the work shift data into the baseline convolutional LSTM model, the data was handled according to the processing rules, defined in the previous chapter. Talos hyperparameter optimisation tool was used to avoid manual tuning for training each separate model. In addition, Talos takes into account the results of the training model, based on the validation accuracy.
To start with, a challenge of the selection of the suitable topology for the neural network arises. Choosing a number of hidden layers and a number of hidden units per layer is a significant part of the decision-making in designing a catchall neural network architecture. Even though these layers do not directly connect to the ambient

environment, they have an incredible impact on the overall yield. Both, the number of hidden layers and the number of hidden units per layer, must be mindfully reviewed. Applying an insufficient amount of nodes in the hidden layers leads to underfitting when the amount of hidden units is not enough to adequately identify patterns in the sophisticated data set. On the other hand, utilising an excessive amount of hidden nodes results into several issues. To begin with, network overfitting occurs. A neural network obtains tremendous capacity for information processing. However, the confined volume of the information, involved in the training data set, is not sufficient to train all of the nodes in the hidden layers. Furthermore, an extensive amount of nodes in the hidden layers can expand the time, taken for training the neural net. Also, the training time can increase to the point that it is infeasible to fairly train the model. Evidently, the compromise between choosing unduly many or excessively few nodes in the hidden layers must be reached, as the proper number of hidden layers must be emitted.
Therefore, the first phase of the experiments was executed to distinguish the best-suitable neural network architecture for analysing Posti Timestamp Verification data set where the numbers of hidden layers and hidden units per layer were scrutinised. Next, that network architecture was tested with varying learning rates, batch sizes, numbers of epochs, activation functions, optimisers, and dropout probabilities. **Figure 19** displays a hyperparameter dictionary, including the set of potential model parameters and hyperparameters. Onwards, the choice of the values for hyperparameters has been updated throughout the tuning process.

```
parameters = {
        'lr': [0.0001, 0.001, 0.01, 0.1],
        'NHL': [1, 2, 3],
        'NHN': [64, 128, 256, 512, 1024],
        'batchSize': [32, 128, 512, 2048],
        'optimiser': [SGD, Adam, Adadelta, Adagrad, Adamax, RMSprop, Nadam],
        'activationFunction': [relu, elu, tanh, softmax],
        'dropout':[0.25, 0.5, 0.75],
        'epochs': [100, 1000, 5000]
            }
```

Figure 19: A hyperparameter dictionary

In each subsequent element in the chain of the experiments, the model with the best performance was selected for the next round. The results on the evaluation metrics were recorded for diverse variations of the baseline convolutional LSTM network, where the default set of hyperparameters, deployed for the comparison reasons, is illustrated in **Table 3**.
As mentioned above, the initial set of experiments was implemented in regards to the network depth analysis. In a deep learning network, different numbers of modules and layers can be allocated on top of each other. Consequently, it is important to study the network depth in terms of the number of hidden layers (NHL) and the number of hidden nodes per layer (NHN) to comprehend the model's behaviour. **Table 4** summarises the obtained results on specified evaluation metrics for multiple numbers of hidden layers and a wide range of hidden nodes.

Table 3: A default set of hyperparameters for a baseline CNN LSTM network. The choice of the values for hyperparameters is further updated, based on the results of the successive experiments.

Hyperparameter	Default Value
Learning rate	0.001
Batch size	32
Optimiser	Adam
Activation function	hyperbolic tangent (tanh)
Dropout	0.25
Number of epochs	1000

The next denotations are employed for the evaluation metrics throughout the carried out research: ACC - classification accuracy, PE - precision, RE - recall, AUC - the area under the receiver operating characteristic curve score, and LogLoss - a binary cross-entropy loss.

Table 4: Results of network depth analysis

NHL	NHN	ACC	PE	RE	F-Score	AUC	LogLoss
1	128	68.14 %	65.78 %	68.17 %	66.95 %	67.42 %	34.01 %
1	256	71.18 %	68.19 %	70.46 %	69.31 %	70.66 %	30.55 %
2	128	77.59 %	76.77 %	77.55 %	77.16 %	78.51 %	24.89 %
2	**256**	**79.13 %**	**77.26 %**	**79.33 %**	**78.28 %**	**79.04 %**	**23.74**%
2	512	76.17 %	75.67 %	76.84 %	76.25 %	77.12 %	25.14 %
3	128	70.72 %	69.91 %	67.82 %	68.85 %	69.86 %	31.79 %
3	256	69.49 %	66.63 %	67.61 %	67.12 %	68.37 %	32.90 %

With a single hidden layer, the models tend to underfit the data, due to their incapability of learning the underlying patterns in the training set. A convolutional LSTM model with one hidden layer and 128 hidden units attained the prediction accuracy of 68.14%, F-score of 66.95%, AUC score of 67.42%, and binary cross-entropy loss of 34.01%. When the number of hidden nodes was increased to 256 units, the performance of the model slightly improved in relation to the whole set of the evaluation metrics. When another hidden layer was appended to the neural network, the performance of the model gradually enhanced. A convolutional LSTM model with two hidden layers and 256 hidden nodes per layer achieved the best results on the evaluation measurements: classification accuracy of 79.13%, F-score as 78.28%, the area under the ROC curve score as 79.04%, and a logarithmic loss of 23.74%. However, adding the third hidden layer to the model drastically diminished the performance by 9.74% on average across the entire set of the evaluation criteria, and, thus, the overfitting on the data occurred.
After identifying the network architecture of the model that acquired the best performance on the indicated evaluation metrics, which is a convolutional LSTM

neural network with two hidden layers and 256 hidden nodes per layer, the process of hyperparameter optimisation has initiated. To begin with, the hyperparameter tuning procedure started with the determination of the most vital hyperparameter for deep learning models, learning rate. Selecting the proper learning rate can result in a significant difference such as developing either a model, completely incapable of scrutinising patterns within the training data set, or a model that provides state-of-the-art output.

Table 5 demonstrates the gained results of CNN LSTM model regarding employing the manifold of learning rates. According to the outcome of this phase of the experiments, the convolutional LSTM model with the learning rate of 0.0001 acquired the highest performance, including the prediction accuracy of 80.54%, the precision of 79.89%, the recall of 81.07%, the F-score of 80.48%, AUC score of 81.03%, and the logarithmic loss of 22.54%. The overall performance of the model enhanced from 1.20% to 2.63% across the entire set of evaluation measures, comparing to the utilisation of the initially derived learning rate of 0.001. When the learning rate was assigned to 0.01, the performance of the deep learning model drastically diminished by 8.19% on average across all evaluation criteria. During this set of experiments, the learning rate of 0.1 was also scrutinised. However, high learning rate led to the disperse behaviour in the target loss function.

Table 5: Results of CNN LSTM model after applying various learning rates

Learning rate	ACC	PE	RE	F-Score	AUC	LogLoss
0.01	70.16 %	68.73 %	70.69 %	69.70 %	70.71 %	29.82 %
0.001	79.13 %	77.26 %	79.33 %	78.28 %	79.04 %	23.74%
0.0001	**80.54 %**	**79.89%**	**81.07%**	**80.48%**	**81.03 %**	**22.54 %**

With a small learning rate, the neural network is capable of learning the data slowly, taking moderate time to converge. Contrariwise, extensively high learning late causes the weight divergence and fluctuations around the minimum. Step size magnifies an opportunity for the model to reach a global minimum during the training process, excluding the chance to be trapped at the local minimum, contributing to the erroneous outputs [137]. A conducive learning rate strategy as a learning rate scheduler was utilised as a part of the callback function. It allows lessening the learning rate when there is no improvement observed on distinct estimation criteria, e.g., training or validation loss, an accuracy score, after traversing through "patient" amount of epochs [159].

After determining the optimal learning rate for the devised CNN LSTM model, next aim for the experiments was to explore the influence of the batch size, i.e., the number of training samples maintained before the optimiser executes a weight update, on the overall performance. **Table 6** displays the outcome of the experiments, conducted with respect to the modification of the batch size. Commonly, larger batch sizes contribute to the boosting of the training process. However, gaining sufficient accuracy is not guaranteed. On the other hand, smaller batch sizes cause deceleration of the training procedure. Although, obtaining higher accuracy is not as

problematical. The selection of suitable batch size grants optimiser a steady estimate of the gradient to deploy. According to this set of experiments, the optimal batch size for the elaborated convolutional LSTM network is 128.

Table 6: Results of CNN LSTM model after deploying different batch sizes

Batch size	ACC	PE	RE	F-Score	AUC	LogLoss
32	80.54 %	79.89 %	81.07 %	80.48 %	81.03 %	22.54 %
128	**82.97 %**	**81.19 %**	**82.35 %**	**81.77 %**	**82.84 %**	**21.23 %**
512	81.48 %	79.84 %	81.63 %	80.73 %	81.68 %	22.40 %
2048	78.01 %	77.49 %	76.87 %	77.18 %	78.53 %	24.84 %

After deducing the optimal learning rate and batch size, the following set of experiments was devoted to defining the optimiser that would lead to the enhancement of the derived so far model's performance. The variety of optimisers was scrutinised, including stochastic gradient descent [184], Adam [89], RMSprop [3], AdaDelta [191], AdaMax [142], Nesterov-accelerated adaptive moment estimation [14], and AdaGrad [32]. The idea of using an optimiser is to modify the weights and biases of the neural network by repeatedly traversing through the training data. Every optimisation procedure has own benefits and drawbacks. Besides, an evident strategy for choosing an appropriate optimiser for a specific machine learning problem does not exist. It is worth noting that Adam algorithm was employed as a default optimisation technique in the previous experiments. The stochastic paradigma of these algorithms assists in convergence within maintaining a smaller amount of epochs on data sets with surplus data and provides a more likely opportunity to avoid local minima.
Table 7 depicts the results from the experiments, implemented in regards to the application of different optimisers for a developed CNN LSTM network. AdaMax optimiser essentially outperforms other optimisation approaches, obtaining superior performance across the whole set of evaluation metrics with the prediction accuracy of 84.45%, the precision of 84.18%, the recall of 84.35%, the F-score of 84.26%, the area under ROC curve score of 85.32%, and a binary cross entropy of 18.45%. AdaMax optimiser reflects a variation of Adam, established by the deployment of infinity norm [89] that makes AdaMax exceedingly stable. As mentioned in Section 5.1.2, the update rule in AdaMax can be formulated as [142]:

$$v_p = \max(\beta_2 v_{p-1}, |\mathbf{w}_p^{(q)}|)$$

, where $\mathbf{w}_p^{(q)}$ is utterly neglected when its value is close to zero. Thus, the updates $v_1, v_2, \ldots, v_N$ are impacted by fewer gradients. Consequently, this optimiser is highly robust to the gradient noise. Besides, AdaMax is a top choice for sparsely modified network parameters, e.g., embeddings.
Moreover, AdaDelta optimiser demonstrated better performance than SGD and AdaGrad approaches. Stochastic gradient descent algorithm acquired the lowest performance among other optimisers. AdaGrad optimiser represents a modification of the SGD method, where the learning rate is updated independently for every network parameter on each training step with respect to the dimension of the gradients for

this parameter [32]. Further, RMSProp optimiser acquired higher performance on the evaluation measures comparing to AdaDelta approach. In fact, RMSProp is an advanced version of AdaGrad that resolves its issue with the drastically diminishing learning rates [3]. RMSProp algorithm is founded on the similar logic with AdaDelta optimiser, although AdaDelta approach applies root mean square of network parameter updates in the numerator modification strategy [191]. Nevertheless, Adam significantly outperforms RMSProp and AdaDelta optimisers. Nesterov-accelerated adaptive moment estimation algorithm comprises concepts of Adam and Nesterov accelerated gradient that allows executing more precise steps towards gradient direction through adjusting the network parameters with the momentum phase before actually calculating the gradient [142]. Nadam optimiser outpaces the traditional Adam optimiser, although it concedes to AdaMax approach.

Table 7: Results of convolutional LSTM model after deploying sundry optimisers

Optimiser	ACC	PE	RE	F-Score	AUC	LogLoss
Adam	82.97 %	81.19 %	82.35 %	81.77 %	82.84 %	21.23 %
AdaDelta	80.46 %	79.96 %	80.78 %	80.37 %	81.53 %	21.82 %
AdaMax	**84.45 %**	**84.18 %**	**84.35 %**	**84.26 %**	**85.32 %**	**18.45 %**
AdaGrad	80.28 %	77.43 %	81.14 %	79.24 %	79.78 %	22.97 %
RMSProp	81.77 %	81.11 %	82.07 %	81.59 %	82.12 %	21.74 %
Nadam	84.08 %	83.86 %	83.11 %	83.48 %	84.03 %	20.03 %
SGD	78.52 %	77.71 %	78.86 %	78.28 %	78.81 %	23.57 %

The next set of the experiments is devoted to studying the impact of the selection of an activation function on the developed convolutional LSTM. The deduced results are introduced in the **Table 8**. A neural network with a hyperbolic tangent activation function in the hidden layers gained the lowest performance, whereas employment of the rectified linear unit function presented relatively better results. This fact can be explained due to a couple of essential benefits of ReLU activation functions: a diminished likelihood of the occurrence of the vanishing gradient issue and the capability to derive less dense solutions. In comparison, a diversity of activation functions, involving hyperbolic tangent functions, will always result in a non-zero value, leading to the vanishing gradient problem. The ReLU activation functions do not always yield a non-zero value, and, therefore, a fewer number of neurons is practised, and the correlations among the features depreciate.

Table 8: Results of CNN LSTM model after utilising different activation functions

Activation function	ACC	PE	RE	F-Score	AUC	LogLoss
ReLU	85.09 %	83.82 %	85.13 %	84.47 %	83.50 %	17.67 %
ELU	**85.62 %**	**84.21 %**	**85.78 %**	**84.99 %**	**85.44 %**	**17.24 %**
Hyperbolic tangent	84.45 %	84.18 %	84.35 %	84.26 %	85.32 %	18.45 %

Utilisation of the exponential linear unit (ELU) activation function in the hidden layers enhanced the results of the CNN LSTM model from the previously implemented chain of the conducted experiments, reaching the prediction accuracy of 85.62%, the precision of 84.21%, the recall of 85.78%, the F-score of 84.99%, the area under the ROC curve score of 85.44%, and a binary cross-entropy loss of 17.24%. The exponential linear unit function can also resolve a vanishing gradient problem through initialisation for positive values. Moreover, ELUs enhanced the learning properties of the model regarding the other activation methods. Collating with ReLUs, ELUs obtain negative values that grant them an opportunity to forward the average unit activations closer to zero. Consequently, the learning process is accelerated, by virtue of the gradient transmitted closer to the unit natural gradient. Similarly to the batch normalisation, ELUs forward the mean to zero, involving essentially moderate computational resources. Furthermore, softmax activation function is maintained in the output layer for all variations of the CNN LSTM model, since the output of the neural network represents the probability that the sample belongs to a verified or non-verified class of work shift.
Hereafter, the acquired results from the subsequent set of the experiments, related to defining the optimum dropout probability, are indicated in **Table 9**. The goal of these experiments is to learn a more robust neural network which is less likely to overfit on the training set. Dropout is a regularisation method where randomly chosen neurons are ignored while training [25]. The output of this regularisation technique is that the neural net turns to be less sensitive to the certain weights of nodes. Thus, a model can perform better generalisation and less probably overfit the data. As a result of this set of the experiments in relation to the convolutional LSTM network, deduced from the previously implemented study phases, the dropout probability of 0.5 is optimal.

Table 9: Results of convolutional LSTM model after employing sundry dropout probabilities

Dropout	ACC	PE	RE	F-Score	AUC	LogLoss
0.25	85.62 %	84.21 %	85.78 %	84.99 %	85.44 %	17.24 %
0.50	**86.03** %	**85.45** %	**85.71** %	**85.58** %	**86.68** %	**15.76%**
0.75	82.55 %	82.00 %	81.33 %	81.66 %	80.37 %	21.36 %

As a final phase in the optimisation of the hyperparameters and selection of appropriate network architecture, the number of epochs, used for training, was scrutinised. The results of the experiments are demonstrated in the **Table 10**. While using 100 epochs for training, the model faces the problem of underfitting where the neural network is incapable of generalising to the new data. As the number of epochs magnifies, the greater is the amount of times the weights are updated in the neural net. The model goes from underfitting state to the optimal condition and, further, to overfitting status. With 1000 training epochs, the model achieves the best-observed performance. As the amount of epochs exceeds the optimal value, the model starts to memorise the data, instead of learning from it. Thus, the performance of the model drastically decreases. With an eye to overcoming bias-variance trade-off, the

early stopping criterion was implemented in the callback function to monitor the performance of the model regarding the loss on the validation set. This optimisation procedure was executed in the experiments to enhance the generalisation performance of the devised CNN LSTM model. Finally, after executing the whole chain of

Table 10: Results of CNN LSTM model after practising diverse numbers of epochs for training

Number of epochs	ACC	PE	RE	F-Score	AUC	LogLoss
100	70.22 %	68.75 %	70.97 %	69.84 %	70.91 %	29.74 %
1000	**86.03** %	**85.45** %	**85.71** %	**85.58** %	**86.68** %	**15.76 %**
5000	76.07 %	77.01 %	74.76 %	75.87 %	76.24 %	25.45 %

experiments with regards to the network architecture, i.e., number of hidden layers and hidden units within each layer, and tuning of sundry hyperparameters, including learning rate, batch size, optimiser type, activation function, dropout probability, and number of epochs, the convolutional LSTM network with the best obtained performance was derived. According to the broad spectrum of the retrieved results, the devised model represents the sophisticated CNN LSTM neural network with two hidden layers and 256 nodes per each layer that also applies the learning rate of 0.0001, the batch size of 128, dropout probability of 0.5, AdaMax optimiser, ELU activation function in the hidden layers, and 1000 epochs for training. This model obtained the following results on the evaluation measures: the prediction accuracy of 86.03%, the precision of 85.45%, the recall of 85.71%, the F-score of 85.58%, the area under the ROC curve score of 86.68%, and the binary cross-entropy of 15.76%.

5.1.4 Comparison with other existing machine learning algorithms

In this section, the effectiveness of the elaborated convolutional LSTM network is further investigated through the comparison with six traditional machine learning approaches, taking into account the earlier introduced evaluation measures:

- support vector machine (SVM) classifier [182] with the radial basis function (RBF) [20] kernel where the multinomial classification is maintained in regards with one-vs-one scheme [58],
- logistic regression (LR) classifier [30] with the categorisation deployed as one-vs-rest scheme [139],
- k-nearest neighbours (k-NN) classifier[125] with originally established 20 neighbours,
- and random forest (RF) [18], gradient boosting decision tree (GBDT) [153] and adaptive boosting (AdaBoost) [31] (with a default base estimator [135]) classifiers, devised on the concept of the ensemble learning.

In this frame of the research, five-fold cross-validation is practised to decrease the influence of the data dependency and improve the reliability of the experiments towards the scrutinisation of different machine learning approaches. The results of the carried out experiments are reported in the **Table 11**. The deduced convolutional LSTM significantly outperforms all of the traditional machine learning algorithms by a tremendous margin.

Table 11: Performance comparison of the developed CNN LSTM towards other machine learning algorithms, including support vector machine, logistic regression, random decision forest, gradient boosting decision tree, AdaBoost, and k-nearest neighbours classifiers with next denotations: ACC - classification accuracy, PE - precision, RE - recall, and AUC score - the area under the receiver operating characteristic curve score.

State-of-the-art machine learning approach	ACC	PE	RE	F-Score	AUC
CNN LSTM	**86.03** %	**85.45** %	**85.71** %	**85.58** %	**86.68** %
SVM	62.83 %	64.86 %	63.02 %	63.93 %	62.77 %
LR	59.27 %	60.63 %	59.47 %	60.04 %	59.44 %
RF	72.32 %	70.92 %	72.25 %	71.58 %	70.51 %
GBDT	76.03 %	75.01 %	76.67 %	75.83 %	76.07 %
AdaBoost	74.78 %	73.56 %	74.73 %	74.14 %	73.58 %
k-NN	72.24 %	71.04 %	71.36 %	71.20%	70.47 %

The worst performance was detected for the logistic regression model with the classification accuracy of 59.27%, the precision of 60.63%, the recall of 59.47%, the F-score of 60.04 %, and the AUC score of 59.44%. The support vector machine model derived slightly better results on the evaluation metrics in relation to the logistic regression model, with an average enhancement across the evaluation measures achieving the value of 3.71%. Logistic regression and support vector machine classifiers (with the radial basis function deployed as a kernel) are a common choice for the classification problems in the supervised machine learning. Logistic regression is a perfect approach for the linearly distinguishable problems, while support vector machine with RBF kernels are capable of maintaining the non-linearities within the data instances. Thus, SVMs can be easily handled in the non-linearly discernible issues, by virtue of a non-linear conversion of the input data via kernel functions and determining the arbitrarily formed decision boundary. Although, RBF SVM classifiers are more likely to overfit on the data, comparing to LR models. Hence, they require more careful hyperparameter tuning.
On the other hand, the ensemble learning methods outpace the SVM approach. The principal differences between random forests, gradient boosting decision trees, and adaptive boosting which affected the variance in the performance are [6, 123]:

- Gradient boosting decision trees employs week classifiers. This machine learning approach is based on the notion that a classifier is appended once per

iteration, and, therefore, the subsequent classifier is trained to augment the previously trained ensemble. GBDT increases the classification accuracy through minimisation of the loss, depreciated primarily due to overcoming bias, and integrating the outcome from multiple models.

- Random forests apply decision trees, inclined to overfit. With an eye to achieving higher accuracy, RF establishes a significant amount of decision trees, grounded on bagging. The fundamental concept is to resample the input numerous times and for every configuration train a novel classifier. In other words, a classifier is trained separately from the remaining ones in every training round, leading to the maximisation of the variance reduction. Diverse classifiers overfit the data in sundry manners. These deviations are averaged by means of voting procedure. RF resolves the error minimisation problem in a contrary way to GBDT, such as decreasing variance. However, a random forest is not capable of diminishing bias that is insignificantly greater than the bias of a single tree in the forest.

- Furthermore, AdaBoost likewise GBDT deploys weak learners and raises the prediction accuracy of a weak learner by repeatedly driving the attention towards uncertain samples that are challenging to predict. AdaBoost upweights the instances that were wrongly classified, whereas gradient boosting decision trees designate sophisticated samples by extensive residuals, derived through earlier training rounds. Thus, AdaBoost more concentrates on the complicated samples. Also, after the training, the predictor in adaptive boosting is added to the strong learner in accordance with its output weight. The higher is the performance of the classifier, the more it facilitates to the strong learner. In GBDT, the promotion of the predictor towards the strong learner is not calculated regarding its performance on the newly propagated instance, but through utilising the gradient descent optimisation procedure.

In addition, the k-nearest neighbours algorithm performs relatively the same in relation to the random forest approach. It is worth noting, the k-NN method is robust to the noise in the input data set and effective in terms of large-scale data.

5.2 Determination of incoming postal task and corresponding timestamp

The following section explores the research approaches for predicting an incoming activity and its timestamp, proposed in the articles, "Predictive Business Process-Monitoring with LSTM Neural Networks" N. Tax et al.[167] and "LSTM Networksfor Data-Aware Remaining Time Prediction of Business Process Instances" by N. Navarin et al. [128], in regards with a case study, Posti Group Oyj. Moreover,the network depth analysis of a baseline LSTM neural network is described, and matching experiments on the PTVD work log are carried out.

5.2.1 Designed approach

The prediction of a succeeding task in a work shift and corresponding timestamp is performed through scrutinising job and time estimate methods in regards to an arbitrary prefix length p:

$$\begin{aligned} f_a^1(hd^p(\sigma)) &= hd^1(tl^p\pi_{\mathcal{A}}(\sigma)) \\ f_t^1(hd^p(\sigma)) &= hd^1(tl^p\pi_{\mathcal{T}}(\sigma)) \end{aligned} \tag{4}$$

Similarly to the research method for predicting the verification status of incoming work shifts, described in section 5.1, every work shift $s \in hd^p(\sigma)$ is converted into a feature vector and deployed as an input for the neural network. Suppose that a work shift s involves T postal tasks where T is the length of the longest work shift in the data set. If the length of a work shift is less than T, then this work shift is padded with "zero tasks" at the end to achieve the same lengths for the work shifts. Intuitively, the tasks within a work shift are ordered according to the time of the implementation. Hence, a work shift can be denoted as $s \in \{x_1, x_2, \ldots, x_T\}$ where x_t depicts a postal job, performed at a time step $t \in [1, T]$ during the work shift. To specify the position of the job in a work shift, indexing is applied. Furthermore, the feature vectors are encrypted via a one-hot encoding scheme where a feature at position $\pi_{\mathcal{A}}(s)$ is indicated with 1, and remaining features are denoted with 0.
Several time-driven characteristics are introduced to the feature vectors for a work shift $s = \sigma(j)$:

- Monitoring time interval between preceding and present jobs in a work shift (or trace) gives LSTM neural network the capability to explore dependencies among the time periods at various tasks.

$$f_{t1}(s) = \begin{cases} 0, & j = 1 \\ \pi_{\mathcal{T}}(s) - \pi_{\mathcal{T}}(\sigma(j-1)), & \texttt{otherwise} \end{cases} \tag{5}$$

- Moreover, the majority of postal tasks can be executed solely during the business hours. Thus, a diurnal-based feature f_{t2} is utilised, corresponding to the daily time intervals (from midnight).

- In addition, a weekly-based feature f_{t3}, covering the time gap, beginning at midnight on Sunday, is also employed.

The time-based features f_{t2} and f_{t3} are responsible for a neural network to learn cases where a postal activity, implemented at the end of the work shift or workweek, impacts the anticipated time gap before the start of the next task.
The operational output o_a^p at the time point p represents the one-hot encoding of the subsequent postal task in the work shift. If the work shift terminates at a time step p, a novel task should not be forecasted. Consequently, the supplementary component is appended to the output vector, assigned with value 1 if the work shift is completed after p steps. The temporal output o_t^p holds the same value as the first time-driven feature vector of the subsequent event (f_{t1}), implying the time period between the

present and coming tasks. With the specified timestamp of the present postal task, the time attribute of the next job can be deduced. Further, the weights of the baseline LSTM net are adjusted via deep learning optimisation method RMSprop, founded on the principle of adaptable learning rates [3]. Hence, the logarithmic loss among the observed and predicted one-hot encodings of the incoming postal task and the mean absolute error (MAE) between actual and forecasted time intervals till succeeding job achieve minimum values.

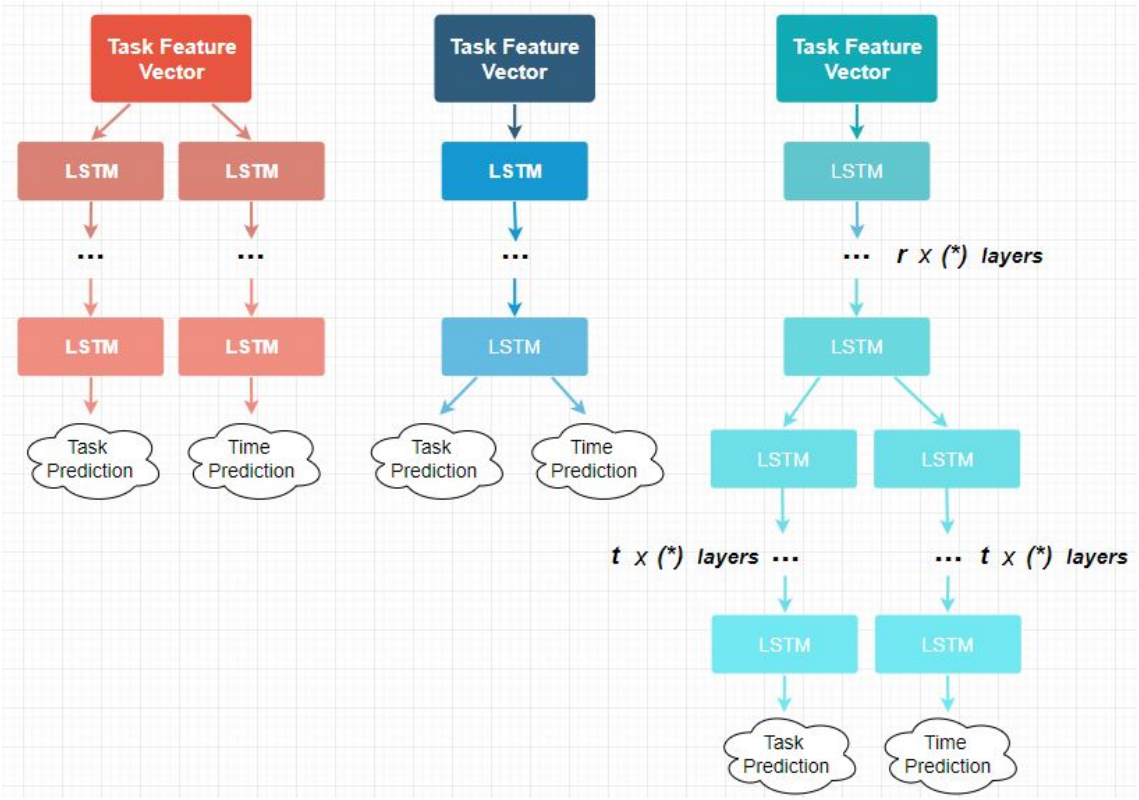


Figure 20: Modeling LSTM nets for the succeeding task (f_a^1) and time (f_t^1) prediction methods: (a) singular-task layers, (b) distributed multi-task layer, and **(c)** $r + t$ layers where r layers are joint [167].

Designing the next task and time prediction functions f_a^1 and f_t^1 can be carried out via manifold model structures, using long short-term memory network as a central component:

1. To start with, **Figure 20 (a)** displays a couple of networks that are trained independently for deriving f_a^1 and f_t^1, employing the identical set of input features at each time iteration.

2. Further, activity and time prediction approaches can be scrutinised in conjunction within a uniform LSTM model that produces several outputs, temporal and operational, adapted for a multi-task learning purpose [7] (**Figure 20 (b)**). The long short-term memory nets have illustrated to enhance the performance

on the entire set of distinct activities in a multi-task learning environment while jointly exploring a variety of NLP problems, involving word-category disambiguation and sequential labelling [105].

3. A combination of architectures, demonstrated on **Figure 20 (a)** and **20 (b)**, is a neural network that includes the set of distributed LSTM layers for deriving predictions on incoming job and time interval until incoming task, further connected to a number of layers, contributing to only one category of the prediction either f_a^1 or f_t^1 (**Figure 20 (c)**).

Moreover, the activity prediction method generates the probability distribution of sundry feasible continuations of a work shift. In order to perform adequate estimation, only the more credible work shift's extensions are taken into account.

5.2.2 Experimental setup

This section explicates the metrics and key approaches, deployed for the assessment of the performance of the aforementioned research method. Root mean square error (RMSE) and mean absolute error (MAE) are generally practised in model assessment studies as error metrics [22]. Mean absolute error estimates the mean magnitude of errors in the forecasted set without taking into account their directions. MAE is an average across the test instance of the absolute differences among the forecasted and observed sample [24]:

$$MAE = \frac{1}{\mathcal{N}} \sum_{i=1}^{N} |y_i - \hat{y}_i|$$

MAE is beneficial if the training data set is distorted with outliers. For instance, if a single predictions should haven beed obtained for all of the actual data instances that attempt to minimise MSE, then this prediction can be considered as the mean of all goal values. Contrariwise, if MAE is minimised, this prediction illustrates the median of all observations. Median is more robust to outliers, therefore MAE is more robust to outliers than MSE. The primary disadvantage of applying MAE is same same gradient during the training process, the gradient value will be immense even for the insignificant loss observations that makes it difficult for the learning process. However, this disadvantage can be resolved via dynamic learning rate that diminishes as the function gets closer to the minima. In the following case, MSE is more preferred since its gradient is high for larger losses and reduces as value of the loss is mininimized, making training process more accurate.
Root mean squared error denotes the square root of mean squared error, measuring the quality of the fit among the observed and predicted by model data [22]:

$$RMSE = \sqrt{\frac{1}{\mathcal{N}} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2}$$

RMSE is highly sensitive to substantial and minor errors in a set of measurements. Consequently, it can thoroughly portray the degree of dispersion within the data.

Willmott and Matsuura have proposed in their paper "Ambiguities inherent in sums-of-squares-based error statistics" that root mean square error is not sufficient marker for drawing the performance of the machine learning model and can be deceptive to imply average error [185]. Therefore, predictions on next task and a corresponding timestamp are estimated via application of MAE metric, since RMSE measure is highly perceptive to errors, acquired by outlier observations, whereas the time periods between implementations of the postal tasks are inclined to be altering on the scale from seconds to hours.
As a baseline method for forecasting the timestamp of the incoming activity, the remaining cycle time prediction approach is employed. Developed by B. van Dongen, R. Crooy and W. van der Aalst, it is based on performing the non-parametric regression on the records in the event logs where the present event is compared with a set of all past events [179]. The same logic can be used for the prediction of the time gap till succeeding job. The transition system is carried out through operating with either sequence, bag or set formats [177], taking as a foundation the subset of pre-processed Posti Timestamp Verification data set. The state in the transition system represents a mean of the time, required until the subsequent task. Overall, the subset of PTVD log includes around 14821 cases (work shifts) and 368454 events (postal tasks).
The performance of the prediction temporal and operative methods is estimated on PTVD log where the work shifts and postal tasks within each shift are organized in chronological order. Three-fours of the data set are employed for the training stage, whereas the performance of the research method is assessed on the rest of the data. The prediction methods are estimated on the entire set of prefixes $hd^p(\sigma)$ of work shifts in the test data set for $2 \leq p < n$ where $n = |\sigma|$.

5.2.3 Results

Table 12 demonstrates the results of the network depth analysis, where a long short-term memory neural net corresponds to a baseline model's architecture. The performance of this research method is designated with respect to the mean absolute error on the forecasted time interval and accuracy of the predicted incoming postal task. Selected prefix lengths reflect concise, intermediate and extended lengths of the work shifts in the PTVD log. In addition, the field "Entire set" stands for the mean performance on the complete set of all feasible prefixes.
The amount of distributed layers depicts the number of layers that contribute to the prediction of a postal job and time gap. The LSTM architecture with no distributed layers is illustrated on **Figure 20 (a)** where temporal and operative predictions are implemented by independent models. A case, when the amount of distributed layers matches the overall amount of layers in the developed model, indicates the neural net that does not involve any specialised layers. The following architecture is introduced on **Figure 20 (b)**.
The LSTM nets with the amount of distributed multi-task layers that correspond to the overall number of layers (**Figure 20 (b)**) and the LSTM networks with the miscellaneous structure (**Figure 20 (c)**) tend to obtain better performance for the

prediction methods on incoming postal job and time interval among the depicted LSTM structures (**Figure 20**).

Table 12: Experimental results for temporal and operative prediction functions with a long short-term memory network, employed as a baseline model for the network architecture

Number of layers	Number of distributed layers	Number of neurons per layer	MAE (hours) per prefix size				Accuracy
			5	30	60	Entire set	
1	0	128	2.39	1.83	1.27	2.09	0.7441
1	1	64	2.41	1.86	1.26	2.13	0.7428
1	1	128	2.33	1.84	**1.14**	2.08	0.7478
1	1	256	2.38	1.82	1.28	2.09	0.7454
2	0	128	2.36	1.80	1.25	2.06	0.7544
2	1	64	2.37	1.84	1.24	2.02	0.7569
2	**1**	**128**	**2.31**	**1.79**	1.15	**1.99**	**0.7663**
2	1	256	2.39	1.82	1.22	2.07	0.7507
2	2	128	2.35	1.81	1.16	2.01	0.7585
3	0	128	2.44	1.89	1.32	2.07	0.7508
3	1	64	2.34	1.83	1.26	2.06	0.7513
3	1	128	2.32	1.87	1.19	2.05	0.7526
3	1	256	2.37	1.80	1.15	2.06	0.7517
3	2	128	**2.31**	1.85	1.31	2.04	0.7519
3	3	128	2.38	1.83	1.27	2.08	0.7489
4	0	128	2.30	1.87	1.32	2.06	0.7505
4	1	64	2.42	1.86	1.30	2.07	0.7493
4	1	128	2.44	1.84	1.28	2.04	0.7518
4	1	256	2.38	1.83	1.24	2.09	0.7482
4	2	128	2.40	**1.79**	1.27	2.08	0.7476
4	3	128	2.41	1.85	1.26	2.09	0.7445
4	4	128	2.44	1.86	1.32	2.01	0.7437

Moreover, the model's behaviour can be scrutinised in terms of a number of neurons per layer. For LSTM models with a single distributed layer, 64 nodes and 256 nodes provide lower performance regarding both prediction methods: a model with 64 units per layer is underfitting the data, and 256 neurons lead to model's overfitting.
Table 13 indicates the outcome from the experiments with a single distributed layer where the traditional recurrent neural network is deployed as the fundamental approach for the prediction model. These models gain essentially lower performance in terms of task and time predictions, compared with the models using LSTM net as a basic structure.

Table 13: Results for temporal and operative prediction functions with a recurrent neural network, employed as a baseline model for the network architecture

Number of layers	Number of distributed layers	Number of neurons per layer	MAE (hours) per prefix size				Accuracy
			5	30	60	Entire set	
1	1	128	2.93	2.42	2.03	2.63	0.6902
2	1	128	2.77	2.44	1.89	2.53	0.7007
3	1	128	2.81	2.37	1.94	2.56	0.6976
4	1	128	2.87	2.40	2.02	2.62	0.6898

Table 14 reports the results from predicting the time intervals until the termination of the next postal task. These experiments are conducted in relation to the adjusted machine learning approach, originally composed by the W. van der Aalst, M. Schonenberg, and M. Song in [177]. All elaborated LSTM models (**Table 12**) outperform this method across the whole set of prefixes.

Table 14: Results for the time prediction approach, devised by W. van der Aalst, H. Schonenberg, H. and M. Song [177]

Type of abstraction	MAE (hours) per prefix size				Accuracy
	5	30	60	Entire set	
Set	3.29	2.33	1.40	2.47	-
Bag	3.32	2.41	1.38	2.44	-
Sequence	3.38	2.36	1.34	2.41	-

The highest performance, derived on the following postal task prediction over an entire set of prefixes, reached the categorisation accuracy of 76.63%. **Table 15** illustrates the results of the comparison analysis, using D. Breuker et al. [19] approach and J. Evermann et al. [49] method. Comparably, the accuracy obtained via D. Breuker et al. [19] algorithm attained the classification accuracy of 70.61%, whereas J. Evermann et al. [49] technique achieved the accuracy of 64.49%.

Table 15: Results for the activity prediction approaches, developed by D. Breuker et al. [19] and J. Evermann et al. [49]

Activity prediction methods	MAE (hours) per prefix size				Accuracy
	5	30	60	Entire set	
D. Breuker et al. [19]	-	-	-	-	0.7061
J. Evermann et al. [49]	-	-	-	-	0.6449

D. Breuker et al. [19] technique is founded on the concept of establishing a probabilistic model on the recorded data set of the preceding behaviour, allowing to forecast how presently executing operation will conduct in the future. Furthermore, J. Evermann et al. [49] approach is also based on the application of the baseline LSTM structure. However, there are a couple of distinctions with the original research method:

- J. Evermann et al. [49] approach employs an autonomous embedding technique to determine relevant feature attributes of the work shifts. Embeddings convert every case into an extensive feature vector. This technique has proven to operate with fairly high performance in the NLP tasks where the vocabulary size can significantly vary. However, in the Posti Timestamp Verification data set, the amount of specialised postal tasks has often a magnitude of dozens and hundreds where a relevant feature vector cannot be explored autonomously.

- J. Evermann et al. [49] technique utilises a neural network with a couple of LSTM layers where a fixed number of neurons is defined in every layer (500

nodes). Next, the performance of the scrutinised LSTM nets tends to diminish, while expanding the number of neurons per layer (from 128 nodes to 256 units). Consequently, J. Evermann et al. [49] method is potentially overfitting on the data set, causing a performance decline.

- The application of multi-task learning seems to enhance the overall forecast performance on the incoming activity method.

5.3 Prediction of postal tasks suffixes within work shifts

This research method is carried out according to the publications, "LSTM Networks for Data-Aware Remaining Time Prediction of Business Process Instances" by N. Navarin et al. [128] and "Predictive Business Monitoring with LSTM Neural Networks" by N. Tax et al. [167], employing PTVD log to predict feasible continuations of a work shift in terms of postal task sequences, further denoted as suffixes, and matching timestamps.

5.3.1 Designed approach

Iteratively applying methods for determining incoming postal task and the corresponding timestamp in a work shift, f_a^1 and f_t^1, provides an opportunity to perform prolonged predictions, capable of defining the entire set of postal tasks and traversing further than a single time step. The prediction methods for forecasting the overall sequence of tasks within the present work shift and time gaps until each subsequent job can be denoted as f_a^θ and f_t^θ. The goal of this approach is to attain the continuation of the work shift from the prefix, ending with element p, to be equal to the suffix of the trace of postal jobs, starting with the element p:

$$\begin{aligned} f_a^\theta(hd^p(\sigma)) &= tl^p(\pi_{\mathcal{A}}(\sigma)) \\ f_t^\theta(hd^p(\sigma)) &= tl^p(\pi_{\mathcal{T}}(\sigma)) \end{aligned} \tag{6}$$

The suffix of the work shift can be defined through repeatedly predicting incoming tasks and the time periods before the succeeding jobs, until the end of the work shift is reached. The completion of the work shift, specified as θ, can be deduced as a chain of implementations of the next task prediction method f_a^1. The continuation of the tasks within the work shift can be formulated:

$$f_a^\theta = \begin{cases} \sigma, & f_a^1(\sigma) = \theta \\ f_a^\theta(\sigma \cdot s), & \texttt{otherwise} \end{cases} \tag{7}$$

, where work shift is designated as $s \in \xi$, $\pi_{\mathcal{A}}(s) = f_a^1(\sigma)$ and $\pi_{\mathcal{T}}(s) = f_t^1(\sigma) + \pi_{\mathcal{T}}(\sigma(n))$ with $n = |\sigma|$.
Furthermore, the sequence of the timestamps until incoming tasks can be derived in the same manner as the extension of the activities within work shift:

$$f_t^\theta = \begin{cases} \sigma, & f_t^1(\sigma) = \theta \\ f_t^\theta(\sigma \cdot s), & \texttt{otherwise} \end{cases} \tag{8}$$

, where $s \in \xi$ implies a a work shift, $\pi_{\mathcal{A}}(s) = f_a^1(\sigma)$ and $\pi_{\mathcal{T}}(s) = f_t^1(\sigma) + \pi_{\mathcal{T}}(\sigma(n))$ with $n = |\sigma|$.

5.3.2 Experimental setup

The performance for devising a sequence of incoming tasks f_a^θ for a specified trace of postal jobs $hd^p(\sigma)$ is estimated via assessing the distance metric among the predicted suffix $f_a^\theta(hd^p(\sigma))$ and the ground-truth extension of the work shift $\pi_{\mathcal{A}}(tl^p(\sigma))$. Generally, a distance between a couple of sequences is evaluated algorithmically with regards to a number of operations, necessary to transfer one sequence into another, or in terms of geometric correlation [82]. A wide scope of distance measures for managing sequences includes Levenshtein distance, Damerau–Levenshtein distance, Hamming distance, Jaro–Winkler distance, Lee distance, and so on. Edit distance is the most common distance metric which detects a minimal amount of substitutions, insertions and deletions operations to convert one sequence into another sequence.
If the activities in a case are implemented simultaneously, the distance metrics, excluding edit distance, should be thoroughly investigated. For instance, a representation of the predicted subsequent tasks as a tuple $\langle s, t \rangle$ and a ground-truth suffix of a work shift as a tuple $\langle t, s \rangle$ is granted as an insignificant mistake, due to the irrelevance of the consecution of the parallel task fulfilment within a tuple. Although, the edit distance calculates a total cost of 2 to this prediction while transferring the predicted set of tasks into a ground-truth one takes single deletion and insertion operations. Conversion of a tuple $\langle s, t \rangle$ to $\langle t, s \rangle$ obtains a cost of 1 via evaluation through Damerau-Levenshtein distance measure. This distance measure is determined in the same way as Levenstein distance, except the permutation of adjacent events is recognised as an interchange operation [101]. Hence, Damerau-Levenstein distance is utilised in this research approach to provide more accurate contemplation of the prediction procedure [166]. Taking into account highly altering amount of tasks per working shift, Damerau-Levenshtein distance metric is regularised by the maximum length of the observed sequences of the postal tasks and the length of the predicted sequences of succeeding jobs [166]. The Damerau-Levenshtein similarity is computed as [28]:

$$\texttt{sim}_{\texttt{Damerau-Levenshtein}}(s_1, s_2) = 1.0 - \frac{\texttt{dist}_{\texttt{Damerau-Levenshtein}}(s_1, s_2)}{\max(|s_1|, |s_2|)} \tag{9}$$

Furthermore, a pioneer time prediction technique has been introduced by W. van der Aalst, M. Schonenberg, and M. Song in [177]. The authors of the publication depict a prediction method that retrieves a transition system from an event log and elaborates it with insights, discovered from the time data which is withdrawn from historical records. The finite state machine includes a transition system, designed in relation to the indicated abstraction of the events in the log file. The time prediction is carried out by employing time-driven data, e.g., a mean continuation of the certain type of tasks in the state of the transition system, referring to the currently executed sample.
One of the most novel and popular methods in predicting continuations of the cases

of arbitrary lengths in the event logs is ProM plugin, described by M. Polato et al. [134] in "Time and Activity Sequence Prediction of Business Process Instances". To begin with, a transition system is derived from an event log. The deep learning model is processed for every state in the transition system to determine the subsequent task. The performance of the framework is assessed in regards to predictions that are gained for a fixed amount of succeeding tasks. On the other hand, the original research method evaluates the postal tasks suffix until the work shift is completed. Hence, the experiments, involving the deployment of the ProM plugin, were maintained in a way that the entire sequence of incoming tasks was predicted.
In order to compare the performances of both research approaches on Posti Timestamp Verification data set, a basic neural network was constructed with several LSTM layers and a single distributed layer where the number of neurons per every layer was assigned to 128 units. This neural network demonstrated the best performance (**Table 12**) with respect to the research approach for the prediction of the incoming task and determination of the time interval until the subsequent postal job. The same subset of PTVD log is used in this research that includes around 14821 cases (work shifts) and 368454 events (postal tasks).

5.3.3 Results

Table 16 accumulates the results of postal tasks suffix prediction on Posti Timestamp Verification data set where the LSTM approach significantly outperforms the method, designed by M. Polato et al. [134]. Although the performance of LSTM neural net is greater comparing to ProM framework, the overall result is moderate. The PTVD log includes a numerous amount of work shifts with the same subsets of postal tasks, e.g., the work shift often starts with sorting the mail volumes which are further delivered to the different addresses. This process represents a chain of identical postal jobs. The work shifts with 5 or more similar postal tasks in a row are not rare. Unfortunately, LSTM nets are not persistent to the following peculiarity of the PTVD work log. While performing the forecasts of the postal task sets, LSTM net results into the prediction of suffixes with extensively long job sequence, comparing to the ground-truth observations.

Table 16: Postal tasks suffix prediction in relation to the Damerau-Levenshtein similarity

Research approach for activity suffix prediction	**PTVD log**
Devised LSTM neural network	0.4286
ProM framework [134]	0.0857

6 Conclusion & Future Work

Nowadays, the terms "neural networks", "machine learning" and "deep learning" have grown from the comprehension of buzzwords to actual industrial, educational and business practices. These research areas can be perceived as the layers of a pyramid. To begin with, neural networks constitute a foundation of this pyramid. The aim of utilising neural networks is to further maintain the base for the deployment of artificial intelligence. Next, the subsequent layer comprises the machine learning, serving as a software platform for the neural networks. Thus, neural networks allow developers to discover potential insights within the large-scale data sets. The top level of the pyramid is taken by the deep learning, a special field of machine learning, which has earned incredible popularity among researchers over the last decades, owing to its outstanding opportunities: diminished inquiry in the computational power and capability to operate with extensive data sources.
Deep learning has introduced state-of-the-art performances in a broad spectrum of applications such as computer vision [165], speech, text and audio recognitions [87, 132, 138, 152, 158], information retrieval [100], natural language processing [90], bioinformatics [143, 181], and so on. In the industrial systems, data-driven process monitoring is rapidly expanding its influence, due to the widespread practising of low-cost technologies for gathering and processing sundry data sources. Meanwhile, deep learning proposes beneficial tools for operating with and analysing large-scale machinery data sources and human resource information systems.
Posti Group Oyj was deployed as a case study for the Master's thesis. The company is the pioneer organisation in the sphere of postal and logistics maintenance in Finland [83]. Every day the corporation generates a tremendous amount of data regarding:

- outlining and implementation of work shifts for the employees across the entire country,
- an extensive variety of manifold delivery routes,
- sorting procedures, and so on.

Posti Timestamp Verification data set (PTVD log) is employed as a primary data source, representing an event log that comprises all planned and later completed work shifts of Posti Group's mail carriers during 2017. Each work shift in the data set designated a sequence of postal tasks with the beginning and end timestamps where every job is assigned to own identification number.
Inspired by valuable insights, which can be deduced from this data source, and scrutinisation of concepts and strategies in the field of process mining, this paper investigates the following issues:

1. acquiring supervised machine learning model for classification of the eight-hour work shifts of mail carriers and prediction of verification status for incoming work shifts, according to the quality of realisation of each separate task within a work shift,
2. predicting incoming task and its timestamp in a work shift,

3. and forecasting feasible postal task suffixes within work shifts.

Initially, the procedure for determining verification statuses of work shifts has been manually carried out by supervisors at local departments on daily basis. This process was essentially time-consuming since work shifts for every postal worker at the depo had to be monitored autonomously. To optimise this process, convolutional long short-term memory neural network was scrutinised as a baseline approach to implement classifications of the work shifts and forecast the conformation state for each novel work shift.

The primary achievement of this paper is development and enhancement of the convolutional long short-term memory neural network with the best-derived performance on a broad scope of evaluation measures, including classification accuracy, precision, recall, F-score, the area under the receiver operating characteristic curve score, and binary cross-entropy loss, by utilising the capabilities of a novel hyperparameter optimisation tool, named Talos. In order to achieve the established target, the experiments regarding the network depth analysis, i.e., distinguishing the optimal number of hidden layers and hidden nodes per each layer, and hyperparameter tuning for detecting optimal learning rate, batch size, optimiser type, dropout probability, activation function in the hidden layers, and the number of training epochs, were conducted to deduce the CNN LSTM network with the highest-observed performance. According to the wide range of the retrieved results, elaborated model represented the sophisticated convolutional long short-term memory neural network with two hidden layers and 256 units per each layer that also employed the learning rate of 0.0001, the batch size of 128, dropout probability of 0.5, AdaMax optimiser, ELU activation function in the hidden layers, and 1000 epochs for training. This model obtained the exceptional results on the evaluation measures: the prediction accuracy of 86.03%, the precision of 85.45%, the recall of 85.71%, the F-score of 85.58%, the area under the ROC curve score of 86.68%, and the binary cross-entropy of 15.76%. Furthermore, the ultimately developed CNN LSTM network was compared with the state-of-the-art machine learning techniques such as support vector machine, logistic regression, random forest, adaptive boosting, and k-nearest neighbours (k-NN) classifiers. The developed model managed to outperform all of the indicated approaches from 9.04% to 27.24% across the whole set of evaluation criteria.

The devised convolutional LSTM neural network is going to be integrated into a differentiable neural computer and employed by Posti Group Oyj in a software as a service application for the classification and prediction of the verification statuses of work shifts, contributing to the optimisation of supervisors' workload (**Figure 14**). Moreover, supplementary data analyses were executed with an eye to predict incoming postal task within a work shift and its timestamp, using LSTM neural nets as a baseline method. The long short-term memory neural network with two hidden layers and 128 nodes per each layer gained the highest performance in terms of the temporal and operational prediction methods. The corresponding research algorithm provided significantly better results on PTVD log, compared to existing base methods: activity prediction techniques by D. Breuker et al. [19] and J. Evermann et al. [49], and time prediction approach by W. van der Aalst, H. Schonenberg, H. and M. Song [177].

Conclusively, the prediction of the postal tasks suffixes in the work shifts was explored where the LSTM-based approach attained significantly better output than the specialised for this kind of problems algorithm, ProM framework [134]. Besides, a constraint for LSTM models was discovered: if a work shift includes a substantial number of same jobs, then a model forecasts an extensively long series of identical tasks.

References

[1] Gulli A. and Pal S. *Deep Learning with Keras*. Packt Publishing Ltd., February 2017. ISBN: 978-1-78712-842-2.

[2] H. Aa. *Comparing and Aligning Process Representations*. Foundations and Technical Solutions. Springer International Publishing AG, 2018.

[3] S. Abdul-Wahab and S. Al-Alawi. Assessment and prediction of tropospheric ozone concentration levels using artificial neural networks. *Environmental Modelling and Software*, 17(3):219–228, 2002.

[4] M. Abdulmajid and P. Jae-Young. Deep recurrent neural networks for human activity recognition. *Sensors*, 17(11), 2017.

[5] A. Agarap. Deep learning using rectified linear units (relu), 2018. cite arxiv:1803.08375Comment: 7 pages, 11 figures, 9 tables.

[6] J. Ali, R. Khan, Ahmad N., and I. Maqsood. Random forests and decision trees. *IJCSI International Journal of Computer Science Issues*, 9(3), 2012.

[7] A. Argyriou, T. Evgeniou, and M. Pontil. Multi-task feature learning. In B. Schölkopf, J. Platt, and T. Hoffman, editors, *Advances in Neural Information Processing Systems 19*, pages 41–48. MIT Press, 2007.

[8] M. Aufaure and E. Zimányi, editors. *Business Intelligence - Second European Summer School, eBISS 2012, Brussels, Belgium, July 15-21, 2012, Tutorial Lectures*, volume 138 of *Lecture Notes in Business Information Processing*. Springer, 2013.

[9] T.A. AZIM and A. Sarah. *Composing Fisher kernels from deep neural models: A practitioner's approach*. Springer Science and Business Media, 2018.

[10] B. Bakker. Reinforcement learning with long short-term memory. In *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 1475–1482, 2001.

[11] E. Bartocci, Y. Falcone, A. Francalanza, and G. Reger. Introduction to runtime verification. In *Lectures on Runtime Verification*, volume 10457 of *Lecture Notes in Computer Science*, pages 1–33. Springer, 2018.

[12] E. Baum and D. Haussler. What size net gives valid generalization? In D. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 81–90. Morgan-Kaufmann, 1989.

[13] Y. Bengio, P. Simard, and P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Trans. Neural Networks*, 5(2):157–166, 1994.

[14] F. Bianchi, E. Maiorino, M. Kampffmeyer, A. Rizzi, and R. Jenssen. *Recurrent Neural Networks for Short-Term Load Forecasting.* Springer International Publishing, 2017.

[15] M. Bianchini, M. Maggini, F. Scarselli, and L. Jain. Advances in neural information processing paradigms. In *Innovations in Neural Information Paradigms and Applications*, pages 1–7. 2009.

[16] G. Bonaccorso. Pakt Publishing Ltd., Birmingham, UK, 2018.

[17] L. Bottou. *Stochastic Gradient Descent Tricks*, pages 421–436. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.

[18] L. Breiman. Random forests. *Mach. Learn.*, 45(1):5–32, October 2001.

[19] D. Breuker, M. Matzner, P. Delfmann, and J. Becker. Comprehensible predictive models for business processes. *MIS Q.*, 40(4):1009–1034, December 2016.

[20] Martin D. Buhmann and M. D. Buhmann. *Radial Basis Functions.* Cambridge University Press, New York, NY, USA, 2003.

[21] J. Carney and P. Cunningham. The epoch interpretation of learning. Technical report, Trinity College Dublin, Department of Computer Science, 1998.

[22] T. Chai and R. R. Draxler. Root mean square error (rmse) or mean absolute error (mae)? arguments against avoiding rmse in the literature. *Geoscientific Model Development*, 7(3):1247–1250, 2014.

[23] J. Chang-Xue, Z. Yu, and A. Kusiak. Selection and validation of predictive regression and neural network models based on designed experiments. *IIE Transactions*, 38(1):13–23, 2006.

[24] Z. Che, S. Purushotham, K. Cho, D. Sontag, and Y. Liu. Recurrent neural networks for multivariate time series with missing values. *Scientific Reports*, 8, 06 2016.

[25] Freeman D. Chio C. *Machine Learning and Security: Protecting Systems with Data and Algorithms.* O'Reilly Media Inc., February 2018. ISBN: 978-1-491-97990-7.

[26] K. Cho, B. Merrienboer, D. Bahdanau, and Y. Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*, abs/1409.1259, 2014.

[27] F. Chollet. *Deep Learning with Python.* Manning Publications Co., Greenwich, CT, USA, 1st edition, 2017.

[28] P. Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection.* Springer, Berlin, 2012.

[29] P. Cichosz. *Data Mining Algorithms: Explained Using R.* John Wiley and Sons,Ltd, 1st edition, 2015.

[30] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 48(1-3):253–285, September 2002.

[31] M. Collins, R. Schapire, and Y. Singer. Logistic regression, adaboost and bregman distances. *Mach. Learn.*, 48(1-3):253–285, September 2002.

[32] A. Comrie. Comparing neural networks and regression models for ozone forecasting. *Journal of the Air & Waste Management Association*, 47(6):653–663, 1997.

[33] M. Copeland, J. Soh, A. Puca, M. Manning, and D. Gollob. *Microsoft Azure.* Apress, 2015.

[34] P. Dangeti. *Statistics for Machine Learning: Techniques for Exploring Supervised, Unsupervised, and Reinforcement Learning Models with Python and R.* Packt Publishing, 2017.

[35] Y. Dauphin, R. Pascanu, C. Gulcehre, K. Cho, S. Ganguli, and Y. Bengio. Identifying and attacking the saddle point problem in high-dimensional non-convex optimization. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 2933–2941. Curran Associates, Inc., 2014.

[36] M. de Leoni, W. van der Aalst, and M. Dees. A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. *Information Systems*, 56:235 – 257, 2016.

[37] C. Debruyne, H. Panetto, G. Weichhart, P. Bollen, I. Ciuciu, M. Vidal, and R. Meersman, editors. *On the Move to Meaningful Internet Systems. OTM 2017 Workshops - Confederated International Workshops, EI2N, FBM, ICSP, Meta4eS, OTMA 2017 and ODBASE Posters 2017, Rhodes, Greece, October 23-28, 2017, Revised Selected Papers*, volume 10697 of *Lecture Notes in Computer Science*. Springer, 2018.

[38] N. Denz. *Process-oriented analysis and validation of multi-agent-based simulations.* PhD thesis, Uni Hamburg, 2013.

[39] F. DErrico, I. Poggi, A. Vinciarelli, and A. Vincze, editors. *Conflict and Multimodal Communication.* Springer International Publishing, 2015.

[40] C. Di Francescomarino, M. Dumas, F.M. Maggi, and I. Teinemaa. Clustering-based predictive process monitoring. *CoRR*, abs/1506.01428, 2015.

[41] C. Dichev and G. Agre, editors. *Artificial Intelligence: Methodology, Systems, and Applications - 17th International Conference, AIMSA 2016, Varna, Bulgaria, September 7-10, 2016, Proceedings*, volume 9883 of *Lecture Notes in Computer Science*. Springer, 2016.

[42] A. Dieng, C. Wang, J. Gao, and J. Paisley. Topicrnn: A recurrent neural network with long-range semantic dependency. *CoRR*, abs/1611.01702:1–3, 2016.

[43] S. Dipanjan, B. Raghav, and S. Tushar. *Practical Machine Learning with Python: A Problem-Solver's Guide to Building Real-World Intelligent Systems*. Apress, Berkely, CA, USA, 1st edition, 2017.

[44] C. Djork-Arné, T. Unterthiner, and S. Hochreiter. Fast and accurate deep network learning by exponential linear units (elus). *CoRR*, abs/1511.07289, 2015.

[45] R. Drumond, B. Marques, C. Vasconcelos, and E. Clua. Peek - an lstm recurrent network for motion classification from sparse data. In *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP,*, pages 215–222. INSTICC, SciTePress, 2018.

[46] W. Duch, J. Kacprzyk, E. Oja, and S. Zadrozny, editors. *Artificial Neural Networks: Formal Models and Their Applications - ICANN 2005, 15th International Conference, Warsaw, Poland, September 11-15, 2005, Proceedings, Part II*, volume 3697 of *Lecture Notes in Computer Science*. Springer, 2005.

[47] D. Eck and J. Schmidhuber. Finding temporal structure in music: blues improvisation with LSTM recurrent networks. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing, NNSP 2002, Martigny, Valais, Switzerland, September 4-6, 2002.*, pages 747–756, 2002.

[48] J. Eder, H. Pichler, and S. Vielgut. A technique for the prediction of deadline-violations in inter-organizational business processes. In *Proceedings of the 2007 Conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS'2006*, pages 57–71, Amsterdam, The Netherlands, The Netherlands, 2007. IOS Press.

[49] J. Evermann, J.R. Rehse, and P. Fettke. A deep learning approach for predicting process behaviour at runtime. In M. Dumas and M. Fantinato, editors, *Business Process Management Workshops*, pages 327–338, Cham, 2017. Springer International Publishing.

[50] S. Fahlman and C. Lebiere. Advances in neural information processing systems 2. chapter The Cascade-correlation Learning Architecture, pages 524–532. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.

[51] H. Fan. *Sequential Frameworks for Statistics-based Value Function Representation in Approximate Dynamic Programming*. PhD thesis, Arlington, TX, USA, 2008. AAI3320119.

[52] T. Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[53] T. Fawcett. An introduction to roc analysis. *Pattern Recogn. Lett.*, 27(8):861–874, June 2006.

[54] D. Ferreira. *A Primer on Process Mining*. Springer International Publishing, 2017.

[55] D. Florian, K. Barkaoui, and S. Dustdar. Merging computer log files for process mining: An artificial immune system technique. In *Business Process Management Workshops*, BPM 2011 International Workshops, pages 99–111. Springer-Verlag Berlin Heidelberg, 2012.

[56] F. Folino, M. Guarascio, and L. Pontieri. A prediction framework for proactively monitoring aggregate process-performance indicators. In *2015 IEEE 19th International Enterprise Distributed Object Computing Conference*, pages 128–133, Sept 2015.

[57] Y. Gal and Z. Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pages 1027–1035, USA, 2016. Curran Associates Inc.

[58] M. Galar, A. Fernández, E. Barrenechea, H. Bustince, and F. Herrera. Dynamic classifier selection for one-vs-one strategy: Avoiding non-competent classifiers. *Pattern Recogn.*, 46(12):3412–3424, December 2013.

[59] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Comput.*, 4(1):1–58, January 1992.

[60] F. Gers and J. Schmidhuber. Recurrent nets that time and count. In *IJCNN (3)*, pages 189–194, 2000.

[61] F. Gers, N. Schraudolph, and J. Schmidhuber. Learning precise timing with LSTM recurrent networks. *Journal of Machine Learning Research*, 3:115–143, 2002.

[62] F.A. Gers and E. Schmidhuber. Lstm recurrent networks learn simple context-free and context-sensitive languages. *Trans. Neur. Netw.*, 12(6):1333–1340, November 2001.

[63] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[64] A. Graves. *Supervised Sequence Labelling with Recurrent Neural Networks*, volume 385 of *Studies in Computational Intelligence*. Springer, 2012.

[65] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Machine Learning, Proceedings of the Twenty-Third International Conference (ICML 2006), Pittsburgh, Pennsylvania, USA, June 25-29, 2006*, pages 369–376, 2006.

[66] A. Graves, M. Liwicki, H. Bunke, J. Schmidhuber, and S. Fernández. Unconstrained on-line handwriting recognition with recurrent neural networks. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, pages 577–584. Curran Associates, Inc., 2008.

[67] A. Graves, A. Mohamed, and G. Hinton. Speech recognition with deep recurrent neural networks. In *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013, Vancouver, BC, Canada, May 26-31, 2013*, pages 6645–6649, 2013.

[68] A. Graves and J. Schmidhuber. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks*, 18(5-6):602–610, 2005.

[69] A. Gron. *Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems*. O'Reilly Media, Inc., 1st edition, 2017.

[70] J. Gulden, I. Reinhartz-Berger, R. Schmidt, S. Guerreiro, W. Guedria, and P. Bera. *Enterprise, Business-Process and Information Systems Modeling: 19th International Conference, BPMDS 2018, 23rd International Conference, EMMSAD 2018*. Springer Publishing Company, Incorporated, 2018.

[71] Vassileva J. Habib, S. a d and Mühlhäuser M. Mauw, S. and. *Trust Management X: 10th IFIP WG 11.11 International Conference, IFIPTM 2016, Darmstadt, Germany, July 18-22, 2016, Proceedings (IFIP Advances in Information and Communication Technology Book 473)*. Springer, 2016.

[72] R. Hahnloser, R. Sarpeshkar, M. Mahowald, R. Douglas, and S. Seung. Erratum: Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit. *Nature*, 405(6789):947–951, jun 2000.

[73] J. Hanley and B. Mcneil. The meaning and use of the area under a receiver operating characteristic (roc) curve. *Radiology*, 143:29–36, 05 1982.

[74] S. Hanson, J. Cowan, and C Giles, editors. *Advances in Neural Information Processing Systems 5, [NIPS Conference]*, San Francisco, CA, USA, 1993. Morgan Kaufmann Publishers Inc.

[75] R. Hecht-Nielsen. *Neurocomputing*. Addison-Wesley, 1990.

[76] J. Hertz, R. Palmer, and A. Krogh. *Introduction to the Theory of Neural Computation*. Perseus Publishing, 1st edition, 1991.

[77] S. Hochreiter. *Untersuchungen zu Dynamischen Neuronalen Netzen*. PhD thesis, Institut für Informatik, Technische Universität Münch, 1991.

[78] S. Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. 6:107–116, 04 1998.

[79] S. Hochreiter, Y. Bengio, P. Frasconi, and J. Schmidhuber. Gradient flow in recurrent nets: The difficulty of learning LongTerm dependencies. In *A Field Guide to Dynamical Recurrent Networks*. IEEE Press, 2009.

[80] S. Hochreiter, M. Heusel, and K. Obermayer. Fast model-based protein homology detection without alignment. *Bioinformatics*, 23(14):1728–1736, 2007.

[81] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, nov 1997.

[82] S. Hosangadi. Distance measures for sequences. *CoRR*, abs/1208.5713, 2012.

[83] J. Humbley. *Languages for Special Purposes. An International Handbook.* Boston: De Gruyter Mouton, 2018.

[84] A. Ijspeert, T. Masuzawa, and S. Kusumoto, editors. *Biologically Inspired Approaches to Advanced Information Technology, Second International Workshop, BioADIT 2006, Osaka, Japan, January 26-27, 2006, Proceedings*, volume 3853 of *Lecture Notes in Computer Science*. Springer, 2006.

[85] Brownlee J. How to use word embedding layers for deep learning with keras. October 2017.

[86] L. Jain, S. Patnaik, and N. Ichalkaranje. *Intelligent Computing, Communication and Devices: Proceedings of ICCD 2014, Volume 2*. Springer Publishing Company, Incorporated, 2014.

[87] M. Jain, M. Mathew, and C.V. Jawahar. Unconstrained scene text and video text recognition for arabic script. *CoRR*, abs/1711.02396, 2017.

[88] A. Janczak. *Identification of Nonlinear Systems Using Neural Networks and Polynomial Models: A Block-Oriented Approach (Lecture Notes in Control and Information Sciences)*. Springer, 2004.

[89] D. Kingma and J. Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

[90] S. Kombrink, T. Mikolov, M. Karafiát, and L. Burget. Recurrent neural network based language modeling in meeting recognition. In *INTERSPEECH 2011, 12th Annual Conference of the International Speech Communication Association, Florence, Italy, August 27-31, 2011*, pages 2877–2880, 2011.

[91] M. Kotila. Hyperparameter optimization with keras. *Towards Data Science*, May 2018.

[92] M. Koutny, J. Kleijn, and W. Penczek, editors. *Transactions on Petri Nets and Other Models of Concurrency XII*, volume 10470 of *Lecture Notes in Computer Science*. Springer, 2017.

[93] A. Krogh and J. Hertz. A simple weight decay can improve generalization. In J. Moody, S. Hanson, and R. Lippmann, editors, *Advances in Neural Information Processing Systems 4*, pages 950–957. Morgan-Kaufmann, 1992.

[94] M. Kroiss. *Predicting the Lineage Choice of Hematopoietic Stem Cells*. Springer Fachmedien Wiesbaden, 2016.

[95] G. Lakshmanan, D. Shamsi, Y. Doganata, M. Unuvar, and R. Khalaf. A markov prediction model for data-driven semi-structured business processes. *Knowledge and Information Systems*, 42(1):97–126, Jan 2015.

[96] K. Lang, A. Waibel, and G. Hinton. A time-delay neural network architecture for isolated word recognition. *Neural Networks*, 3(1):23–43, 1990.

[97] J. Lawrence and J. Fredrickson. *BrainMaker Users Guide and Reference Manual*. , California Scientific Software, Nevada City, CA, 7 edition, 1998.

[98] M. Lawrence and A. Petterson. *Getting Started With Brain Maker: Neural Network Simulation Software User's Guide and Reference Manual/Introduction to Neural Networks and Disk*. California Scientific Software, 1990.

[99] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[100] M. Lee. Fast text classification using sequential sampling processes. In *Australian Joint Conference on Artificial Intelligence*, 2001.

[101] R. Lee. *Applied Computing & Information Technology*. Springer International Publishing, 2016.

[102] M. Leemans and W. van der Aalst. Process mining in software systems: Discovering real-life business transactions and process models from distributed systems. In *MoDELS*, pages 44–53. IEEE Computer Society, 2015.

[103] T. Lei and Y. Artzi, Y. Zhang. Training rnns as fast as cnns. pages 1–8, 2018. Sixth International Conference on Learning Representations.

[104] A. Leontjeva, R. Conforti, C. Di Francescomarino, M. Dumas, and F.M. Maggi. Complex symbolic sequence encodings for predictive monitoring of business processes. In H. Motahari-Nezhad, J. Recker, and M. Weidlich, editors, *Business Process Management*, pages 297–313, Cham, 2015. Springer International Publishing.

[105] L. Liang and Y. Shu. Deep automated multi-task learning. *CoRR*, abs/1709.05554, 2017.

[106] T. Lin, B. Horne, P. Tiňo, and C. Giles. Learning long-term dependencies in NARX recurrent neural networks. *IEEE Trans. Neural Networks*, 7(6):1329–1338, 1996.

[107] R. Lippmann. An introduction to computing with neural nets. *IEEE ASSP Magazine*, pages 4–22, apr 1987.

[108] Z. Lipton. *A Critical Review of Recurrent Neural Networks for Sequence Learning*, volume abs/1506.00019. 2015.

[109] M. Liwicki and H. Bunke. *Recognition of Whiteboard Notes - Online, Offline and Combination*, volume 71 of *Series in Machine Perception and Artificial Intelligence*. WorldScientific, 2008.

[110] X. Lu, M. Nagelkerke, D. Wiel, and D. Fahland. Discovering interacting artifacts from ERP systems. *IEEE Trans. Services Computing*, 8(6):861–873, 2015.

[111] L. Ly, F. Maggi, M. Montali, S. Rinderle-Ma, and W. van der Aalst. Compliance monitoring in business processes: Functionalities, application, and tool-support. *Inf. Syst.*, 54:209–234, 2015.

[112] X. Ma and Zaobin G. Lu, H. Implicit trust and distrust prediction for recommender systems. In *Web Information Systems Engineering WISE 2015*, pages 185–199. Springer International Publishing Switzerland, 2015.

[113] F. Maggi, C. Di Francescomarino, M. Dumas, and C. Ghidini. Predictive monitoring of business processes. In M. Jarke, J. Mylopoulos, C. Quix, C. Rolland, Y. Manolopoulos, H. Mouratidis, and J. Horkoff, editors, *Advanced Information Systems Engineering*, pages 457–472. Springer International Publishing, 2014.

[114] M. Mahsereci, L. Balles, C. Lassner, and P. Hennig. Early stopping without a validation set. *CoRR*, abs/1703.09580, 2017.

[115] C. Manning. Part-of-speech tagging from 97% to 100%: Is it time for some linguistics? In *Proceedings of the 12th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part I*, CICLing'11, pages 171–189, Berlin, Heidelberg, 2011. Springer-Verlag.

[116] R. Mans, W. van der Aalst, and R. Vanwersch. *Process Mining in Healthcare - Evaluating and Exploiting Operational Healthcare Processes*. Springer Briefs in Business Process Management. Springer, 2015.

[117] A. Mansur and M. Tsunenori. Dynamic bus travel time prediction using an ann-based model. In *Proceedings of the 12th International Conference on Ubiquitous Information Management and Communication*, IMCOM '18, pages 20:1–20:8, New York, NY, USA, 2018. ACM.

[118] G. Marchandani and W. Cao. On hidden nodes for neural nets. *IEEE Transactions on Circuits and Systems*, 36(5):661–664, 1989.

[119] M. Mehdipour-Ghazi, M. Nielsen, A. Pai, J. Cardoso, M. Modat, S. Ourselin, and L. Sørensen. Robust training of recurrent neural networks to handle missing data for disease progression modeling. *CoRR*, abs/1808.05500, 2018.

[120] T. Mikolov, A. Joulin, S. Sumit Chopra, M. Mathieu, and M. Ranzato. Learning longer memory in recurrent neural networks. *CoRR*, abs/1412.7753, 2014.

[121] J. Mizera-Pietraszko, Y. Chung, and P. Pichappan, editors. *Advances in Digital Technologies - Proceedings of the 7th International Conference on Applications of Digital Information and Web Technologies, Keelung, Taiwan, March 29-31, 2016*, volume 282 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2016.

[122] G. Montavon, G. Orr, and K. Müller, editors. *Neural Networks: Tricks of the Trade*. Springer Berlin Heidelberg, 2012.

[123] J. Moreira, A. Carvalho, and T. Horvath. *A General Introduction to Data Analytics*. Wiley-Interscience, 2018.

[124] M. Mozer, M. Jordan, and T. Petsche, editors. *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*. MIT Press, 1997.

[125] A. Mucherino, P. Papajorgji, and P. Pardalos. *k-Nearest Neighbor Classification*, pages 83–106. Springer New York, New York, NY, 2009.

[126] V. Nath. *Proceedings of the International Conference on Nano-electronics, Circuits & Communication Systems*. Springer Publishing Company, Incorporated, 1st edition, 2017.

[127] F. Naumann and M. Herschel. *An Introduction to Duplicate Detection*. Morgan and Claypool Publishers, 2010.

[128] N. Navarin, B. Vincenzi, M. Polato, and A. Sperduti. Lstm networks for data-aware remaining time prediction of business process instances. *CoRR*, abs/1711.03822, 2017.

[129] Y. Nesterov. A method of solving a convex programming problem with convergence rate O(1/sqr(k)). *Soviet Mathematics Doklady*, 27:372–376, 1983.

[130] S. Nurcan, P. Soffer, M. Bajec, and J. Eder, editors. *Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*, volume 9694 of *Lecture Notes in Computer Science*. Springer, 2016.

[131] V. Pham, C. Kermorvant, and J. Louradour. Dropout improves recurrent neural networks for handwriting recognition. *CoRR*, abs/1312.4569, 2013.

[132] H. Phan, P. Koch, F. Katzberg, M. Maaß, and A. Mazur, R.and Mertins. Audio scene classification with deep recurrent neural networks. *CoRR*, abs/1703.04770, 2017.

[133] G. Pitsilis, X. Zhang, and W. Wang. Clustering recommenders in collaborative filtering using explicit trust information. In *Trust Management V: 5th IFIP WG 11.11 International Conference* , pages 82–98. IFIP International Federation for Information Processing, 2011.

[134] M. Polato, A. Sperduti, A. Burattin, and M. Leoni. Time and activity sequence prediction of business process instances. pages 4–9, 02 2016.

[135] J. Quinlan. Learning decision tree classifiers. *ACM Comput. Surv.*, 28(1):71–72, March 1996.

[136] R. Radiuk. Impact of training set batch size on the performance of convolutional neural networks for diverse datasets. *Information Technology and Management Science*, 20(1):20–24, 2017.

[137] K. Raju and D. Kumar. *Impact of Climate Change on Water Resources: With Modeling Techniques and Case Studies (Springer Climate).* Springer, 2017.

[138] A. Ray, S. Rajeswar, and S. Chaudhury. A hypothesize-and-verify framework for text recognition using deep recurrent neural networks. *CoRR*, abs/1502.07540, 2015.

[139] Ryan Rifkin and Aldebaro Klautau. In defense of one-vs-all classification. *J. Mach. Learn. Res.*, 5:101–141, December 2004.

[140] A. Rogge-Solti and M. Weske. Prediction of business process durations using non-markovian stochastic petri nets. *Inf. Syst.*, 54(C):1–14, December 2015.

[141] R. Rojas. *Neural Networks: A Systematic Introduction.* Springer-Verlag, Berlin, Heidelberg, 1996.

[142] S. Ruder. An overview of gradient descent optimization algorithms, 2016. cite arxiv:1609.04747Comment: Added derivations of AdaMax and Nadam.

[143] R. Saidi, S. Aridhi, E. Nguifo, and M. Maddouri. Feature extraction in protein sequences classification: A new stability measure. In *Proceedings of the ACM Conference on Bioinformatics, Computational Biology and Biomedicine*, BCB '12, pages 683–689, New York, NY, USA, 2012. ACM.

[144] H. Salehinejad, J. Baarbe, S. Sankar, J. Barfett, E. Colak, and S. Valaee. Recent advances in recurrent neural networks. *CoRR*, abs/1801.01078, 2018.

[145] D. Sarkar, R. Bali, and T. Ghosh. *Hands-On Transfer Learning with Python: Implement advanced deep learning and neural network models using TensorFlow and Keras.* Packt Publishing, 2018.

[146] S. Scardapane, D. Comminiello, A. Hussain, and A. Uncini. Group sparse regularization for deep neural networks. *Neurocomputing*, 241:81 – 89, 2017.

[147] J. Schmidhuber. Learning complex, extended sequences using the principle of history compression. *Neural Computation*, 4(2):234–242, 1992.

[148] J. Schmidhuber, F. Gers, and D. Eck. Learning nonregular languages: A comparison of simple recurrent networks and LSTM. *Neural Computation*, 14(9):2039–2041, 2002.

[149] J. Schumann and Y. Liu. *Applications of Neural Networks in High Assurance Systems*. Springer Publishing Company, Incorporated, 1st edition, 2010.

[150] A. Senderovich, M. Weidlich, A. Gal, and A. Mandelbaum. Queue mining – predicting delays in service processes. In M. Jarke, J. Mylopoulos, Christoph Quix, Colette Rolland, Y. Manolopoulos, H. Mouratidis, and J. Horkoff, editors, *Advanced Information Systems Engineering*, pages 42–57, Cham, 2014. Springer International Publishing.

[151] J. Seo, Y.and Giampapa and K. Sycara. Text classification for intelligent portfolio management. Technical Report CMU-RI-TR-02-14, Carnegie Mellon University, Pittsburgh, P.A., May 2002.

[152] Y. Shkarupa, M. Roberts, and S. Matthia. Offline handwriting recognition using lstm recurrent neural networks. 1:88, 11 2016.

[153] S. Si, H. Zhang, Z. Keerthi, D. Mahajan, I. Dhillon, and C. Hsieh. Gradient boosted decision trees for high dimensional sparse output. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3182–3190, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR.

[154] S. Sønderby and O. Winther. Protein secondary structure prediction with long short term memory networks. *CoRR*, abs/1412.7828, 2014.

[155] S. van der Spoel, M. van Keulen, and C. Amrit. Process Prediction in Noisy Data Sets: A Case Study in a Dutch Hospital. In W. van der Aalst, J. Mylopoulos, M. Rosemann, M.J. Shaw, C. Szyperski, P. Cudre-Mauroux, P. Ceravolo, and D. Gašević, editors, *2nd International Symposium on Data-Driven Process Discovery and Analysis (SIMPDA)*, volume LNBIP-162 of *Data-Driven Process Discovery and Analysis*, pages 60–83, Campione d'Italia, Italy, June 2012. Springer.

[156] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

[157] R. Staudemeyer and C. Omlin. Evaluating performance of long short-term memory recurrent neural networks on intrusion detection data. page 218, 10 2013.

[158] B. Su and S. Lu. Accurate scene text recognition based on recurrent neural network. In *Computer Vision – ACCV 2014*, pages 35–48. Springer International Publishing, 2015.

[159] V. Subramanian. *Deep Learning with PyTorch: A practical approach to building neural network models using PyTorch*. Packt Publishing, 2018.

[160] S. Sun, H. Liu, H. Lin, and A. Abraham. Twitter part-of-speech tagging using pre-classification hidden markov model. In *SMC*, pages 1118–1123. IEEE, 2012.

[161] W. Susilo and R. Reyhanitabar, editors. *Provable Security - 7th International Conference, ProvSec 2013, Melaka, Malaysia, October 23-25, 2013. Proceedings*, volume 8209 of *Lecture Notes in Computer Science*. Springer, 2013.

[162] I. Sutskever. Training recurrent neural networks. *University of Toronto, Toronto, Ont., Canada*, 2013.

[163] I. Sutskever, J. Martens, and G. Hinton. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1017–1024, 2011.

[164] Y. Tan. *Anti-Spam Techniques Based on Artificial Immune System*. CRC Press, 2016.

[165] Y. Tan, Y. Shi, and Q. Tang, editors. *Data Mining and Big Data - Third International Conference, DMBD 2018, Shanghai, China, June 17-22, 2018, Proceedings*, volume 10943 of *Lecture Notes in Computer Science*. Springer, 2018.

[166] N. Tax, N. Sidorova, R. Haakma, and W. van der Aalst. Mining process model descriptions of daily life through event abstraction. *CoRR*, abs/1705.10202, 2017.

[167] N. Tax, I. Verenich, M. Rosa, and M. Dumas. Predictive business process monitoring with lstm neural networks. In Eric Dubois and Klaus Pohl, editors, *CAiSE*, volume 10253 of *Lecture Notes in Computer Science*, pages 477–492. Springer, 2017.

[168] A. Tchamova, J. Dezert, P. Konstantinova, N. Bocheva, B. Genova, and M. Stefanova. Human heading perception based on form and motion combination. In *2018 Innovations in Intelligent Systems and Applications, INISTA 2018, Thessaloniki, Greece, July 3-5, 2018*, pages 1–8, 2018.

[169] I. Teinemaa. Classifying business process execution traces with lstm. Technical report, Tartu University, December 2015.

[170] E. Teniente and M. Weidlich, editors. *Business Process Management*, volume 308 of *Lecture Notes in Business Information Processing*. Springer, 2018.

[171] T. Tieleman and G. Hinton. Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. Technical report, 2012.

[172] W. van der Aalst. Process discovery: Capturing the invisible. *IEEE Comp. Int. Mag.*, 5(1):28–41, February 2010.

[173] W. van der Aalst. *Process Mining: Data Science in Action*. Springer Publishing Company, Incorporated, 2nd edition, 01 2016.

[174] W. van der Aalst, J. Mylopoulos, M. Rosemann, M. Shaw, C. Szyperski, P. Cudre-Mauroux, P. Ceravolo, and D. Gasevic. *Data-Driven Process Discovery and Analysis : Second IFIP WG 2.6, 2.12 International Symposium, SIMPDA 2012, Campione d'Italia, Italy, June 18-20, 2012*, volume 162. 01 2013.

[175] W. van der Aalst, M. Rosemann, and M. Dumas. Deadline-based escalation in process-aware information systems. *Decis. Support Syst.*, 43(2):492–511, March 2007.

[176] W. van der Aalst, V. Rubin, H. Verbeek, B. van Dongen, E. Kindler, and C. Günther. Process mining: a two-step approach to balance between underfitting and overfitting. *Software and System Modeling*, 9(1):87–111, 2010.

[177] W. van der Aalst, H. Schonenberg, and M. Song. Time prediction based on process mining. *Inf. Syst.*, 36(2):450–475, 2011.

[178] W. van der Aalst, A. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[179] B. van Dongen, R. Crooy, and W. van der Aalst. Cycle time prediction: When will this case finally be finished? In R. Meersman and Z. Tari, editors, *On the Move to Meaningful Internet Systems: OTM 2008*, pages 319–336, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[180] B. van Dongen, J. Luin, and E. Verbeek. Process mining in a multi-agent auctioning system. In D. Moldt, editor, *Proceedings of the 4th International Workshop on Modelling of Objects, Components, and Agents*, pages 145–160, Turku, June 2006.

[181] J. Wang, S. Rozen, B. Shapiro, D. Shasha, Z. Wang, and M. Yin. New techniques for dna sequence classification. 6:209–218, 06 1999.

[182] L. Wang. *Support Vector Machines: Theory and Applications*, volume 177 of *Studies in Fuzziness and Soft Computing*. Springer Berlin, Heidelberg, Germany, 2005.

[183] X. Wang, W. Jiang, and Z. Luo. Combination of convolutional and recurrent neural network for sentiment analysis of short texts. In *COLING*, 2016.

[184] J. Watt, R. Borhani, and A. Katsaggelos. *Machine Learning Refined: Foundations, Algorithms, and Applications*. Cambridge University Press, 2016.

[185] Cort J. Willmott, Kenji Matsuura, and Scott M. Robeson. Ambiguities inherent in sums-of-squares-based error statistics. 2008.

[186] J. Wu, S. Chanson, and Q. Gao, editors. *Formal Methods for Protocol Engineering and Distributed Systems, FORTE XII / PSTV XIX'99, IFIP TC6 WG6.1 Joint International Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE XII) and Protocol Specification, Testing and Verification (PSTV XIX), October 5-8, 1999, Beijing, China*, volume 156 of *IFIP Conference Proceedings*. Kluwer, 1999.

[187] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhutdinov, R. Zemel, and Y. Bengio. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, abs/1502.03044, 2015.

[188] N. Yadav. *An Introduction to Neural Network Methods for Differential Equations (SpringerBriefs in Applied Sciences and Technology)*. Springer, 2015.

[189] E. Yolaçan. Learning from sequential data for anomaly detection, 2014.

[190] Z. Yu, V. Ramanarayanan, D. Suendermann-Oeft, X. Wang, K. Zechner, L. Chen, J. Tao, A. Ivanou, and Y. Qian. Using bidirectional lstm recurrent neural networks to learn high-level abstractions of sequential features for automated scoring of non-native spontaneous speech. In *2015 IEEE Workshop on Automatic Speech Recognition and Understanding, ASRU 2015, Scottsdale, AZ, USA, December 13-17, 2015*, pages 338–345, 2015.

[191] M. Zeiler. Adadelta: An adaptive learning rate method. *CoRR*, abs/1212.5701, 2012.

[192] K. Zhang, J. Xu, M. Renqiang, G. Jiang, K. Pelechrinis, and H. Zhang. Automated IT system failure prediction: A deep learning approach. In *2016 IEEE International Conference on Big Data (Big Data)*. IEEE, dec 2016.

[193] M. Zweig and G. Campbell. Receiver-operating characteristic (roc) plots: A fundamental evaluation tool in clinical medicine. *Clinical chemistry*, 39:561–77, 05 1993.